

UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

**Uma abordagem *Ensemble Learning* para aprimorar a
predição de energia livre de ligação entre complexos
proteína-ligante**

Patricia Padula Lopes

Dissertação de Mestrado apresentada ao Programa
de Pós-Graduação em Computação da Universi-
dade Federal do Rio Grande - FURG, como requi-
sito parcial para a obtenção do grau de Mestre em
Engenharia de Computação

Orientador: Prof. Dr. Adriano Velasque Werhli
Co-orientadores: Prof. Dr. Andrej Aderhold
Prof. Dr. Karina S. Machado

Rio Grande, 2021

Ficha Catalográfica

L864a Lopes, Patricia Padula.

Uma abordagem *Ensemble Learning* para aprimorar a predição de energia livre de ligação entre complexos proteína-ligante / Patricia Padula Lopes. – 2021.
166 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2021.

Orientador: Dr. Adriano Velasque Werhli.

Coorientador: Dr. Andrej Aderhold.

Coorientadora: Dra. Karina dos Santos Machado.

1. Funções de Pontuação 2. Aprendizado de Máquina 3. *Ensemble*
4. Predição 5. Afinidade de Ligação I. Werhli, Adriano Velasque
II. Aderhold, Andrej III. Machado, Karina dos Santos IV. Título.

CDU 004

Catálogo na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

DISSERTAÇÃO DE MESTRADO

Uma abordagem Ensemble Learning para aprimorar a predição de energia livre de ligação entre complexos proteína-ligante

Patricia Padula Lopes

Banca examinadora:

Adriano V. Werhli

Profa. Dra. Lucianna Helene Santos

Adriano V. Werhli

Prof. Dr. Eduardo Nunes Borges

Adriano V. Werhli

Profa. Dra. Karina dos Santos Machado
Coorientadora

Adriano V. Werhli

Prof. Dr. Andrej Aderhold
Coorientador

Adriano V. Werhli

Prof. Dr. Adriano Velasque Werhli
Orientador

Dedico... aos meus pais.

AGRADECIMENTOS

A minha mãe Maria da Graça, meu pai Rui André, minha irmã Letícia e meu namorado Thiago. A ajuda e o apoio de cada um de vocês foram para mim de valor inestimável.

Aos meus orientadores Adriano e Andrej, obrigado por toda atenção, disponibilidade, incentivo na construção e conclusão deste trabalho.

Aos meus amigos que mesmo distantes sempre estiveram na torcida pela realização deste objetivo.

Em especial, ao Oscar, por todo empenho, dedicação e paciência para me ajudar.

A todos que de alguma forma contribuíram com meu trabalho.

E sobre tudo, a Deus, que me concedeu a oportunidade de chegar até aqui mesmo diante das dificuldades e intempéries encontradas pelo caminho.

*APRENDER é a única coisa de que a mente nunca se cansa, nunca tem medo
e nunca se arrepende.*

— AUTOR DESCONHECIDO

RESUMO

LOPES, Patricia Padula. **Uma abordagem *Ensemble Learning* para aprimorar a predição de energia livre de ligação entre complexos proteína-ligante.** 2021. 166 f. Dissertação de Mestrado. Universidade Federal do Rio Grande - FURG, Rio Grande.

Apesar da intensa pesquisa ao longo dos anos, melhorar a precisão das funções de pontuação para prever a energia livre de ligação em complexos proteína-ligante provou ser uma tarefa desafiadora. Para superar este desafio, várias técnicas de aprendizado de máquina têm sido usadas para estimar os parâmetros que são mais importantes para predição da afinidade entre complexos proteína-ligante. Nesse contexto, este trabalho se propõe a apresentar uma metodologia, cuja finalidade é combinar os resultados da utilização de diferentes técnicas de aprendizado de máquina para seleção de atributos em uma abordagem *Ensemble*, ao empregar a função de pontuação DeltaVinaRF20. Comparando os escores obtidos, utilizando o nosso modelo, em relação ao Poder de Classificação e do Poder de Pontuação e de acordo com a avaliação do CASF-2016, ocupamos a segunda posição deste *ranking*. Para essas métricas, não foi possível igualar o desempenho da nossa função de pontuação a do DeltaVinaRF20. Porém, observamos que a função de pontuação que foi proposta, conseguiu superar várias das funções de escore com licença comercial avaliadas no CASF-2016 em relação a estas duas métricas. Já em relação ao Poder de Ancoragem, conseguimos igualar nosso escore ao obtido pela função de pontuação DeltaVinaRF20, mas utilizando o método *Support Vector Machine* e aplicando várias técnicas computacionais e de aprendizado de máquina como: Ensemble, técnicas de seleção de atributos e otimização de parâmetros.

Palavras-chave: Funções de Pontuação, Aprendizado de Máquina, Ensemble, Predição, Afinidade de Ligação.

ABSTRACT

LOPES, Patricia Padula. **An Ensemble Learning Approach to Enhance Binding Free Energy Prediction between Protein-Binder Complexes**. 2021. 166 f. Relatório de Trabalho Individual. Universidade Federal do Rio Grande - FURG, Rio Grande.

Despite intense research over the years, improving the accuracy of scoring functions to predict free binding energy in protein-ligand complexes has proved to be a challenging task. To overcome this challenge, several machine learning techniques have been used to estimate the parameters that are most important for predicting the affinity between protein-ligand complexes. In this context, this work proposes to present a methodology, the purpose of which is to combine the results of using different machine learning techniques for selecting attributes in an Ensemble approach, by using the DeltaVinaRF20 scoring function. Comparing the scores obtained, using our model, in relation to the Classification Power and the Scoring Power and according to the evaluation of CASF-2016, we occupy the second position of this ranking. For these metrics, it was not possible to match the performance of our scoring function to that of DeltaVinaRF20. However, we observed that the scoring function that was proposed, managed to overcome several of the scoring functions with commercial license assessed in CASF-2016 in relation to these two metrics. In relation to the Anchoring Power, we managed to match our score to that obtained by the DeltaVinaRF20 scoring function, but using the Support Vector Machine method and applying various computational and machine learning techniques such as: Ensemble, attribute selection techniques and parameter optimization.

Keywords: scoring functions, machine learning, Ensemble, prediction, binding affinity.

LISTA DE FIGURAS

Figura 1	Etapas realizadas na metodologia de <i>docking</i> molecular	18
Figura 2	Representação do fluxo de informação em sistemas biológicos. Fonte: Adaptado de (Alberts et al., 2010)	24
Figura 3	Níveis de organização da estrutura molecular de uma proteína. Fonte: Adaptado de (Nelson and Cox, 2018).	25
Figura 4	Estrutura de um aminoácido e as interações para gerar uma proteína. Fonte: Adaptado de (Nelson and Cox, 2018)	27
Figura 5	Escala de valores para K_d . Fonte: Adaptado de (Nelson and Cox, 2018).	29
Figura 6	Representação gráfica da interação com o ligante. Fonte: Adaptado de (Nelson and Cox, 2018)	31
Figura 7	Cinética Enzimática. Fonte: Adaptado de (Alberts et al., 2010).	33
Figura 8	Reta ajustada por mínimos quadrados. Fonte: Adaptado de (Passari et al., 2011).	38
Figura 9	Arquitetura tradicional de <i>Ensemble Learning</i> . Fonte: Adaptado de (Kamber and Pei, 2006).	46
Figura 10	Modelo conceitual: Bagging. Fonte: Adaptado de (Shafiei and Jazayeri-Rad, 2012).	47
Figura 11	Modelo conceitual: <i>Boosting</i> . Fonte: (Merrill et al., 2019).	47
Figura 12	Funcionamento do algoritmo <i>Random Forest</i> . Fonte: Adaptado de (Maheshwari, 2019)	49
Figura 13	Funcionamento do algoritmo <i>Random Forest</i> . Fonte: Adaptado de (Polishchuk, 2015)	50
Figura 14	<i>Feature Selection Decision Tree</i> . Fonte: Adaptado de (Yiu, 2019)	51
Figura 15	Representação de um conjunto de dados	54
Figura 16	Funcionamento do Método PCA. Fonte: Adaptado de (Serrano, 2019)	56
Figura 17	Visão geral do SVR. Fonte: Adaptado de (Sayad, 2019)	57
Figura 18	Distribuição de probabilidade	58
Figura 19	Metodologia aplicada	66
Figura 20	Ranking acumulativo para Δp_{k_d} (1 a 41)	73
Figura 21	Ranking acumulativo para Δp_{k_d} (42 a 82)	74
Figura 22	Ranking acumulativo para Δp_{k_d} (83 a 123)	75
Figura 23	Ranking acumulativo para Δp_{k_d} (124 a 164)	76
Figura 24	Ranking acumulativo para Δp_{k_d} (165 a 205)	77

Figura 25	Ranking acumulativo para p_{k_d} (1 a 41)	78
Figura 26	Ranking acumulativo para p_{k_d} (42 a 82)	79
Figura 27	Ranking acumulativo para p_{k_d} (83 a 123)	80
Figura 28	Ranking acumulativo para p_{k_d} (124 a 164)	81
Figura 29	Ranking acumulativo para p_{k_d} (165 a 205)	82
Figura 30	Poder de pontuação avaliado pelo sistema <i>Ensemble</i> utilizando a média para Δp_{k_d}	84
Figura 31	Poder de pontuação avaliado pelo sistema <i>Ensemble</i> utilizando a mediana para Δp_{k_d}	84
Figura 32	Poder de pontuação avaliado pelo sistema <i>Ensemble</i> utilizando o <i>trim rule</i> para Δp_{k_d}	85
Figura 33	Poder de classificação avaliado pelo sistema <i>Ensemble</i> utilizando a média para Δp_{k_d}	85
Figura 34	Poder de classificação avaliado pelo sistema <i>Ensemble</i> utilizando a mediana para Δp_{k_d}	86
Figura 35	Poder de classificação avaliado pelo sistema <i>Ensemble</i> utilizando o <i>trim rule</i> para Δp_{k_d}	86
Figura 36	Poder de classificação avaliado pelo sistema <i>Ensemble</i> utilizando a média para p_{k_d}	87
Figura 37	Poder de classificação avaliado pelo sistema <i>Ensemble</i> utilizando a mediana para p_{k_d}	88
Figura 38	Poder de classificação avaliado pelo sistema <i>Ensemble</i> utilizando o <i>trim rule</i> para p_{k_d}	88
Figura 39	Poder de pontuação avaliado pelo sistema <i>Ensemble</i> utilizando a média para p_{k_d}	89
Figura 40	Poder de pontuação avaliado pelo sistema <i>Ensemble</i> utilizando a mediana para p_{k_d}	89
Figura 41	Poder de pontuação avaliado pelo sistema <i>Ensemble</i> utilizando o <i>trim rule</i> para p_{k_d}	90
Figura 42	Comparação do <i>Ensemble</i> e dos métodos individualmente para a predição de Δp_{k_d}	91
Figura 43	Comparação do <i>Ensemble</i> e dos métodos individualmente para a predição de p_{k_d}	91
Figura 44	Ranking acumulativo para p_{k_d} (1 a 41)	97
Figura 45	Ranking acumulativo para p_{k_d} (42 a 82)	98
Figura 46	Ranking acumulativo para p_{k_d} (83 a 123)	99
Figura 47	Ranking acumulativo para p_{k_d} (124 a 164))	100
Figura 48	Ranking acumulativo para p_{k_d} (124 a 164))	101
Figura 49	Ranking acumulativo para Δp_{k_d} (42 a))	102
Figura 50	Ranking acumulativo para Δp_{k_d} (42 a 81))	103
Figura 51	Ranking acumulativo para Δp_{k_d} (82 a 123))	104
Figura 52	Ranking acumulativo para Δp_{k_d} (124 a 163))	105
Figura 53	Ranking acumulativo para Δp_{k_d} (164 a 205))	106
Figura 54	Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Classificação	107
Figura 55	Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Pontuação	108

Figura 56	Comparação do modelo otimizado e não otimizado para Δp_{k_d} - Avaliação do Poder de Pontuação	108
Figura 57	Comparação do modelo otimizado e não otimizado para Δp_{k_d} - Avaliação do Poder de Classificação	109
Figura 58	Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Ancoragem	110
Figura 59	Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Ancoragem	110
Figura 60	Comparação do Poder de Classificação para p_{k_d} com o CASF-2016	112
Figura 61	Comparação do Poder de Pontuação para p_{k_d} com o CASF-2016	113
Figura 62	Comparação do Poder de Classificação para Δp_{k_d} com o CASF-2016	114
Figura 63	Comparação do Poder de Pontuação para Δp_{k_d} com o CASF-2016	115
Figura 64	Comparação do Poder de Ancoragem para Δp_{k_d} com o CASF-2016	116
Figura 65	Comparação do Poder de Ancoragem para p_{k_d} com o CASF-2016	117

LISTA DE TABELAS

Tabela 1	Nomenclatura dos 20 aminoácidos essenciais. Fonte: Adaptado de (Verli, 2014)	28
Tabela 2	Funções de pontuação desenvolvidas com métodos ML (<i>Machine Learning</i>) nos últimos 20 anos. Fonte: Adaptado de (Pason and Sotriffer, 2016).	43
Tabela 3	Trabalhos Correlatos - Resumo	64
Tabela 4	Configuração para escolha dos parâmetros do método <i>Decision Tree</i> para seleção de atributos	93
Tabela 5	Configuração para escolha dos parâmetros do método <i>Random Forest</i> para seleção de atributos	93
Tabela 6	Configuração para escolha dos parâmetros no <i>ranking</i> acumulativo . .	94

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
DNA	<i>DeoxyriboNucleic Acid</i>
DT	<i>Decision Tree</i>
EF	<i>Enhancement Factor</i>
EI	Enzima Inibidor
ES	Enzima Substrato
IA	Inteligência Artificial
LASSO	<i>Least Absolute Shrinkage and Selection Operator</i>
ML	<i>Machine Learning</i>
NN	<i>Neural Networks</i>
PLS	<i>Partial Least Squares</i>
RF	<i>Random Forest</i>
RL	Receptor Ligante
RMSD	<i>Root Mean Square Deviation</i>
RNA	<i>RiboNucleic Acid</i>
RS	<i>Spearman Correlation</i>
SASA	<i>Solvent-Accessible Surface Area</i>
SD	<i>Standard Deviation</i>
SVM	<i>Suport Vector Machines</i>

SUMÁRIO

1	INTRODUÇÃO	16
1.1	Objetivo Geral	19
1.2	Objetivos Específicos	19
1.3	Estrutura do Trabalho	20
2	INTRODUÇÃO AO APRENDIZADO DE MÁQUINA	21
2.1	Conceitos Iniciais	21
2.2	Introdução à técnica de seleção de atributos	22
3	CONCEITOS BIOLÓGICOS	24
3.1	O Dogma Central da Biologia Molecular	24
3.2	Proteínas	25
3.3	Aminoácidos: Estrutura e Propriedades	26
3.4	Interações entre Proteínas e Ligantes	27
4	AValiação e Predição de Funções de Escore	36
4.1	Poder de pontuação	37
4.2	Poder de classificação	38
4.3	Poder de ancoragem	39
4.4	Poder de triagem	40
5	ML APLICADO EM DOCKING SCORING PREDICTION	41
5.1	Técnicas de seleção de atributos aplicadas em docking scoring para melhorar a predição da afinidade de ligação	43
5.2	Tipos de modelos <i>Ensemble Learning</i>	45
5.3	Métodos de ML utilizados para seleção de atributos e <i>feature importance</i>	48
5.3.1	<i>Random Forest</i>	49
5.3.2	<i>Decision Tree</i>	50
5.3.3	<i>Gradient Boosting Regression</i>	52
5.3.4	<i>Cross Validation Lasso</i>	53
5.3.5	<i>Principal Component Analysis</i>	55
5.3.6	<i>Support Vector Regression</i>	56
5.3.7	<i>Automatic Relevance Determination Regression</i>	57
5.4	<i>Hyperparameter Tuning</i> : Técnicas de otimização de hiper-parâmetros	59
6	TRABALHOS RELACIONADOS	61
7	METODOLOGIA	65

8	DADOS E SIMULAÇÕES	68
8.1	Base de dados para treinamento e teste	68
8.2	PDBbind refined set 2018	68
8.3	CASF-2016	68
8.4	Conjunto de atributos utilizados	69
8.5	Atributos do DeltaVinaRF20	69
8.6	Atributos do AutoDock Vina	69
8.7	Atributos gerados com RDKit 2D	70
8.8	Atributos alvo	70
9	RESULTADOS	71
9.1	<i>Feature Importance</i> e <i>Cumulative Ranking</i>	71
9.2	Resultados da seleção de atributos utilizando um sistema <i>Ensemble</i>	83
9.3	Resultados da otimização do modelo proposto	92
10	DISCUSSÕES	118
11	CONCLUSÕES	119
	REFERÊNCIAS	121
	ANEXO A CONJUNTO DE DESCRITORES	128
A.1	Descritores do DeltaVina RF20	128
A.2	Descritores do AutoDock Vina	129
A.3	Descritores gerados com RDKit	130
A.4	Gráficos para <i>Feature Importance</i>	130
A.5	Gráficos com a comparação das predições para cada método individualmente combinado a cada grupo de atributos	139
A.6	Gráficos para <i>Feature Importance</i> com otimização de parâmetros	152
A.7	Configuração da escolha dos parâmetros para p_{k_d}	165
A.8	Configuração da escolha dos parâmetros para Δp_{k_d}	166

1 INTRODUÇÃO

As proteínas são tipos de macromoléculas essenciais para vida na Terra (Verli, 2014). A grande maioria das reações químicas que ocorrem em todos os seres vivos do nosso planeta, de alguma forma envolve as proteínas em atividades estruturais, de defesa e de síntese, degradação e reação química em todos os organismos (Watson et al., 2015). A execução de todas estas atividades depende, exclusivamente, da proteína ter função biológica ativa, que por sua vez depende do seu estado nativo, pois sua forma estrutural diz muito a respeito de suas funções bioquímicas no organismo (Almeida, 2016). Essencialmente, a sequência de aminoácidos (estrutura primária) determina como a proteína se dobrará em uma determinada conformação tridimensional nativa (estrutura terciária), a fim de executar sua função biológica (Verli, 2014). De uma forma geral, se compreendermos como a proteína é formada por sua sequência de aminoácidos para determinar a estrutura nativa, isto é, se compreendermos a relação sequência-estrutura, poderemos “desenhar” qualquer proteína. Entretanto, não se conhece ainda como ocorre o processo exato de enovelamento para que seja possível determinar especificamente estas estruturas (Verli, 2014).

De acordo com (Almeida, 2016) apenas 0,17% de todas as proteínas conhecidas (no mundo) tem a sua estrutura determinada. Diante deste cenário, vários pesquisadores, ainda buscam explicações para fenômenos básicos que ocorrem ao longo desse processo para que seja possível determinar a estrutura tridimensional dessas macromoléculas.

Até o momento, muitos esforços ainda são necessários para prever a estrutura nativa de uma proteína a partir de uma sequência primária de aminoácidos. Além disso, os métodos experimentais utilizados para esta finalidade, envolvem um alto custo, demandam um tempo elevado e são complexos (Congreve et al., 2005).

Contudo, mesmo com todos estes desafios estudar a forma tridimensional dessas proteínas é uma questão relevante e envolve pesquisadores de diversas áreas do conhecimento, pois conhecer as estruturas tridimensionais, de maneira completa, em última análise é entender como funciona um ser vivo. Uma das aplicações possíveis, a partir da estrutura 3D é o estudo das interações entre moléculas pequenas, com as proteínas, deste modo permitindo o estudo de fármacos. Uma técnica computacional que permite estudar

moléculas pequenas interagindo com proteínas é o *docking*.

Para exemplificar quais são as etapas e como esta técnica funciona, na Figura 1 é apresentado um esquema de como este processo é realizado (Chaudhary and Mishra, 2016): Primeiramente, no painel A temos a estrutura tridimensional da proteína, a qual é selecionada de banco de dados virtual de moléculas denominado PDB. Na prática, poderíamos obter essa estrutura direto da cristalografia, por predição de estrutura ou modelagem comparativa e os ligantes direto do laboratório de química, nesse caso, nem buscaríamos em base de dados. Contudo, neste exemplo, vamos demonstrar um processo de *docking* onde as macromoléculas estão sendo selecionadas de um banco de dados de proteínas na Internet.

A partir das informações que são armazenadas nesses banco de dados, é possível apontar prováveis regiões de ligação onde uma pequena molécula (ligante) pode se ligar a esse receptor. Baseado nas prováveis regiões de ligação identificadas na etapa anterior, é selecionado no painel B, um conjunto de possíveis candidatos, chamados ligantes. Usualmente ligantes podem ser buscados em bancos de dados de compostos como o ZINC. As diferentes conformações e orientações que determinado ligante pode assumir dentro do sítio de ligação de uma determinada proteína são simuladas por software de docagem molecular como o AutoDock Vina como mostra o painel C. Esse programa analisa as poses obtidas através do ancoramento molecular utilizando funções de pontuação. O receptor pode ter muitos sítios ativos, mas um deve ser selecionado. Assim, quando há interação entre um determinado alvo biológico e diferentes moléculas são comparadas, aquela que gerar menor energia de ligação será referente à molécula que, teoricamente, apresentará melhor atividade biológica e neste caso, será docada na estrutura da proteína, conforme mostra o painel D. O resultado deste processo é uma lista de compostos ordenados segundo sua pontuação.

Em resumo, o *docking* é uma técnica que consiste em duas partes. A busca pelo espaço dos graus de liberdade configuracionais e conformacionais (que envolve o posicionamento do ligante no sítio ativo), e a atribuição de uma pontuação a essas conformações segundo uma função de pontuação.

Uma função de pontuação calcula a afinidade, ou adequação, da ligação proteína-ligante, somando as contribuições de um número de atributos individuais. Cada um desses termos geralmente representa um importante fator energético na ligação proteína-ligante. Neste trabalho, desenvolvemos o método, mas não executamos de fato o processo de *docking*. No geral, desenvolvemos uma função de pontuação que combina diferentes tipos de atributos de proteínas, ligantes e da interação entre eles para estimar a energia de ligação.

Embora as funções de pontuação utilizem poucos recursos computacionais e investimentos financeiros, quando comparadas à aplicação de métodos experimentais de Oliveira Rodrigues et al. (2015), do ponto de vista computacional, elas apresentam alguns

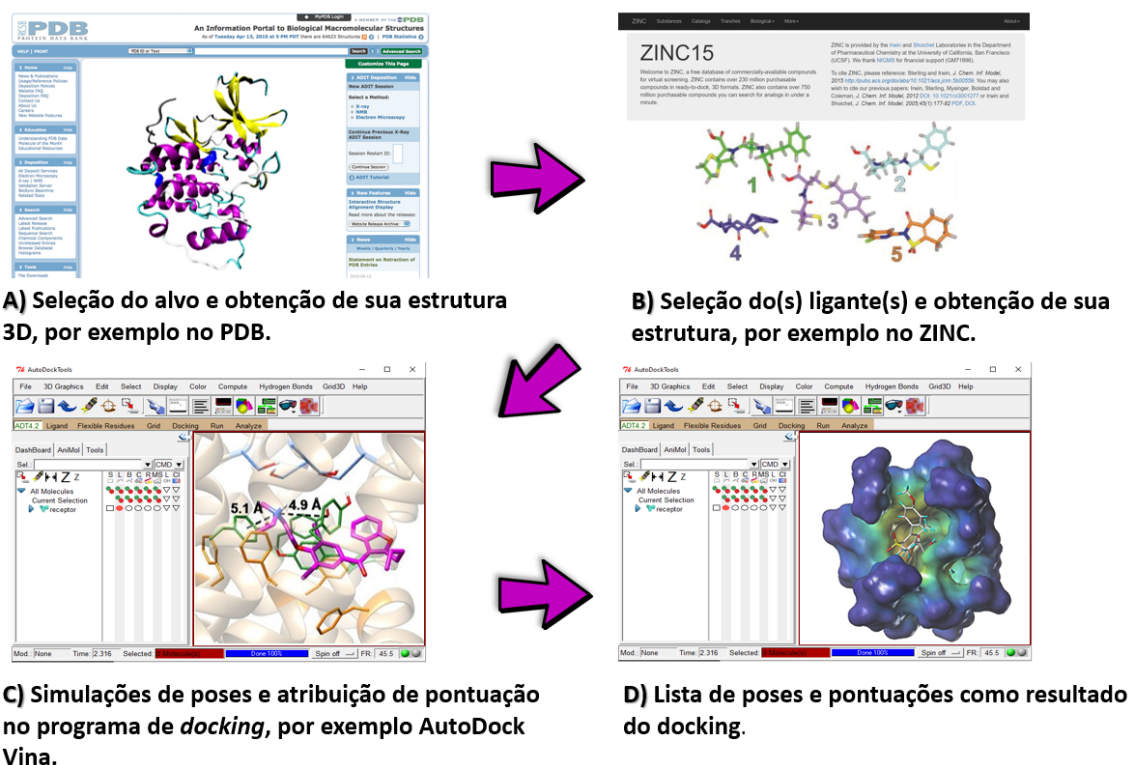


Figura 1: Etapas realizadas na metodologia de *docking* molecular

desafios. Um deles está em selecionar as características realmente relevantes para que ocorra uma previsão da função das proteínas com exatidão, reduzindo assim a possibilidade de *overfitting*, uma vez que existem vários parâmetros envolvidos em cada uma dessas funções, que podem ser modificados para melhorar as previsões (Quiroga and Villarreal, 2016). Além disso, ainda é um grande desafio fazer com que uma função de pontuação consiga reconhecer com melhor exatidão determinadas conformações que tenham grandes possibilidades de serem a pose real de interação entre o ligante e a macromolécula e atribuir um alto valor de escore para estas poses.

Diante desses desafios, novas ferramentas computacionais vêm, ao longo dos anos, sendo incorporadas para prever a intensidade e a especificidade com que pequenas moléculas, normalmente denominadas de ligantes (drogas ou fármacos, em potencial) se ligam ao sítio ativo de um receptor biológico (região da biomolécula onde ocorre as interações químicas entre o ligante e o alvo biológico) (Lengauer and Rarey, 1996). Uma das ferramentas que pode ser utilizada com tal finalidade é o AutoDock Vina, um *software* que utiliza uma função de pontuação (*score*) baseada em campos de força (*force field*) para predição da afinidade de ligação ligante-receptor através de uma estimativa calculada pelo somatório das forças intermoleculares de van der Waals e interações eletrostáticas entre todos os átomos do complexo usando cálculos de mecânica clássica (Trott and Olson, 2010).

No entanto, embora o programa AutoDock Vina, segundo (Wang et al., 2016), te-

nha sido eficiente para avaliar os possíveis ligantes de uma proteína de interesse, uma correlação relativamente fraca foi observada entre as funções de pontuação que tentavam prever a afinidade de ligação (Su et al., 2018). Para otimizar o poder de pontuação do AutoDock Vina e obter uma melhor correlação dos dados experimentais, foi proposto por (Wang and Zhang, 2017) uma nova função de pontuação, denominada Δ VinaRF. Essa função, combina a força de afinidade calculada pelo AutoDock Vina e utiliza um termo de correção calculado pelo método de *Random Forest* (Wang and Zhang, 2017).

A utilização dessa função de pontuação, vêm demonstrando um grande sucesso para criar modelos que auxiliam na seleção de conformações promissoras de um receptor e um dado ligante, onde futuramente, podem ser utilizados em experimentos de docagem molecular, esse recurso foi denominado DeltaVinaRF20 (Li et al., 2015). Com esta solução, é possível obter um conjunto de recursos significativos e as técnicas utilizando algoritmos avançados de aprendizado de máquina permitiram alcançar um desempenho superior em comparação às funções clássicas de pontuação (Wang and Zhang, 2017).

Dada a importância do DeltaVinaRF20, este trabalho tem o objetivo de propor uma metodologia para extração de atributos relevantes de proteínas, utilizando os 20 atributos selecionados por (Wang and Zhang, 2017), sendo 10 de Autodock Vina e os outros 10 relacionados a área superficial acessível ao solvente (bSASA). Estes atributos foram combinados com o conjunto do RDKit 2D, composto por 147 características físico-químicas e mais 58 características do AutoDock Vina. Totalizando 205 atributos.

Neste trabalho, analisamos estes atributos para a predição de Δp_{k_d} e p_{k_d} . Para isso, classificamos ambos os atributos de acordo com sua ordem de importância e combinamos os resultados em uma pontuação acumulada, utilizando métodos de ML para a identificação dos fatores mais relevantes e que exibem a maior influência no resultado da classificação.

1.1 Objetivo Geral

Este trabalho tem como objetivo aprimorar a energia livre de ligação entre complexos proteína-ligante, utilizando Δp_{k_d} e p_{k_d} para sua predição.

1.2 Objetivos Específicos

Para que o objetivo geral seja alcançado, propõem-se os seguintes passos a serem executados no decorrer deste trabalho:

- Realizar a seleção de atributos com Aprendizado de Máquina.
- Realizar a predição de energia livre de ligação, utilizando Δp_{k_d} e p_{k_d} a partir de diferentes métodos de Aprendizado de Máquina, combinando estes resultados em um *Ensemble*.

1.3 Estrutura do Trabalho

Este trabalho está dividido da seguinte maneira: no Capítulo 2 é feita uma contextualização para o entendimento do restante do trabalho no que se refere ao Aprendizado de Máquina. No Capítulo 3 são apresentados alguns dos conceitos biológicos que são necessários para compressão de alguns dos assuntos tratados ao longo do texto. No Capítulo 4 é explicado as métricas que são utilizadas para avaliação de funções de escore. No Capítulo 5 é mencionado como os métodos de ML podem ser aplicados para melhorar a predição da afinidade de ligação; Neste mesmo capítulo, detalhamos cada um dos métodos que são utilizados neste estudo e as técnicas que foram aplicadas. No Capítulo 6 são apresentados os trabalhos relacionados. O Capítulo 7 detalha a metodologia proposta. O Capítulo 8 aborda os dados e as simulações realizadas para desenvolvimento da técnica apresentada neste estudo. No Capítulo 9 os resultados dos experimentos são exibidos. Por fim, o Capítulo 10 traz as discussões e o Capítulo 11 as considerações finais deste trabalho.

2 INTRODUÇÃO AO APRENDIZADO DE MÁQUINA

Uma área de conhecimento central no presente trabalho é a de Aprendizado de Máquina. Nesta subseção foi realizada uma contextualização para o entendimento do restante do trabalho no que se refere ao Aprendizado de Máquina; Nessa mesma subseção são apresentadas algumas das técnicas de combinação de classificadores que são utilizadas para *feature selection* e *docking scoring prediction* no contexto da Bioinformática.

2.1 Conceitos Iniciais

O Aprendizado de Máquina (AM) é uma área da Inteligência Artificial (IA), ao qual engloba um conjunto de metodologias para o desenvolvimento de sistemas computacionais capazes de criar por si próprios, a partir de experiências passadas, uma hipótese, ou função, capaz de resolver problemas que se deseja tratar computacionalmente (Carvalho et al., 2011).

Dessa forma, a partir da manipulação de dados de treinamento ou experiências passadas, algoritmos de AM são amplamente utilizados tanto para aquisição automática de conhecimento, quanto para a modelagem de aplicações que envolvem um grande volume de dados, e por este motivo esta área de estudo tem grande interesse da comunidade de pesquisa científica como um todo (Dias et al., 2007).

Em geral, algoritmos de AM podem lidar com tarefas bastante distintas, ao qual podem ser organizadas de acordo com diferentes critérios (Carvalho et al., 2011). Contudo, um dos princípios fundamentais, diz respeito ao paradigma de aprendizagem que deve ser adotado para lidar com a tarefa preditiva (Faceli et al., 2011). Seguindo este critério, de acordo com Monard and Baranauskas (2003), as tarefas de aprendizado podem ser divididas em preditivas e descritivas e diferem quanto aos tipos de problemas que cada algoritmo é capaz de resolver.

Em tarefas de previsão, são empregados algoritmos ou métodos de AM que induzem modelos preditivos, seguindo o paradigma de aprendizado supervisionado (Mitchell et al., 2013). No aprendizado supervisionado o modelo é treinado sobre um conjunto de dados pré-definido e, baseado no treinamento com os dados pré-definidos, é possível tomar

decisões precisas quando novos dados são recebidos (Lior et al., 2014). Basicamente, neste tipo de tarefa são utilizados modelos de classificação e regressão (Han et al., 2006).

Classificação e regressão são dois tipos de aplicações de aprendizado indutivo. O conceito de aprendizado indutivo, está relacionado ao fato do algoritmo de AM aprender a partir de um conjunto de dados de treinamento, ao qual tenta procurar uma hipótese, no espaço de possíveis hipóteses, capaz de descrever as relações entre os objetos e que melhor se ajuste aos dados de treinamento (Carvalho et al., 2011).

Nas tarefas de classificação e regressão, se o domínio em relação aos dados de treinamento for um conjunto finito e não ordenado de valores, o problema é dito de classificação. Porém, se o domínio em relação aos dados de treinamento for um conjunto infinito e ordenado de valores, o problema é dito regressão. Podemos assim afirmar que, regressão é a predição no espaço dos números reais e que classificação é a predição em um conjunto discreto de valores; respectivamente o resultado da regressão pertence ao conjunto dos reais $\mathbb{R}$ e o resultado da classificação pertence a um conjunto $\{c_1, c_2, \dots, c_m\}$ de m classes.

Em tarefas de descrição, o objetivo é explorar ou descrever um conjunto de dados. Os algoritmos de AM utilizados nessas tarefas, podem automaticamente encontrar padrões e relações em um conjunto de dados (Carvalho et al., 2011). Por isso, seguem o paradigma de aprendizado não supervisionado.

No aprendizado não supervisionado, geralmente, costuma-se lidar com grandes quantidades de dados não rotulados, ou seja, não se sabe quantas ou quais classes existem (Harrington, 2012). O objetivo é encontrar estruturas intrínsecas aos dados e, nesse caso, o processo é guiado somente pelas relações entre os dados e nenhum conhecimento externo sobre os rótulos é fornecido. Para isso, utilizam-se técnicas de associação, agrupamento e sumarização (Han et al., 2006).

A partir destas técnicas, é possível desvendar a organização dos padrões existentes nos dados, descobrir similaridades e diferenças, assim como derivar conclusões úteis a respeito deles (Lior et al., 2014). Este tipo de abordagem é bastante utilizada em aplicações de mineração de dados (*datamining*), onde o conteúdo de grandes bases de dados não é conhecido antecipadamente (Harrington, 2012).

2.2 Introdução à técnica de seleção de atributos

Uma das técnicas que auxilia no estudo de um grande volume de dados é a seleção de atributos, ao qual tem representado um papel muito importante na fase de pré-processamento dos dados, seja focado em ferramentas de análise ou de aprendizado de máquina. Trata-se de uma das etapas fundamentais que precede a realização de análises e predições. Ela pode, inclusive, impactar no desempenho dos modelos de previsão, que são gerados a partir dos dados.

Uma vez que o número total de possíveis atributos que podem ser tratados por algoritmos de aprendizado de máquina é relativamente alto. Caso todos sejam utilizados, resultaria no problema de alta dimensionalidade, o que dificulta o processamento, ou até mesmo, tornando-o impraticável. Sem mencionar nas variáveis irrelevantes ou redundantes que podem prejudicar as previsões dos modelos. Para evitar que isso aconteça, a redução do número de atributos pode melhorar o desempenho dos modelos preditivos, reduzir o custo computacional, simplificando o modelo e tornando mais fácil sua compreensão.

Existem várias técnicas de seleção de atributos que podem ser utilizadas para isso. Essas técnicas procuram reduzir a dimensionalidade dos dados, ao obter um subconjunto de atributos a partir do conjunto de atributos original, de acordo com determinado critério de busca, ao qual seleciona dentre estes, os mais relevantes. O objetivo é remover recursos irrelevantes e/ou redundantes e reter apenas os recursos relevantes sem afetar o desempenho do aprendizado (Sammut and Webb, 2011).

Neste contexto, diferentes estratégias para buscar os melhores atributos podem ser utilizadas. Dependendo do tipo de procedimento de busca que é utilizado, os métodos de seleção de atributos podem ser divididos em *forward increase*, *backward deletion*, *random*, e *hybrid models* (Cai et al., 2018). Segundo Cai et al. (2018), esses métodos podem ser categorizados de acordo com vários padrões. De acordo com os dados de treinamento utilizados (rotulados, não rotulados ou parcialmente rotulados), os métodos de seleção de recursos podem ser divididos em modelos supervisionados, não supervisionados e semi-supervisionados. Além disso, podem ser divididos em três principais categorias, de acordo com a relação entre os métodos de aprendizado, sendo estes classificados em modelos de filtro, wrapper e *Embedded Models* e dependendo do tipo de saída, podem ser divididos em *feature rank (weighting)* e *subset selection models*.

No presente trabalho, utilizamos os *Embedded Models*. Nesse tipo de abordagem a etapa de seleção de atributos deixa de ser uma fase de pré processamento e é realizada dinamicamente durante a fase de treinamento. Desse modo, os próprios algoritmos de aprendizado que foram utilizados durante este estudo, realizam a seleção dos atributos internamente. Os atributos são avaliados observando quais deles mais favorecem o acerto do modelo. A partir da medida de importância de cada atributo no treinamento é realizada a seleção. A ideia básica é tentar combinar diferentes métodos de seleção de atributos que gerem *rankings* a partir da medida de importância de cada recurso, encontrar os melhores métodos e os melhores atributos que mais contribuem para predição da afinidade de ligação, quando utilizados para estimar os fatores de correção da função DeltaVinaRF20.

3 CONCEITOS BIOLÓGICOS

O objetivo deste capítulo é o de apresentar um entendimento biológico mínimo necessário para que, nos capítulos seguintes, seja possível compreender os aspectos computacionais envolvidos.

3.1 O Dogma Central da Biologia Molecular

O Dogma Central da Biologia Molecular postulado por Francis Crick em 1958, tem como objetivo explicar como ocorre o fluxo da informação genética (Watson et al., 2015). Esse dogma demonstra todas as etapas de acordo com o qual o fluxo da informação genética ocorre do DNA através do RNA para a proteína (Nelson and Cox, 2018).

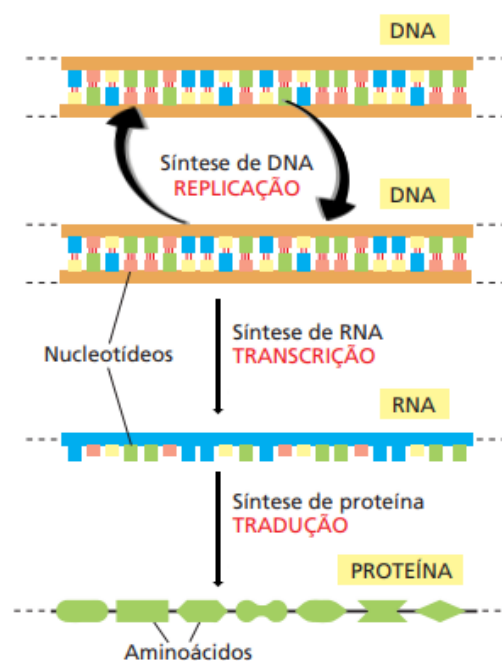


Figura 2: Representação do fluxo de informação em sistemas biológicos. Fonte: Adaptado de (Alberts et al., 2010)

Esse processo, como é apresentado na Figura 2 compreende a fase de transcrição, onde o DNA é processado, gerando o mRNA, ou RNA mensageiro, que é enviado para

fora do núcleo da célula, onde a mensagem é traduzida em proteínas (Nelson and Cox, 2018). Na transcrição, os segmentos de uma sequência de DNA são usados para guiar a síntese de moléculas de RNA (Verli, 2014). Depois, na tradução, as moléculas de RNA são usadas para guiar a síntese de moléculas de proteínas (Watson et al., 2015).

3.2 Proteínas

As proteínas, são polímeros longos de aminoácidos e constituem a maior fração de macromoléculas (além da água) que compõe uma célula (Nelson and Cox, 2018). Algumas proteínas têm atividade catalítica e funcionam como enzimas; outras servem como elementos estruturais, receptores de sinal, ou transportadores que carregam substâncias específicas para dentro ou para fora das células (Alberts et al., 2010). Cada uma destas funções, são desempenhadas pela ligação entre os aminoácidos que formam uma proteína (Verli, 2014).

Como os aminoácidos se relacionam entre si de maneira variada, as proteínas tem conformações estruturais bastante diversificadas (Watson et al., 2015). Assim, cada tipo de proteína possui uma configuração molecular específica que é determinada pela sequência de aminoácidos e pelo grau de inclinação entre as ligações químicas, proporcionada pelos arranjos intermoleculares (Voet et al., 2014). A estrutura molecular das proteínas é muito complexa, por essa razão a estrutura de uma proteína é representada, basicamente, de 4 maneiras diferentes conforme é apresentado na Figura 3 (Nelson and Cox, 2018).

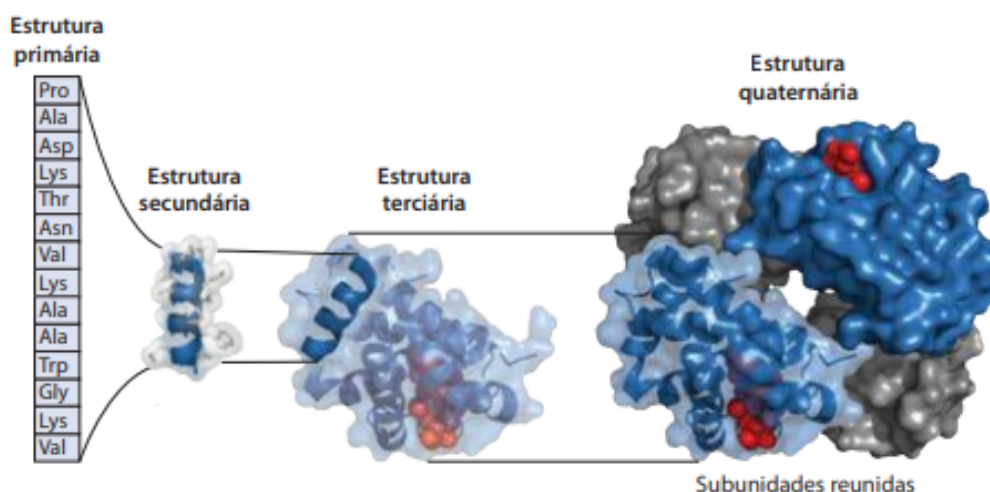


Figura 3: Níveis de organização da estrutura molecular de uma proteína. Fonte: Adaptado de (Nelson and Cox, 2018).

- Estrutura Primária: Refere-se ao número linear e a ordem dos aminoácidos presentes;
- Estrutura Secundária: Compreende arranjos particularmente estáveis de resíduos de aminoácidos dando origem a padrões estruturais recorrentes;

- Estrutura Terciária: Consiste em como os segmentos das estruturas secundárias se associam dentro de uma única cadeia polipeptídica para formar toda a estrutura tridimensional da proteína;
- Estrutura Quaternária: Refere-se ao número e à combinação de duas ou mais cadeias, ou subunidades, de proteínas, formando um complexo de multi-subunidades.

A configuração espacial final das proteínas (estrutura terciária ou quaternária) é determinante para as funções biológicas por elas exercidas. No entanto, em muitos casos, partes das proteínas carecem de estruturas perceptíveis. Esses segmentos proteicos são intrinsecamente desordenados. Em alguns casos, proteínas inteiras são intrinsecamente desordenadas, e ainda assim funcionais (Nelson and Cox, 2018).

Todos estes fatores dificultam a compreensão dos processos envolvidos nas interações intermoleculares das proteínas com outras moléculas. Sabe-se, no entanto, que a conformação nativa está associada a um baixo nível de energia livre do sistema. A energia livre da proteína serve para determinação da sua estrutura 3D, mas não é relacionada com a energia livre de ligação. É importante salientar que nos processos para realização de *docking* essa última que é interessante.

3.3 Aminoácidos: Estrutura e Propriedades

Em geral, a função desempenhada por uma proteína depende de sua estrutura tridimensional que é constituída a partir de um repertório de moléculas, conhecidas como aminoácidos. Para gerar uma determinada proteína, os aminoácidos ligam-se entre si a partir de interações iônicas, interações hidrofóbicas, ligações de hidrogênio e ponte dissulfeto, em uma sequência linear característica.

Conforme é apresentado na Figura 4, os aminoácidos são compostos originados da ligação entre um grupo funcional amina (H_2N), um grupo carboxílico ($COOH$), um átomo de hidrogênio e uma cadeia lateral (R). Essas estruturas, são primordiais para desempenhar diversas funções nos organismos vivos, atuam no controle do pH intracelular, servem como reserva de nitrogênio e regulam diferentes aspectos do metabolismo, seja agindo em suas formas finais ou provendo moléculas do metabolismo intermediário que atuam como efetores na regulação da expressão gênica.

Em relação as suas estruturas, as proteínas podem ser primárias (principal), secundárias, terciárias ou quaternárias, que resultam de ligações entre moléculas ou em partes da molécula. Na estrutura primária a ligação entre os aminoácidos é covalente e ocorre entre o carbono (C) de um aminoácido e o nitrogênio (N) do aminoácido vizinho, classificada como ligação peptídica. Para a estrutura secundária, os aminoácidos estão ligados entre si covalentemente na estrutura primária, mas as moléculas podem sofrer rotações a partir do carbono alfa e a cadeia pode interagir com ela mesma de três formas: alfa-

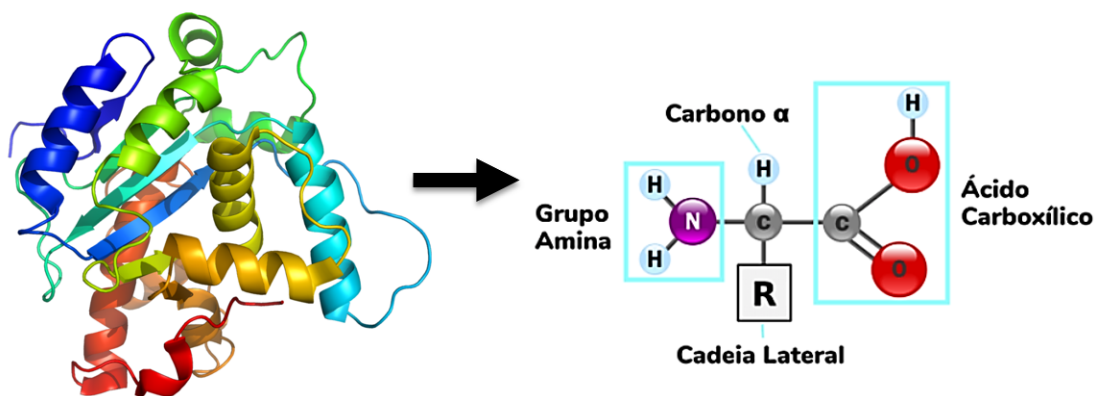


Figura 4: Estrutura de um aminoácido e as interações para gerar uma proteína. Fonte: Adaptado de (Nelson and Cox, 2018)

hélice (formam-se ligações de hidrogênio entre os aminoácidos); folhas-beta (as ligações de hidrogênio entre um aminoácido e outro gera uma estrutura folhear e rígida) e laços (formam-se fora do dobramento da proteína, não é uma estrutura regular no núcleo). Já em sua forma terciária, o dobramento da estrutura secundária faz com que a proteína assuma uma estrutura tridimensional no espaço. Em sua última instância, na estrutura quaternária, as proteínas compõem uma junção de várias estruturas terciárias.

Essas estruturas são responsáveis por todas as tarefas essenciais à manutenção da vida. Contudo, apesar dos diversos papéis desempenhados por esta classe de moléculas, os organismos vivos constroem as proteínas a partir de uma combinação de apenas 20 tipos de aminoácidos encontrados na natureza, dentre os quais, estão apresentados na Tabela 1.

3.4 Interações entre Proteínas e Ligantes

O conhecimento do modo de interação entre pequenas moléculas ligantes e proteínas alvo é de grande importância para o planejamento racional de fármacos (Verli, 2014). Uma vez que a maioria dos medicamentos atua em proteínas, reconhecendo e se ligando a uma proteína específica, alterando sua atividade, enquanto deixa as demais proteínas idealmente com suas funções inalteradas. Contudo, esse tipo de interação somente é possível se a superfície de contorno da molécula do ligante se ajusta muito precisamente à proteína (Alberts et al., 2010).

Para isso, o ligante seleciona uma determinada conformação entre as preexistentes (com a qual possui maior afinidade). Geralmente, se ligando em uma conformação energeticamente favorável à proteína (Verli, 2014). Neste processo, há uma combinação entre forças intermoleculares atrativas e repulsivas, isto é, interações hidrofóbicas, eletrostáticas e estéricas, que permitem que essa interação ocorra. O somatório dessas forças, é responsável por gerar uma resposta fisiológica específica.

Esses possíveis papéis fisiológicos são determinados pela afinidade de ligação, ao

Tabela 1: Nomenclatura dos 20 aminoácidos essenciais. Fonte: Adaptado de (Verli, 2014)

Aminoácido	Abreviação	Código
Alanina	A	ALA
Arginina	R	ARG
Asparagina	N	ASN
Aspartato	D	ASP
Cisteína	C	CYS
Glutamina	Q	GLN
Glutamato	E	GLU
Glicina	G	GLY
Histidina	H	HIS
Isoleucina	I	ILE
Leucina	L	LEU
Lisina	K	LYS
Metionina	M	MET
Fenilalanina	F	PHE
Prolina	P	PRO
Serina	S	SER
Treonina	T	THR
Triptofano	W	TRP
Tirosina	Y	TYR
Valina	V	VAL

qual traduz a capacidade da micromolécula em se complexar com o sítio complementar de interação (Verli, 2014). A afinidade de ligação é normalmente medida e reportada pela constante de dissociação de equilíbrio (Verli, 2014).

Em geral, como é apresentado na equação (1), a ligação de um ligante a um receptor geralmente pode ser vista como uma reação reversível simples, em que R e L são as concentrações, respectivamente, do receptor e do ligante livres e RL é a concentração do complexo receptor-ligante:



Onde k_{ligado} e $k_{\text{desligado}}$ são as constantes de velocidade para as reações de formação e dissociação do complexo RL (para a frente associação e reversa dissociação). Ou seja, $k_{\text{desligado}}$ é a taxa constante de dissociação de um ligante ao seu receptor, e k_{ligado} é a taxa constante de formação de um complexo receptor-ligante a partir do ligante livre e do receptor.

No equilíbrio, as velocidades de formação e dissociação são iguais e a lei de ação das massas pode ser aplicada para a reação, conforme é demonstrado na equação (2).

$$k_{\text{ligado}}[R][L] = k_{\text{desligado}} \quad (2)$$

Em que R e L são as concentrações, respectivamente, do receptor livre (i.e., receptor sem ligante ligado) e do ligante, no ponto de equilíbrio, e $[RL]$ é a concentração do complexo receptor-ligante. Assim, no equilíbrio, é possível descrever K_d (constante de dissociação de equilíbrio), como mostra a equação (3).

$$\frac{[RL]}{[R][L]} = \frac{k_{\text{desligado}}}{k_{\text{ligado}}} = K_d \quad (3)$$

Com base nessa equação, é possível observar que quanto menor for $k_{\text{desligado}}$ em relação a k_{ligado} mais estável será o complexo RL e, conseqüentemente, menor será o valor de K_d . Logo, se K_d é baixa, a afinidade de ligação é alta.

A Figura 5 ilustra as variações das constantes de dissociação típicas de várias classes de interações nos sistemas biológicos. A afinidade de ligação é alta quando K_d possui valores de concentração molar muito baixos, geralmente 10^{-7} ou menos. Isso significa que o receptor detecta concentrações na ordem de micromolar ou nano molar.

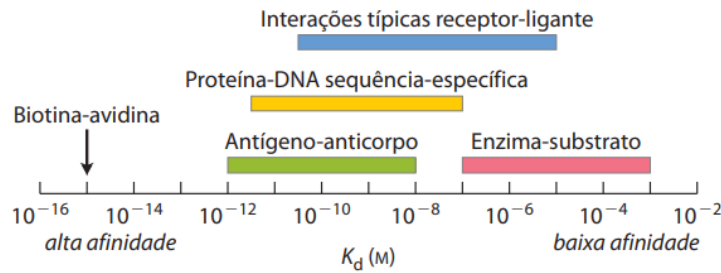


Figura 5: Escala de valores para K_d . Fonte: Adaptado de (Nelson and Cox, 2018).

Os valores K_d para os casos de interação fraca tem valores acima de 10 mM, já em interações muito fortes se alcança valores de 10^{-16} M.

Como os valores dessa escala são expressos em potências de 10, a medida de afinidade de um receptor pelo seu ligante, conforme Rocha and de Azevedo Júnior (2009), pode ser expressa como o negativo do logaritmo da constante de dissociação no equilíbrio (K_d), ou seja, $p_{K_d} = -\log_{10} K_d$, como apresentado nas equações (4), (5) e (6).

$$\log_{10} K_d = -p_{K_d} \leftrightarrow 10^{-p_{K_d}} = K_d \quad (4)$$

$$-p_{K_d} = \log_{10} K_d \leftrightarrow 10^{-p_{K_d}} = K_d \quad (5)$$

$$p_{K_d} = -\log_{10} K_d \quad (6)$$

A partir dos valores de p_{K_d} podemos selecionar características estruturais e geométricas de complexos proteína-ligante que permitem uma melhor predição da afinidade de ligação. Porém, quando a concentração do ligante for muito maior do que a concentração dos sítios

de interação com o ligante, a interação com a proteína não se altera de modo significativo e a concentração do ligante livre (não ligado) – isto é, $[L]$ permanece constante na equação (1).

Essa condição é muito aplicável à maioria dos ligantes que interagem com proteínas nas células e simplifica a descrição do equilíbrio de ligação. Com isso, também é possível considerar o equilíbrio da ligação do ponto de vista da fração, Θ (teta), dos sítios de interação com o ligante na proteína que estão ocupados pelo ligante, como apresentado na equação (7).

$$\Theta = \frac{[\textit{sitios ocupados}]}{[\textit{total de sitios disponiveis}]} = \frac{[RL]}{[RL] + [R]} \quad (7)$$

Como essas interações em sua maior parte são reversíveis e podem ser descritas através de um expressão de equilíbrio simples, como foi demonstrado em (1). No sentido de associação, a medida de afinidade do ligante $[L]$ pela proteína é representada por K_A , como é apresentado através da equação (8).

$$K_A = \frac{[RL]}{[R][L]} = \frac{k_{\textit{ligado}}}{k_{\textit{desligado}}} \quad (8)$$

O termo de equilíbrio K_A é equivalente também à razão entre as velocidades das reações que formam o complexo RL . A velocidade de associação é descrita pela constante de velocidade $k_{\textit{ligado}}$, e a dissociação pela constante de velocidade $k_{\textit{desligado}}$. Observa-se que K_A é a recíproca de K_d . Logo, um valor maior de K_A corresponde a uma afinidade mais alta do ligante pela proteína.

Mas, se redefinirmos a primeira parte da equação (8), obteremos na equação (9), ou seja, a razão entre o receptor ligado e livre que é diretamente proporcional à concentração ligante livre.

$$K_A = \frac{[RL]}{[R]} \quad (9)$$

A partir dessa equação, é possível simplificar Θ . Assim, partindo da equação (9) e rearranjando os termos, chegamos a equação (10).

$$K_A = [R][L] = [RL] \therefore [RL] = K_A[R][L] \quad (10)$$

Dessa forma, podemos reescrever Θ com base na equação (10).

$$\Theta = \frac{K_A[R][L]}{K_A[R][L] + R} \quad (11)$$

Dividindo ambos os lados da equação (11) por $[R]$ e K_A obtemos a equação (12) para Θ .

$$\Theta = \frac{[L]}{[L] + \frac{1}{K_A}} \quad (12)$$

Como K_d é o contrário de K_A de acordo com a (13), é possível simplificar Θ , resultando na equação (14)

$$K_d = \frac{1}{K_A} \quad (13)$$

$$\Theta = \frac{[L]}{[L] + K_d} \quad (14)$$

Graficamente, como apresentado na Figura 6 o valor de K_A pode ser determinado a partir de uma curva de Θ versus a concentração do ligante livre, $[L]$.

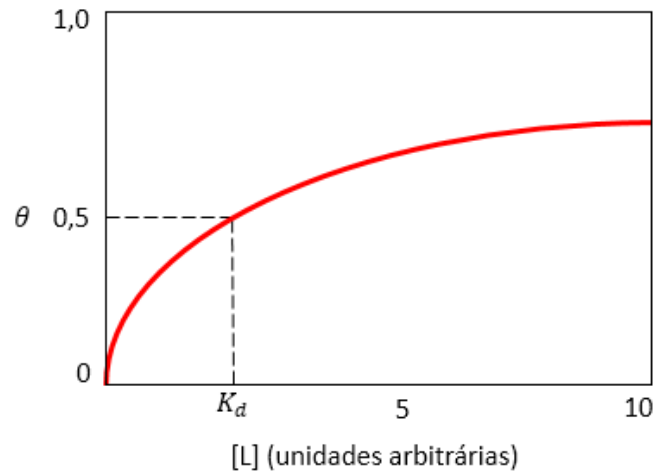


Figura 6: Representação gráfica da interação com o ligante. Fonte: Adaptado de (Nelson and Cox, 2018)

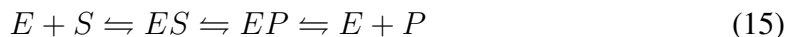
No gráfico, Θ representa a fração dos sítios de ligação ocupados pelo ligante, ou seja, a concentração do ligante na qual a metade dos sítios está ocupada. Sendo assim, o ponto onde a curva de ligação cruza a linha onde $\Theta = 0,5$ é a constante de dissociação (K_d).

A fração dos sítios de interação ocupados pelo ligante se aproxima assintoticamente da saturação à medida que $[L]$ aumenta. A $[L]$ na qual a metade dos sítios disponíveis está ocupada (i.e, $\Theta = 0,5$) corresponde a $\frac{[1]}{K_A}$.

No entanto, é mais comum (e intuitivamente mais simples) considerar a constante de dissociação, K_d , que é a recíproca de K_A ($K_d = \frac{[1]}{K_A}$), sendo dada em unidades de concentração molar (M). Como sabemos, K_d é a constante de equilíbrio para a liberação do ligante. Assim; quando $[L] = K_d \rightarrow [L]$ metade dos sítios de ligação estarão ocupados pelo ligante.

Contudo, há outras proteínas nas quais a ligação com o ligante é somente a primeira etapa necessária nas suas funções. Esse é o caso de uma grande e importante classe de

proteínas chamadas de enzimas. Essas macromoléculas se ligam a um ou mais ligantes, chamados de substratos e os convertem em um ou mais produtos quimicamente modificados. Sabendo estes pressupostos, uma reação enzimática, pode ser equacionada a partir da equação (15).



Nessa equação, E significa Enzima, S é Substrato e P representa Produto. Na catálise de uma reação química, as enzimas interagem com os substratos, formando com eles, temporariamente, o chamado complexo enzima-substrato. Na formação das estruturas secundária e terciária de uma enzima, acabam surgindo certos locais na molécula que servirão de encaixe para o alojamento de um ou mais substratos.

Esses locais de encaixe são chamados de sítio ativos e ficam na superfície da enzima. Ao se encaixarem nos sítios ativos, os substratos ficam próximos um do outro e podem reagir mais facilmente. Assim que ocorre a reação química com os substratos, desfaz-se o complexo enzima-substrato. Liberam-se os produtos e a enzima volta a atrair novos substratos para a formação de outros complexos. No caso mais simples, o caminho básico dessa reação pode ser expresso pela equação (16).



Nessa equação, considera-se a reação reversa, na qual $E + P$ recombina-se para formar EP e então ES , isso ocorre tão raramente que podemos ignorá-la. Assim, podemos expressar a taxa da reação conhecida como sua velocidade. A velocidade com que o complexo ES é formado a partir da enzima e substrato livre é chamada de K_1 e a velocidade de dissociação do complexo ES em substrato-enzima-livre é chamada K_{-1} . A velocidade de formação de enzima e produto livre, partindo do complexo ES é chamado de K_2 e a formação de ES a partir da enzima-produto livre é chamado de K_{-2} , como mostra a equação (17)



K_{-2} depende da velocidade de formação do complexo EP , mas como essa etapa pode ser removida, podemos também remover K_{-2} ficando assim, com a equação (18).



O fator que afeta a velocidade da cinética da enzima é $[S]$ (concentração de substrato). Porém, estudar os efeitos da $[S]$ é complicado pois a mesma se altera no curso de uma reação. Na inibição enzimática, o que ocorre é um acúmulo de substrato sem permitir a formação de produto na catálise enzimática.

Assim, se $[S] \ll [E]$ e neste caso, podemos mensurar a velocidade inicial da reação v_0 . Quando $[S]$ é baixa, temos a enzima livre. Mas, na medida em que aumentamos $[S]$, a enzima vai se complexando com o substrato e então, é possível mensurar a velocidade inicial da reação. Para um melhor entendimento de como isso acontece, é preciso considerar novamente a equação (18).

Nessa situação, $[S]$ vai aumentando, como $[S] \ll [E]$ ocorrerá um deslocamento do equilíbrio para direita para formação do $[ES]$. Porém, quando $[ES]$ se forma ele se dissocia para formar a enzima e o produto. Logo, a concentração do produto vai aumentando.

Se $[S]$ se mantém constante, o complexo $[ES]$ vai diminuir para formação do produto. Mas, se $[S]$ estiver bem alta (geralmente é bem alta) a velocidade de dissociação e associação começa a ficar constante e $[ES]$ começa a não mais variar com o tempo. A partir disso, é possível medir a velocidade de uma enzima com base nos postulados de Michaelis e Menten.

Michaelis e Menten, em 1913, propuseram um modelo de reação enzimática para apenas um substrato. A partir deste modelo, foi desenvolvida a equação (19) que permite demonstrar como a velocidade de uma reação varia com a variação da concentração do substrato. Esta equação pode ser expressa graficamente, como mostra a Figura 7 e representa o efeito da concentração de substrato sobre a velocidade de reação enzimática. Michaelis e Menten deduziram uma constante (K_m) que é indicadora da velocidade da reação e conseqüentemente do grau de afinidade que a enzima tem pelo substrato.

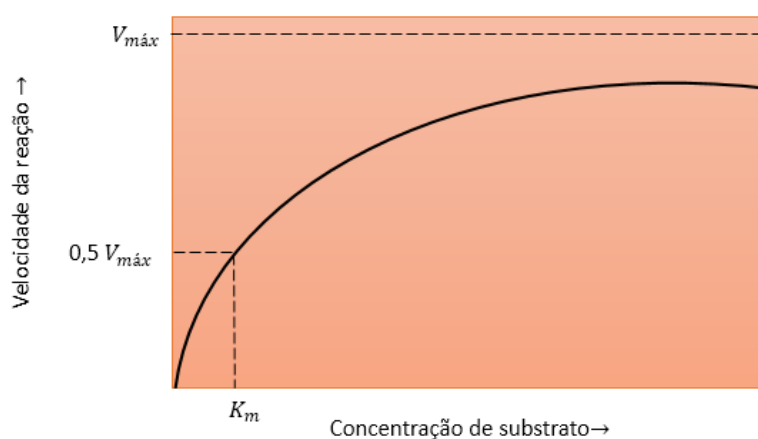


Figura 7: Cinética Enzimática. Fonte: Adaptado de (Alberts et al., 2010).

A constante de Michaelis-Menten (K_m) é definida como a concentração de substrato necessária para que metade da velocidade máxima da reação seja atingida. O K_m de um substrato para uma enzima específica é característico, e fornece um parâmetro de especificidade deste substrato em relação à enzima. Quanto menor o K_m , maior a especificidade, e quanto maior o K_m , menor a especificidade.

$$V_0 = \frac{V_{max}[S]}{K_m + [S]} \quad (19)$$

Além destas interações, as enzimas podem sofrer também diferentes tipos de inibição, classificadas quanto a sua ligação e que interferem com a catálise, diminuindo ou interrompendo as reações enzimática. Esses inibidores causam um acúmulo de substrato sem permitir a formação de produto na catálise enzimática. Na inibição competitiva, um inibidor tenta se ligar no mesmo local onde o substrato se liga. Assim, forma-se dois complexos, um complexo $[ES]$ e um $[EI]$ como apresentado na equação (20).



O esquema acima representa um mecanismo em que E (a enzima livre), I e EI estão em equilíbrio químico durante todo o processo catalítico; existe assim uma constante de equilíbrio K_i (à frente designada de constante de inibição) que relaciona as concentrações de equilíbrio destas três substâncias e que pode ser definida como a constante de dissociação do complexo EI, de acordo com a equação (21).

$$K_i = \frac{[I][E]}{[EI]} \quad (21)$$

O desenvolvimento matemático deste modelo conduz à equação (22) que determina a velocidade na presença do inibidor. Onde: I é a concentração do substrato, K_i é a constante de dissociação de EI e S é a concentração do substrato.

$$V_I = \frac{V_{max}S}{K_m \left(1 + \frac{I}{K_i}\right) + S} \quad (22)$$

Esta equação mostra que o valor da constante de Michaelis para um determinado valor de [I]. O valor da constante de Michaelis aparente (que pode ser observada na presença de uma dada concentração de I) é expresso na equação (22) por $1 + \frac{I}{K_i}$. De acordo com Yung-Chi and Prusoff (1973), quando o inibidor competitivo produz metade da inibição, temos uma concentração denominada IC_{50} . Assim, quando: $I = IC_{50}$, $V_0 = 2V_I$ obtém-se a equação 23.

$$\frac{2V_{max}S}{K_m \left(1 + \frac{IC_{50}}{K_i}\right) + S} = \frac{V_{max}S}{K_m + S} \quad (23)$$

Rearranjando os termos da equação (23) e isolando IC_{50} obtemos a equação (24).

$$IC_{50} = K_i \left(1 + \frac{S}{K_m}\right) \quad (24)$$

Logo, se isolarmos K_i na equação (24) ele poderá ser determinado a partir da (25).

$$K_i = \frac{IC_{50}}{1 + \frac{S}{K_m}} \quad (25)$$

Com essa equação, a constante K_i pode ser redefinida como a concentração do inibidor que dobra o valor de K_m ; quanto menor for o seu valor maior será a capacidade inibidora da substância I . Essa medida é muito importante uma vez que muitos fármacos exercem seu efeito biológico como consequência da inibição enzimática (Yung-Chi and Prusoff, 1973).

4 AVALIAÇÃO E PREDIÇÃO DE FUNÇÕES DE ESCORE

A previsão precisa e eficiente da energia livre de ligação de grandes conjuntos de complexos proteína-ligante é um desafio-chave na ciência biomolecular computacional, com aplicações na descoberta de novas drogas, na biologia química e na biologia estrutural (Ashtawy and Mahapatra, 2012).

Diversas abordagens com diferentes níveis de complexidade foram desenvolvidas para prever a energia livre de ligação. Contudo, os métodos mais precisos para previsão dessa medida são computacionalmente muito custosos. Uma das alternativas apontadas por Schulz-Gasch and Stahl (2004) para aumentar a acurácia de tais predições, é a utilização das funções de pontuação.

Uma função de pontuação é um modelo preditivo matemático o qual produz uma pontuação que representa a afinidade de ligação de um receptor pelo seu ligante. O resultado do processo de *docking*, é um conjunto de ligantes classificados de acordo com os escores de ligação previstos (Ashtawy and Mahapatra, 2012).

A afinidade de ligação de um ligante a uma proteína pode ser prevista em diferentes maneiras e, portanto, as funções de pontuação podem ser classificadas em uma das três categorias: campo de força, empírica e baseada em conhecimento, conforme Liu and Wang (2015).

As funções de pontuação baseadas no campo de força, buscam quantificar as forças moleculares reais que existem entre uma proteína e uma molécula pequena (Cheng et al., 2009). As interações de van der Waals, as interações eletrostáticas e as interações de ligações de hidrogênio são componentes comuns das funções de pontuação baseadas em campo de força (Koes et al., 2013). Esse tipo de função é frequentemente projetada para uso em simulações de dinâmica molecular (Ewing et al., 2001).

Funções empíricas foram criadas para reproduzir a afinidade de ligação experimental com alta precisão (Liu and Wang, 2015). Essas são baseadas na correlação entre a energia livre de ligação e um somatório ponderado de variáveis independentes (Quiroga and Villarreal, 2016). De acordo com Khamis and Gomaa (2015), os coeficientes são obtidos através da análise de afinidades de ligação conhecidas. Ao qual, supõe-se que toda a energia seja composta de termos de energia ponderada. Métodos de regressão linear são

usados para aprender os coeficientes do modelo. Isso pode ser feito ajustando as energias de ligação experimentais conhecidas a um conjunto de dados de treinamento.

Funções baseadas em conhecimento são fundamentadas na análise estatística da interação entre os pares de átomos do complexo receptor-ligante e incorporam elementos de funções de pontuação baseadas no campo de força e baseadas no conhecimento (Huang and Zou, 2010). Elas consistem em termos fisicamente significativos que são parametrizados para reproduzir afinidades de ligação ou poses de ligação, porém o grande número de termos aumenta o risco de *overfitting*, necessitando de um protocolo de validação rigoroso (Koes et al., 2013).

É importante ressaltar que não existe uma função de avaliação universal, que pode ser aplicada com acurácia em qualquer caso, uma vez que existe uma variedade de funções de pontuação que foram desenvolvidas até o momento. Contudo, alguns parâmetros objetivos são desejados para avaliar seus pontos fracos e pontos fortes. Conforme o guia comparativo de funções de pontuação CASF-2016 elas devem ser avaliadas a partir de quatro métricas (Su et al., 2018): poder de pontuação, poder de classificação, poder de ancoragem e poder de triagem.

4.1 Poder de pontuação

O poder de pontuação avalia a capacidade da função de escore em produzir pontuações de ligação em uma correlação linear com os dados de ligação experimental. A técnica empregada para a medida da correlação linear é o coeficiente de correlação de Pearson (equação (26)). Esse coeficiente deve indicar o grau de intensidade da correlação linear entre o escore de ligação e as afinidades de ligação experimentais (Li et al., 2014).

$$R = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}} \quad (26)$$

Onde, x_i é a pontuação de ligação calculada por uma certa função de pontuação no complexo; y_i é a afinidade de ligação experimental; i representa cada um dos valores x e y observados; $\bar{x}$ e $\bar{y}$ descrevem a média aritmética de todos os valores observados das variáveis x e y , respectivamente; as constantes de ligação são todas dadas em unidades logarítmicas (Li et al., 2014).

Para verificar a variabilidade dos valores preditos *versus* experimentais, pode-se utilizar o desvio padrão, como apresentado na equação (27).

$$SD = \sqrt{\frac{\sum (y_i - (a + bx_i))^2}{n - 1}} \quad (27)$$

Nessa equação, $\sum (y_i - (a + bx_i))^2$ refere-se a utilização do método dos mínimos quadrados para determinar os parâmetros da relação entre o valor experimental y_i , e o valor predito x_i pela regressão como é ilustrado na Figura 8.

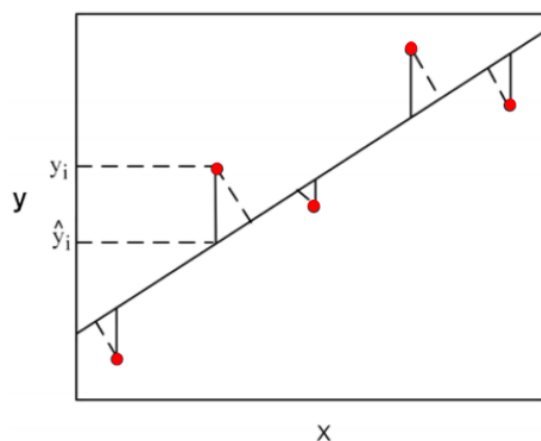


Figura 8: Reta ajustada por mínimos quadrados. Fonte: Adaptado de (Passari et al., 2011).

Nesse método a melhor localização da reta especificada pelos coeficientes a e b da equação é aquela que minimiza a soma dos quadrados dos comprimentos dos segmentos verticais, indicado pelas linhas sólidas, que ligam os pontos experimentais à reta. Este critério implica que os valores de x_i sejam fixos, definidos pelo pesquisador no planejamento estatístico, e os valores de y_i as variáveis aleatórias afetadas por erros experimentais (Passari et al., 2011).

Essa equação depois é dividida por $(n - 1)$, ou seja, pelo número de graus de liberdade da amostra, ao qual n representa a quantidade de complexos proteína-ligante utilizados nos conjuntos de treinamento e de teste. Os graus de liberdade indicam os espaços entre esses dados; e são iguais a $(n - 1)$ porque os espaços entre eles estão sempre uma unidade abaixo do número dos próprios dados (Passari et al., 2011).

É importante ressaltar que o poder de pontuação é avaliado através do coeficiente de correlação de Pearson, que visa mensurar a capacidade de produzir pontuações para as diferentes poses de vinculação; essas pontuações devem estar linearmente correlacionadas com a ligação de afinidade medida experimentalmente dos complexos proteína-ligante de estruturas 3D conhecidas (Khamis and Gomaa, 2015). O desvio padrão é somente um indicador adicional neste caso.

4.2 Poder de classificação

O poder de classificação, refere-se à capacidade de uma função de pontuação para classificar corretamente os ligantes conhecidos de uma determinada proteína alvo por suas afinidades de ligação quando as posições precisas de ligação desses ligantes são fornecidas (Gaillard, 2018).

Para calcular essa métrica, é utilizado o coeficiente de correlação de Spearman (RS), como é apresentado na equação (28).

$$RS = \frac{\sum x_i y_i - \frac{\sum x_i \sum y_i}{n}}{\sqrt{\sum x_i^2 - \frac{\sum x_i^2}{n}} \sqrt{\sum y_i^2 - \frac{\sum y_i^2}{n}}} \quad (28)$$

Essa medida avalia a precisão de classificação das funções de pontuação de complexos proteína-ligante divididos em grupos específicos com diferentes ligantes ligados à mesma proteína alvo (Khamis and Gomaa, 2015). Pode-se interpretar os coeficientes de correlação de Spearman (RS) como uma medida alternativa para a precisão da classificação, que visa mensurar qual função de pontuação ranqueia melhor os ligantes com base nas afinidades de ligação previstas e determinadas experimentalmente a partir de um grupo específico de ligantes.

4.3 Poder de ancoragem

O poder de ancoragem, refere-se à capacidade da função para identificar a posição de ligação do ligante nativo entre posições gerados por computador. Cada função de pontuação é aplicada para classificar o conjunto de dados de cada complexo proteína-ligante e deve encontrar consequentemente, a pose de ligação melhor pontuada. O conjunto de dados de cada complexo inclui a pose de ligação nativa para garantir que exista pelo menos uma pose de ligação correta.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n [(\alpha_i - \alpha'_i)^2 + (\beta_i - \beta'_i)^2 + (\gamma_i - \gamma'_i)^2]}{n}} \quad (29)$$

Para estimar esta medida, é utilizada a raiz quadrática média ($RMSE$) que é calculada a partir da equação (29). Se o valor da raiz quadrática média ($RMSE$) entre a ligação nativa e a posição de ligação melhor pontuada entre todos os dados mais o nativo caiu abaixo do limite predefinido, por exemplo, se $RMSE < 2,0 \text{ \AA}$ é registrado como uma previsão bem-sucedida. Na equação (29), os coeficientes: α_i , β_i , γ_i e α'_i , β'_i , γ'_i são as coordenadas cartesianas do i -ésimo átomo em duas poses de ligação e n é a quantidade de átomos de cada complexo proteína-ligante utilizado nos conjuntos de treinamento e de teste.

De acordo com Khamis and Gomaa (2015), para avaliar uma função de pontuação por meio dessa medida, um conjunto de poses de ligação (*decoys*) são geradas para cada complexo proteína-ligante. Cada função de pontuação é aplicada para classificar o conjunto de *decoys* de cada complexo proteína-ligante e deve encontrar consequentemente, a pose de ligação melhor pontuada. O conjunto de dados de cada complexo inclui a pose de ligação nativa para garantir que exista pelo menos uma pose de ligação correta.

4.4 Poder de triagem

O poder de triagem, refere-se à capacidade de uma função de pontuação para identificar os verdadeiros ligantes de uma proteína alvo determinada entre um grupo de moléculas aleatórias. O poder de triagem de uma função de pontuação é medido através do fator de enriquecimento (EF) por meio da contagem do número total de ligantes verdadeiros entre os ligantes de 1%, 5% e 10% da melhor classificação, utilizando a equação (30).

$$EF_{x\%} = \frac{NTB_{\%}}{NTB_{totalx\%}} \quad (30)$$

Onde $NTB_{totalx\%}$ é o número de ligantes verdadeiros observados entre os candidatos top $x\%$ (onde $x = 1\%$, 5% ou 10%) selecionados por uma determinada função de pontuação. NTB_{total} é o número total de ligantes verdadeiros para a proteína alvo dada (que é tipicamente 3 para cada proteína alvo se não houver ligantes cruzados). Como outro indicador de desempenho para o poder de triagem em Li et al. (2014); se o melhor ligante foi encontrado entre os candidatos de 1%, 5% ou 10%, a previsão da função de escore foi bem sucedida.

Assim, para cada proteína é ancorado um conjunto de ligantes. Entre todas as poses disponíveis é tomada a posição de ligação prevista e a correspondente pontuação de ligação é tomada pela afinidade de ligação prevista por uma determinada função de pontuação. O valor $EF_{x\%}$ então, avalia a capacidade da função de pontuação em conseguir identificar a molécula nativa através da afinidade de ligação calculada.

5 ML APLICADO EM *DOCKING SCORING PREDICTION*

As técnicas de atracamento molecular são utilizadas para identificar compostos com afinidade anteriormente desconhecida para um alvo a partir de sua estrutura tridimensional, usando funções de pontuação intrínsecas dos programas de *docking*. Normalmente, se analisa a saída destes programas e se identifica manualmente as *features* que poderiam ser utilizadas para seleção de uma pose (“conformação de baixa energia”) ou para distinguir entre ligantes ativos e *decoys*. O problema inerente a essa abordagem rígida é que ela leva a uma baixa previsibilidade para os complexos que não estão em conformidade com as premissas de modelagem (Ain et al., 2015).

Sistemas baseados em aprendizado de máquina vêm sendo utilizados com sucesso para melhorar o resultado de *docking*, tanto no que diz respeito na melhora da performance das funções de pontuação quanto na construção de classificadores de afinidade de ligação (Hecht and Fogel, 2009). Um dos grandes diferenciais das estratégias que empregam aprendizado de máquina é a capacidade de lidar com a dependência não linear entre diversas interações envolvidas entre o ligante e o receptor. Ao não impor nenhuma forma funcional específica para a função de pontuação, qualquer tipo possível de interação pode ser inferido diretamente a partir de dados experimentais (Ballester and Mitchell, 2010). Isso possibilita um maior desempenho para prever as afinidades de ligação de diversos complexos proteína-ligante.

Em sua pesquisa, Ashtawy and Mahapatra (2013) demonstraram que as funções de pontuação as quais empregam diferentes abordagens de aprendizado de máquina superam amplamente as funções de pontuação convencionais. Nesse cenário, os métodos de AM podem ter a sua aplicação bem sucedida, pois há várias vias de pesquisa que provavelmente levarão a novas melhorias de desempenho. No momento, as principais estratégias usadas em *docking scoring prediction* são redes neurais (Neural Networks - NN), máquinas de vetores de suporte (Support Vector Machines - SVM) e floresta aleatória (Random Forest - RF) (Li et al., 2019).

O método SVM tem sido frequentemente empregado na triagem virtual baseada em estrutura para discriminar poses ativas do ligante de poses inativas e também utilizado para predição de afinidade de ligação (Zhang et al., 2017). Esse método consiste em uma

combinação linear de padrões de suporte nas quais são o subconjunto de padrões de treinamento que estão mais próximos do limite de decisão, sendo aplicável a uma variedade de funções de classificação. Estudos envolvendo a utilização desse tipo de abordagem foram apontadas em Bock and Gough (2002), onde os pesquisadores utilizaram o algoritmo SVM no mapeamento funcional entre um conjunto de características do receptor-ligante e a energia total de ligação livre do complexo.

Outra técnica de aprendizado de máquina utilizada para predição das funções de pontuação é o RF. Um algoritmo com base em uma combinação de preditores de árvore em que cada árvore depende dos valores de um vetor aleatório amostrado independentemente e com a mesma distribuição para toda a “floresta”.

Ao utilizar eficientemente um grande conjunto de dados experimentais, composto por vários atributos, as funções aleatórias de pontuação baseada em RF podem obter melhores correlações com os dados experimentais de ligação proteína-ligante com estruturas cristalinas conhecidas (Li et al., 2019).

Foi demonstrado por Li et al. (2019), que o RF-Score, um tipo de função de pontuação baseada em RF é, em média, muito mais precisa do que as funções de pontuação clássicas na predição da afinidade de ligação dos complexos proteína-ligante.

No entanto, Gabel et al. (2014) consideraram que os descritores não descrevem verdadeiras interações proteína-ligante, e o RF-Score considera apenas a relação entre a contagem par a par de proteína-ligante e as afinidades de ligação, o RF-Score não funciona bem para a previsão de pose de ligação e triagem virtual. Recentemente, (Wang and Zhang, 2017) empregaram descritores diferentes para descrever as interações proteína-ligante e aumentar a precisão preditiva (Li et al., 2019). Outro método de aprendizado de máquina que tem demonstrado várias vantagens para a predição da afinidade de ligação são as NN. Um modelo computacional para simulação da função cerebral, ao qual tem sido amplamente utilizado em pesquisas recentes sobre descoberta de novos fármacos (Li et al., 2019).

Um exemplo da utilização desta técnica é a função clássica de pontuação de redes neurais (NNScore) proposta por Durrant et al. (2013). Essa função é utilizada para rastrear ligantes candidatos por uma camada de saída de classificação binária. O NNScore versão 2.0 considera mais características de ligação e a saída pode prever a afinidade de ligação. Além disso, Durrant et al. (2013) relatam que, usando a NNScore, os resultados podem melhorar o desempenho da pontuação. Embora os SFs baseados em NN tenham uma precisão preditiva consideravelmente alta para a afinidade de ligação, eles tem um bom desempenho para dados de alta dimensão, o que limita bastante seu uso em aplicativos de acoplamento comercial (Li et al., 2019).

De acordo com uma avaliação sobre o uso dos métodos de AM, analisando o desempenho de diferentes modelos de predição, Pason and Sottriffer (2016) indicaram em seu estudo as principais funções de pontuação desenvolvidas nos últimos 20 anos, dentre as

quais são descritas na Tabela 2.

Tabela 2: Funções de pontuação desenvolvidas com métodos ML (*Machine Learning*) nos últimos 20 anos. Fonte: Adaptado de (Pason and Sotriffer, 2016).

Scoring Function	Ano de pub.	Mét. de Reg.	Total de artigos / Citações
SCORE1	1994	MLR	691/15
ChemScore	1997	MLR	944/59
SCORE2	1998	MLR	310/11
X-CScore	2002	MLR	506/53
SFCscore	2008	MLR, PLS	38/5
PESD-SVM	2010	ML: SVM	21/4
NNScore	2010	ML: NN	32/9
RF-Score	2010	ML: RF	71/23
SFCscore	2013	ML: RF	15/9

Foi demonstrado que essas funções de aprendizado de máquina superam uma ampla variedade de funções de pontuação clássicas tanto na previsão de afinidade de ligação quanto na triagem virtual (Pason and Sotriffer, 2016). Embora tenham sido recentemente introduzidas, as funções de pontuação baseadas em modelos de regressão, tem sido capazes de explorar efetivamente quantidades muito maiores de dados experimentais quando comparadas as demais (Li et al., 2019).

No entanto, apesar das contribuições com o desenvolvimento dessas funções de pontuação e dos avanços significativos dos métodos de AM, prever a afinidade de ligação das interações proteína-ligante permanece um dos problemas mais críticos e desafiadores (Ain et al., 2015). Para tentar solucionar as deficiências das abordagens existentes, pode ser aplicada uma etapa de pré-processamento, ao qual foi utilizada neste trabalho. Essa etapa é utilizada antes da classificação e, é denominada seleção de atributos. Essa técnica é combinada em um sistema *Ensemble* e tem por objetivo selecionar os recursos que mais contribuem para a variável de previsão ou saída a fim de fornecer predições mais precisas, aperfeiçoando sistematicamente esses tipos de modelos preditivos a uma variedade de algoritmos de aprendizado de máquina (Ain et al., 2015).

5.1 Técnicas de seleção de atributos aplicadas em docking scoring para melhorar a predição da afinidade de ligação

Atualmente, são disponibilizados uma ampla variedade de dados de biologia molecular por meio de banco de dados públicos, como Genbank, EMBL, DDBJ, UniProt, NCBI-Protein. Nos últimos anos, a quantidade de dados biológicos coletados e armazenados nesses bancos de dados e disponíveis para análise tem aumentado exponencialmente (Li

et al., 2019). Para lidar com essa enorme quantidade de dados, pesquisas nessa área têm se dedicado ao estudo e a proposta de métodos de processos para tratar esses dados, como é o caso da utilização de técnicas de seleção de atributos.

Por meio dessa técnica, como foi abordado na tese de Santos (2017), objetiva-se encontrar um conjunto de características que melhor represente uma função proteica em meio as inúmeras informações acerca das estruturas das proteínas disponibilizadas nestas diferentes bases de dados. Nessa pesquisa, os autores descrevem que a utilização de técnicas de seleção de atributos, podem diminuir o tempo computacional necessário para treinar os algoritmos de aprendizado de máquina, com uma melhora na precisão preditiva e um aumento na compreensibilidade do problema.

No geral, esse método tem demonstrado resultados promissores para a seleção dos parâmetros físico-químicos em funções de pontuação eficazes. No trabalho de (Wang et al., 2015), a técnica de seleção de atributos utilizando o método dos mínimos quadrados e de análise de componentes principais foi implementada.

A ideia central da aplicação do método de análise de componentes principais, no estudo de (Wang et al., 2015), foi reduzir a dimensionalidade de um conjunto de dados de várias medidas inter-relacionadas. Essa redução foi alcançada transformando os parâmetros em um novo conjunto de variáveis (os principais componentes), para que os primeiros conseguissem reter a maior parte da variação (presumida como representando informações) presente no conjunto de dados original.

De fato, pode-se observar no trabalho de (Wang et al., 2015) que a representação dada por esse método alcança resultados satisfatórios para a redução da dimensão linear. Além disso, facilita a interpretação dos resultados, porque extrai recursos diretamente relacionados ao conjunto de dados original. Nesse mesmo estudo, o método dos mínimos quadrados foi aplicado para diferentes conjuntos de dados da família de proteínas, o que indica que diferentes recursos contribuem para diferentes modelos. Três modelos específicos de famílias foram construídos para três importantes famílias alvo de proteínas. As comparações com os outros métodos demonstraram que a representação individual e a construção do modelo para cada família de proteínas é uma maneira mais razoável de prever a afinidade de uma família de proteínas específica (Wang et al., 2015).

Um estudo apresentado por Ballester and Mitchell (2010) demonstrou que a utilização do algoritmo de aprendizado de máquina *Random Forest* para seleção das características físico-químicas melhorou a predição dos resultados a partir de um conjunto de instâncias reduzido e, portanto, a disponibilidade futura de mais dados estruturais e de interação de alta qualidade deverá levar a versões aprimoradas da função de pontuação desenvolvida pelos pesquisadores. Além disso, o *Random Forest* também pode ser usado para estimar a importância variável como uma maneira de identificar os contatos proteína-ligante que mais contribuem para a previsão de afinidade de ligação entre complexos conhecidos (Ballester and Mitchell, 2010).

Távora et al. (2017) descreve a utilização do método Wrapper. Esse método avalia todas as combinações possíveis dos recursos e seleciona a combinação que produz o melhor resultado para o algoritmo de aprendizado de máquina específicos. Esta investigação introduz um novo método, denominado Wrapper de Pontuação para o Cálculo da Importância de Recursos (SWtoFIC). O SWtoFIC fornece o subconjunto dos recursos mais importantes e sua importância relativa, não apenas relacionados à variável de destino, mas também entre si. Os resultados fornecidos indicam que o modelo proposto é uma ferramenta muito promissora para esta aplicação específica.

Em Fogel et al. (2010), os autores utilizaram algoritmos de aprendizado de máquina para seleção de recursos e geração de modelos de funções de pontuação de acoplamento personalizadas para o estudo dos inibidores da tuberculose conhecidos da proteína transportadora enoil-acil redutase. Os autores apresentam funções personalizadas de pontuação de encaixe específicas do alvo, geradas com regressão linear múltipla, mínimos quadrados parciais, máquinas de vetor de suporte e redes neurais artificiais. Os modelos resultantes podem ser usados para identificar os principais descritores da inibição da proteína transportadora enoil-acil-redutase e são úteis para a triagem de alto rendimento de novos compostos de drogas.

Abordagens utilizando algoritmos genéticos foram utilizadas por Pathak et al. (2016), para seleção de atributos físicos químicos em conjuntos específicos de proteína-ligante. Devido à sua eficácia na solução de problemas de otimização, esse algoritmo pode ser promissor para os problemas de acoplamento proteína-ligante de acordo com Pathak et al. (2016).

Neste trabalho, foi adotada um tipo de abordagem para seleção de características, que utiliza alguns dos métodos mencionados anteriormente no decorrer desta seção. Foram adotadas cinco técnicas para seleção de componentes: *Random Forest*, *Decision Tree*, *Cross Validation Lasso*, *Principal Component Analysis* e *Automatic Relevance Determination Regression*. O uso desses métodos objetiva identificar os atributos que tem maior poder explicativo para predição da afinidade da ligação e excluir os que possuem pouca influência, que possivelmente introduzem ruído e podem causar *overfitting*.

5.2 Tipos de modelos *Ensemble Learning*

Nesta seção será apresentado um conceito conhecido por *Ensemble*. Uma vez que, este trabalho engloba uma etapa de seleção de características físico químicas que caracterizam complexos proteína-ligante, ao qual teve como objetivo tentar selecionar em um *ranking* acumulativo apenas aquelas características geradas que contribuem para a melhora da saída do *Ensemble*, através da predição da afinidade de ligação entre estes complexos. Com isso, as previsões das saídas geradas por esses elementos selecionados (características) foram combinados em uma única saída, que corresponde ao resultado

final do *Ensemble*. Dessa forma, torna-se necessário conhecer como é realizada esta combinação, as diferentes maneiras e principalmente os métodos que podem ser utilizados.

O termo *Ensemble* consiste em criar um modelo de aprendizagem de máquina que irá combinar as previsões de outros modelos. Espera-se que o resultado desse modelo combinado possua uma melhor capacidade de previsão, ou seja, que possa por exemplo identificar mais classes corretas em um teste do que qualquer modelo individual que foi utilizado (Paris et al., 2017). A Figura 9 ilustra uma arquitetura básica abordada pelo *Ensemble*.

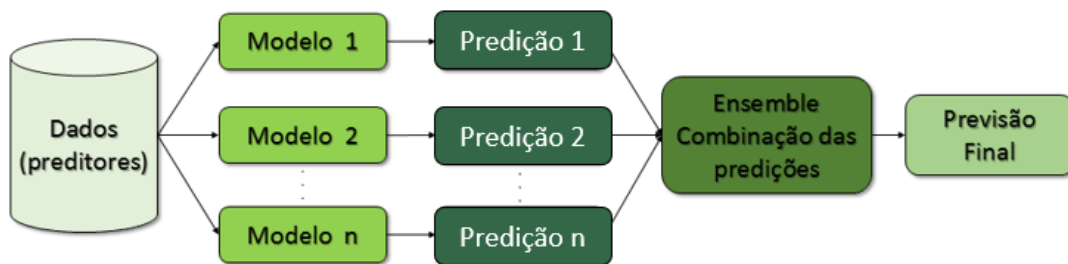


Figura 9: Arquitetura tradicional de *Ensemble Learning*. Fonte: Adaptado de (Kamber and Pei, 2006).

Com relação a abordagens de ensembles de classificadores, podemos destacar o *Bagging* proposto por Breiman (1996) e *Boosting* desenvolvido por Freund et al. (1996). *Bagging* treina um conjunto de classificadores-base com diferentes amostragens da base de treinamento. A amostragem é realizada com reposição e tem o mesmo tamanho da base de treinamento original. Após obter os resultados dos classificadores-base, *Bagging* combina-os por votação majoritária, e a classe mais votada é a predita. Para regressores, uma combinação algorítmica é aplicada. A Figura 10 ilustra o modelo conceitual de funcionamento do *Bagging*.

É importante destacar que o algoritmo *Bagging* também pode ser usado para um regressor. Como é o caso do método *Random Forest*, aplicado neste trabalho, que se baseia na aplicação de *bagging* e também consiste em uma combinação (ensemble) de árvores de decisão.

Ao contrário do *Bagging*, a técnica de *Boosting*, usa inteiramente a base de treinamento para criar classificadores em série, sendo que a cada iteração dá-se mais ênfase às instâncias que foram classificadas incorretamente na iteração anterior.

Para isso, os pesos dos exemplos classificados incorretamente são aumentados e dos exemplos classificados corretamente diminuídos. Finalmente, quando uma nova instância é apresentada, cada classificador-base dá seu voto, ponderado por sua acurácia global, e o rótulo do exemplo é selecionado com base na maioria. O esquema de funcionamento do *Boosting* é representado pela Figura 11.

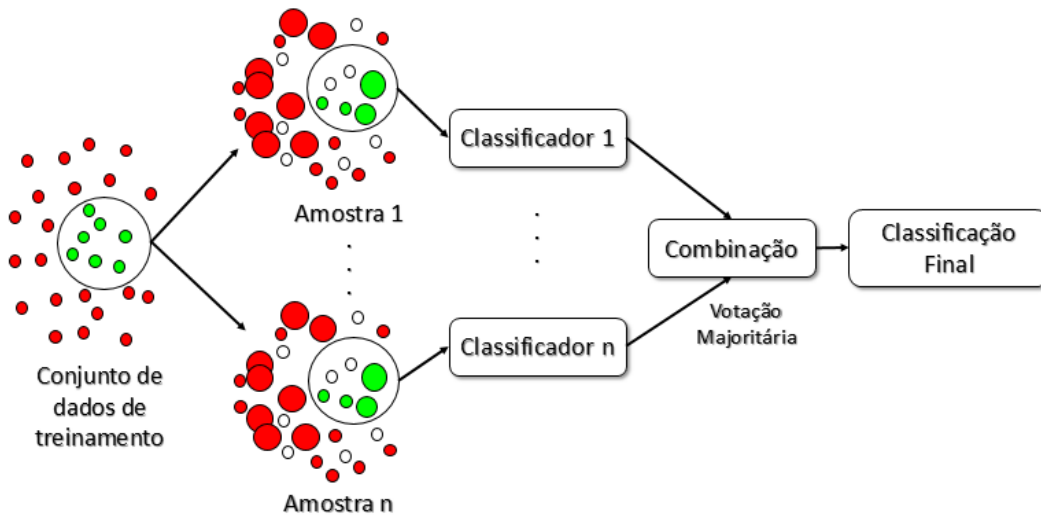


Figura 10: Modelo conceitual: Bagging. Fonte: Adaptado de (Shafiei and Jazayeri-Rad, 2012).

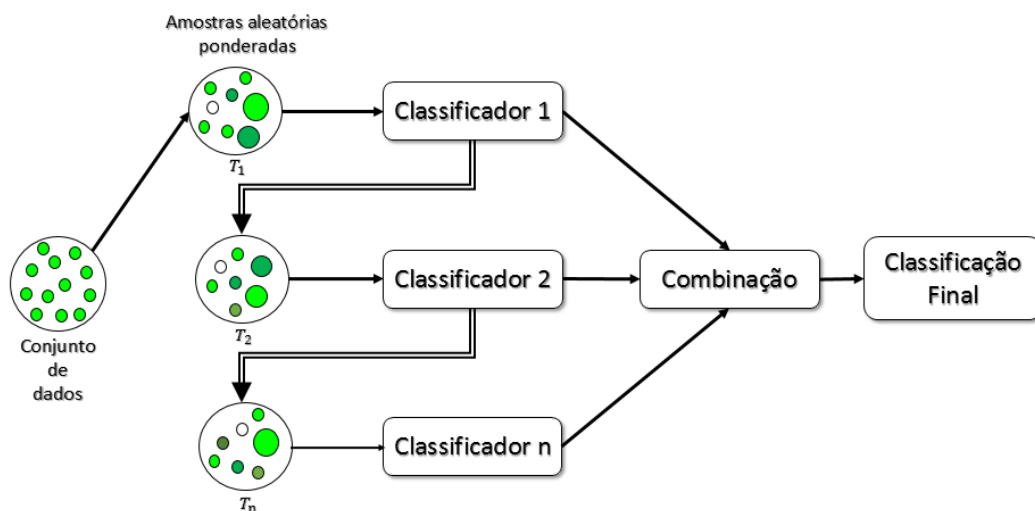


Figura 11: Modelo conceitual: *Boosting*. Fonte: (Merrill et al., 2019).

Pode-se observar que a principal diferença entre os conceitos de Bagging e Boosting está ligada ao seu funcionamento, no qual no processo de Bagging o treinamento é dado pela combinação de modelos independentes o que pode generalizar o treinamento. Já no processo de Boosting, o treinamento é dado por uma combinação dos modelos, normalmente gerada por uma aplicação de um fator entre cada iteração, minimizando assim as chances de generalização (super ajuste). No geral, esses métodos tem se mostrado bastante eficazes, em diferentes cenários de aplicação, levando uma acurácia preditiva estável e, muitas vezes, superior àquela obtida por um classificador isoladamente.

Neste trabalho aplicamos um sistema *Ensemble* que funciona da seguinte forma: primeiramente, um conjunto de atributos foi avaliado de acordo com sua medida de importância, utilizando os métodos *Decision Tree*, *Random Forest*, *Lasso*, *ARD*, e *PCA*. Combinamos os métodos de seleção de atributos que geram uma lista ordenada de atri-

butos em um *ranking* acumulativo. Em mais detalhes, depois de escolher as “ k ” *features* com a maior importância, foi realizada a predição de Δp_{k_d} e p_{k_d} , para calcular a energia livre de ligação entre os complexos proteína-ligante utilizados, mas para diferentes tamanhos k . Neste conjunto de dados, seriam $k = 205$ (todas as *features*), depois $k = 5$, $k = 10$, $k = 15$, $k = 20$, $k = 25$, $k = 30$, $k = 40$, $k = 50$, $k = 70$, $k = 80$, $k = 90$, $k = 100$, $k = 105$ e $k = 200$.

Para realizar a predição, utilizamos os fatores de correção Δp_{k_d} e p_{k_d} da função $\Delta V_{\text{VinaRF20}}$ ao aplicar cada um dos conjuntos de atributos selecionados. Nesta etapa, empregamos os algoritmos *Decision Tree*, *Random Forest*, *Lasso*, *Gradient Boosting* e *Support Vector Machine*. Realizamos 10 repetições para este experimento. Com estes resultados, geramos a média, a mediana e o *trim rule*. Um escore foi gerado ao calcular o poder de pontuação e o poder de classificação para cada interação. A partir deste sistema, verificamos quais são os conjuntos de atributos que obtiveram uma melhor pontuação em prever a afinidade de ligação entre os complexos proteína-ligante utilizados e comparamos nossos resultados com a função de pontuação utilizada no programa AutoDock Vina.

5.3 Métodos de ML utilizados para seleção de atributos e *feature importance*

Na área da Bioinformática, modelos preditivos podem ser utilizados para estimar a energia de ligação entre complexos proteína-ligante, dado um conjunto de características físico químicas responsáveis pelas interações biológicas que ocorrem entre essas moléculas. Seus resultados, podem trazer implicações positivas na redução de custos no processo de desenvolvimento de novos fármacos.

Esta seção trata sobre os algoritmos de seleção de subconjuntos de recursos, bem como à aplicação de algoritmos de aprendizado de máquina para análises preditivas, dentre os quais foram utilizados neste estudo. Para analisar a importância dos recursos, foram utilizados os métodos: *Decision Tree*, *Random Forest*, *Lasso*, *ARD* e *PCA*.

Para a predição dos atributos alvo foram utilizados ambos os métodos, com exceção do *ARD* e *PCA*. Isso porque embora o *ARD* tenha um bom desempenho preditivo para a predição das variáveis, ele tem um custo computacional substancial devido à exploração necessária do espaço do modelo (Paananen et al., 2019). Por esse motivo, ele foi descartado para este experimento.

Com relação ao *PCA*, esse método realiza uma redução na dimensionalidade dos dados, ou seja, uma transformação. Nosso objetivo inicialmente, não é transformar os dados, porque já temos nosso conjunto de atributos (sem transformação) e que devem estar presentes em nosso modelo para realizar a predição dos atributos alvos.

Para mais detalhes, a seguir estão presentes alguns conceitos sobre como esses métodos funcionam e como são utilizados, tanto para a seleção de atributos quanto para a predição.

minem no mesmo conjunto. A medida com base na qual a condição ótima (localmente) é escolhida é chamada de impureza.

Para classificação, normalmente é a impureza de Gini ou o ganho de entropia de informações e, para as árvores de regressão, é a variação. Assim, ao treinar uma árvore, pode-se calcular quanto cada recurso diminui a impureza ponderada em uma árvore. Para uma floresta, a diminuição da impureza de cada recurso pode ser calculada como média e os recursos são classificados de acordo com esta medida.

5.3.2 *Decision Tree*

O método de aprendizado de máquina *Decision Tree*, apresentado na Figura 13 cria modelos e gera previsões utilizando uma adaptação do algoritmo *Random Forest*, ao qual foi explicado na seção anterior.

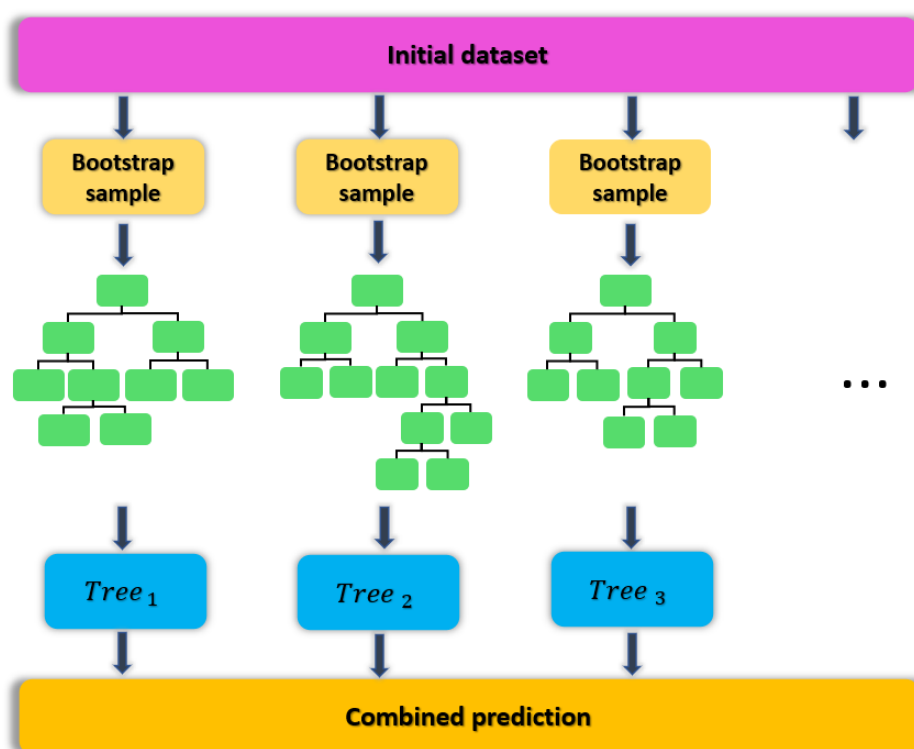


Figura 13: Funcionamento do algoritmo *Random Forest*. Fonte: Adaptado de (Polishchuk, 2015)

Nesse modelo, o conjunto de dados é dividido aleatoriamente e a árvore cresce a partir das amostras *bootstrap* tiradas do conjunto de treinamento. Para cada elemento do conjunto de teste, compara-se a classe prevista com a classe verdadeira. A proporção das vezes que a classe estimada difere da verdadeira é a taxa de erro de classificação *bagging*. A partir disso, a divisão aleatória dos dados nos conjuntos teste e treinamento é repetida. O resultado final reportado consiste nas taxas médias de erro sobre as árvores, bem como os respectivos erros padrão.

Os principais passos desse algoritmo, são resumidos de acordo com Breiman (1996) da seguinte forma: Dado um conjunto de dados de treinamento $X = x_1, x_2, \dots, x_j$ e $Y = y_1, y_2, \dots, y_j$. Para $b = 1, 2, 3, \dots, B$, repita: Cria uma amostra *bootstrap* (X_b, Y_b) com n exemplos de (X, Y) . Ajusta uma árvore de decisão f^b para um conjunto de treinamento (X_b, Y_b) , utilizando m atributos para escolha de cada nó. Gera o modelo final: $\hat{f}(x) = \sum_{b=1}^B f^b(x)$, que calcula os votos obtidos por cada modelo f^b , resultando uma classificação final de acordo com a votações majoritária.

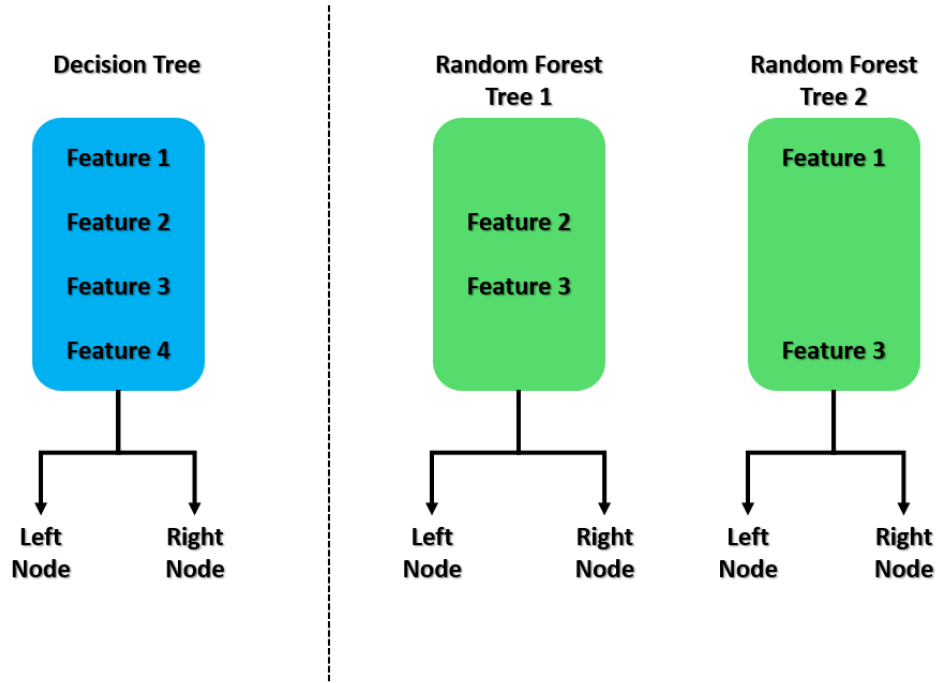


Figura 14: *Feature Selection Decision Tree*. Fonte: Adaptado de (Yiu, 2019)

As florestas aleatórias também oferecem um bom indicador de seleção de recursos. No *scikit-learn*, de acordo com Singh and Gupta (2014), a *feature importance* para árvores de decisão (*Decision Tree*) é calculada como a diminuição da impureza do nó ponderada pela probabilidade de atingir esse nó. A probabilidade do nó pode ser calculada pelo número de amostras que atingem o nó, dividido pelo número total de amostras. Quanto maior o valor, mais importante é o recurso.

Para cada árvore, o *scikit-learn* calcula a importância dos nós usando o *Gini Importance*, assumindo apenas dois nós filhos, como mostra a equação (31).

$$n_{ij} = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \quad (31)$$

Onde: n_{ij} é a importância do nó j , w_j é número ponderado de amostras que atingem o nó j , C_j é o valor da impureza do nó j , $left(j)$ é o nó filho da divisão esquerda no nó j e $right(j)$ é o nó filho da divisão direita no nó j .

A importância de cada recurso em uma árvore de decisão é calculada como mostra a equação (32).

$$f_{i_i} = \frac{\sum j : \text{node}_j \text{ splits on feature}_i n_{i_j}}{\sum k \in \text{all nodes } n_{i_k}} \quad (32)$$

Na equação (32) f_i é a importância da característica i e n_{i_j} é a importância do nó j . Os valores calculados a partir da equação (32) podem ser normalizados para um valor entre 0 e 1 dividindo pela soma de todos os valores de importância do recurso, como mostra a equação (33).

$$\text{norm}f_{i_i} = \frac{f_{i_i}}{\sum j \in \text{all features } f_{i_j}} \quad (33)$$

A importância final do recurso, no nível de uma floresta aleatória (*Random Forest*), é a média de todas as árvores. A soma do valor de importância do recurso em cada árvore é calculada e dividida pelo número total de árvores, como apresenta a equação (34).

$$RFf_{i_i} = \frac{\sum j \in \text{all trees } \text{norm}f_{i_j}}{T} \quad (34)$$

Na equação (34), RFf_{i_i} é a importância do recurso i calculado a partir de todas as árvores no modelo *Random Forest*, $\text{norm}f_{i_j}$ é a importância do recurso normalizado para i na árvore j e T é o número total de árvores. É assim que o ajuste ou precisão do modelo diminui quando você solta uma variável. Quanto maior a diminuição, mais significativa é a variável. Aqui, a diminuição média é um parâmetro significativo para a seleção de variáveis. O índice de Gini pode descrever o poder explicativo geral das variáveis.

5.3.3 Gradient Boosting Regression

Com o *Random Forest* conseguimos melhorar ainda mais o treinamento e buscar uma maior acurácia, quando comparado ao método *Decision Tree*. Uma vez que, o classificador *Random Forest* consiste na agregação do resultado de várias árvores de decisão.

No entanto, existe ainda um modelo bem mais sofisticado ao qual é possível melhorar ainda mais essa acurácia. O *Gradiente Boosting* utiliza o mesmo conceito do *Random Forest*, porém agora, as árvores são sequenciais. Após construir a primeira árvore de decisão, o *Gradient Boosting* verifica a probabilidade de erros e acertos e atribui um peso diferente aquela que obteve um maior erro.

A amostra de linhas para construção da segunda árvore depende do que aconteceu na primeira árvore. Aquela que tiver um maior erro, de acordo com a primeira árvore, passa a fazer parte da segunda. Isso acontece porque, pode acontecer que a previsão dessa amostra para um outro conjunto de dados, ainda não utilizado, pode ser melhor do que os que foram utilizados na primeira árvore.

Então, sempre irá existir uma amostra de linhas e colunas onde a segunda amostra das linhas depende do quanto o método está errando na anterior, ou seja, o método vai aumentando a chance dessa mesma amostra cair novamente na segunda.

Como resultado, é obtido uma média de todas as árvores de decisão que foram construídas ao longo do processo. Com isso, o que acontece é que, se o método está carregando uma determinada amostra ao qual está errando mais, ao ir chegando ao final do processo, ele começará a acertar e isso irá impactar em uma acurácia maior.

Uma maneira de demonstrar matematicamente o funcionamento desse método é através da regressão por mínimos quadrados, como é apresentado por Li (2019) ao considerar um modelo F , onde pretende-se prever os valores: $\hat{y} = F(x)$, de forma a minimizar o erro quadrático médio dado pela Equação 35.

$$\frac{1}{n} \sum_i (\hat{y}_i - y_i)^2 \quad (35)$$

Na Equação 36, $\hat{y}_i$ é o valor previsto pela função $F(x)$, y_i é o valor real da variável a ser prevista e n é o equivalente ao número de amostra do conjunto de dados em questão.

O método de *Gradient Boosting* realiza M etapas e sob cada etapa m ($1 \leq m \leq M$), adiciona um novo estimador $h_m(x)$ para melhorar o modelo F_m . Assim,

$$F_{m+1} = F_m(x) + h_m(x) = y \therefore h_m(x) = y - F_m(x) \quad (36)$$

Portanto, o *Gradient Boosting* ajustará h ao resíduo $y - F_m(x)$. Como em outras variantes de *boosting*, cada F_{m+1} tenta corrigir os erros de seu antecessor F_m .

5.3.4 Cross Validation Lasso

O Lasso é um algoritmo que busca selecionar e regularizar os melhores parâmetros, ou seja, aqueles que possuem uma boa capacidade preditiva. Para isso, exclui atributos que não contribuem de forma significativa para a predição da variável resposta, reduzindo seus coeficientes para zero e excluindo-os do modelo.

Para exemplificar como esse método funciona, partimos de uma representação de um conjunto de dados qualquer, como mostra a Figura 15.

Como sabemos, em um conjunto de dados, são realizadas n observações, de p variáveis explicativas, como mostrado na figura. Nesse exemplo, y_i é o valor da variável resposta na i -ésima observações enquanto x_{ij} é o valor da variável na i -ésima observações, $j = 1, \dots, p$. De uma forma geral, para se estimar y_i é utilizado a Equação (37) ao qual ajusta uma função linear aos dados do modelo.

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \varepsilon_i \quad (37)$$

Por meio da obtenção dos estimadores β aos quais representam a variação dos valores

y	x_1	x_2	$\dots$	x_p
y_1	x_{11}	x_{12}	$\dots$	x_{1p}
y_2	x_{21}	x_{22}	$\dots$	x_{2p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
y_n	x_{n1}	x_{n2}	$\dots$	x_{np}

Figura 15: Representação de um conjunto de dados

y em função da variação de uma unidade na variável x , é possível alcançar a minimização da soma de quadrados dos erros. Esse erro, está associado à distância entre o valor observado y_i e o valor correspondente do modelo proposto, para o mesmo nível i de x . O Lasso, consegue fazer isso adicionando um termo de penalidade λ , de forma que isso reduz o tamanho da Equação (38) (Tibshirani et al., 2005).

$$\sum_{i=1}^n (y_i - x'_i)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (38)$$

Na Equação (37), pelo método dos mínimos quadrados, a regressão linear visa estimar a variável dependente através de um conjunto de coeficientes, adicionando o termo de penalidade λ , em que valores próximos a 0 nós retornará aproximadamente a mesma solução dos mínimos quadrados padrão. Ao passo que, se aumentarmos tal parâmetro, obtemos uma solução que comprime até que, para um valor muito alto, todos os coeficientes tenderão a um valor nulo.

Quando aplicado a *Feature Selection*, em um conjunto de atributos altamente correlacionado (ou seja, *features* que se comportam da mesma maneira) a regularização Lasso seleciona apenas uma dessas *features* e zera os coeficientes das outras, de forma a minimizar o termo de penalização. Desse modo, esse modelo realiza *feature selection* automaticamente, gerando vários coeficientes com peso zero, ou seja, que são ignorados pelo modelo.

Com o Lasso, isso é calculado a partir de vários vetores de coeficientes que são gerados, tomam-se todos valores absolutos e calcula-se uma média. Como esse método é de regressão linear, estes coeficientes são iguais aos pesos de cada *feature*. Então quando um coeficiente é maior, maior será também o peso da *feature* correspondente (Pedregosa et al., 2011).

5.3.5 Principal Component Analysis

A Análise de Componentes Principais, segundo Serrano (2019) tem por objetivo converter um conjunto de dados grande, composto por várias variáveis contínuas, em um pequeno número de componentes as quais devem concentrar a maior quantidade de informação possível do conjunto de dados original. O método busca uma redução da dimensionalidade dos dados, ao agrupar os dados em grupos menores de acordo com a relação entre suas características.

De posse de um conjunto de dados de dimensão n versus p , o PCA busca centralizar os dados em (x_0, y_0) . Para isso, normaliza esses dados, subtraindo-os da média. Calcula a matriz de correlação de acordo com a dimensionalidade para os dados normalizados, essa matriz é simétrica, e de ordem p versus p como mostra a Equação 39.

$$S = \begin{bmatrix} \hat{Var}(x_1) & \hat{Covar}(x_1, x_2) & \hat{Covar}(x_1, x_3) & \cdots & \hat{Covar}(x_1, x_p) \\ \hat{Covar}(x_2, x_1) & \hat{Var}(x_2) & \hat{Covar}(x_2, x_3) & \cdots & \hat{Covar}(x_2, x_p) \\ \hat{Covar}(x_3, x_1) & \hat{Covar}(x_3, x_2) & \hat{Var}(x_3) & \cdots & \hat{Covar}(x_3, x_p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{Covar}(x_p, x_1) & \hat{Covar}(x_p, x_2) & \hat{Covar}(x_p, x_3) & \cdots & \hat{Var}(x_p) \end{bmatrix} \quad (39)$$

Encontra os autovetores e autovalores para a matriz de covariância, de acordo com a Equação 40, de tal forma que para a equação ter uma solução, é necessário que o determinante da matriz dos coeficientes seja zero, ou seja, $\det(S - \lambda I) = 0$.

$$S = \begin{bmatrix} \lambda_1 \cdot \hat{Var}(x_1) & \hat{Covar}(x_1, x_2) & \hat{Covar}(x_1, x_3) & \cdots & \hat{Covar}(x_1, x_p) \\ \hat{Covar}(x_2, x_1) & \lambda_2 \cdot \hat{Var}(x_2) & \hat{Covar}(x_2, x_3) & \cdots & \hat{Covar}(x_2, x_p) \\ \hat{Covar}(x_3, x_1) & \hat{Covar}(x_3, x_2) & \lambda_3 \cdot \hat{Var}(x_3) & \cdots & \hat{Covar}(x_3, x_p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{Covar}(x_p, x_1) & \hat{Covar}(x_p, x_2) & \hat{Covar}(x_p, x_3) & \cdots & \lambda_p \cdot \hat{Var}(x_p) \end{bmatrix} \quad (40)$$

O que resulta em um polinômio de grau n em λ , conhecido como polinômio característico. As raízes do polinômio característico são os autovetores da matriz S .

Após todos esses cálculos, o método ordena os autovetores por ordem de importância. Uma vez escolhidos os componentes (autovetores), o PCA multiplica os dados pelo autovetor(es) escolhidos. Um resumo de todas as etapas que o método PCA realiza para criar os componentes principais é apresentado na Figura 16.

Como resultado, obtém uma transformação nos dados de maneira que expressam os padrões entre eles. Os PCs (*Principal Components*) são combinações lineares de todas as características, produzindo assim novas características não correlacionadas.

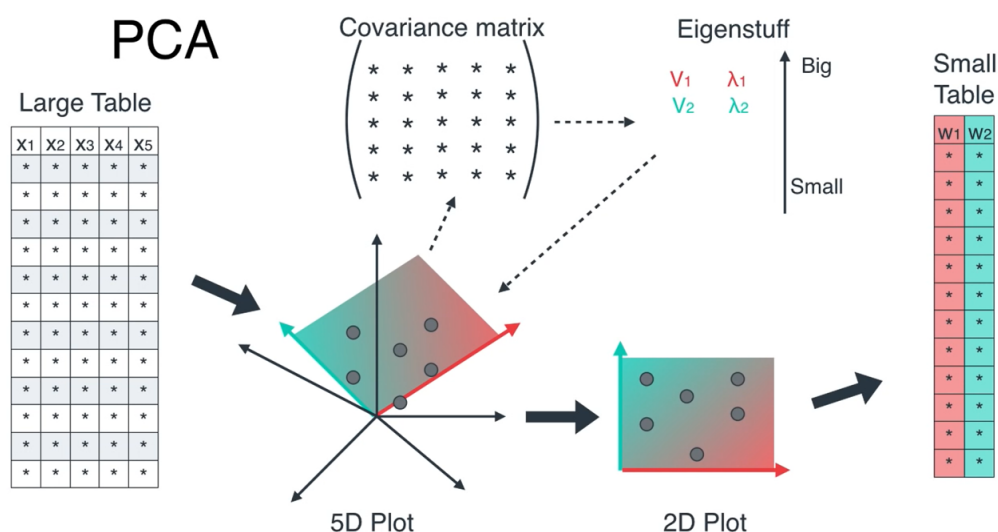


Figura 16: Funcionamento do Método PCA. Fonte: Adaptado de (Serrano, 2019)

5.3.6 Support Vector Regression

A análise de regressão é uma abordagem para modelar a relação entre um conjunto de variáveis Y (variáveis dependentes) e variáveis explicativas X (chamadas regressores ou ainda variáveis independentes). Um modelo que tem ganhando atenção ultimamente é o modelo de regressão por meio do SVM (*Support Vector Machine*) e é denominado *Support Vector Regression* (SVR). A ideia básica do SVR é mapear um conjunto de dados X em um espaço multidimensional através de um mapeamento não-linear (usualmente utilizando kernels) e então realizar uma regressão linear neste espaço multidimensional transformado.

Para um melhor entendimento desse algoritmo, segue em A) na Figura 17 uma representação gráfica de um conjunto de dados qualquer que pode ser modelado por uma regressão que parece ser polinomial como é apresentado por (Sayad, 2019). Neste exemplo, temos uma curva que sintetiza bem a regressão desse conjunto de dados. A regressão utilizando o método SVR aplica uma função, denominada Kernel. Essa função separa o conjunto de dados, colocando ele algumas dimensões acima, fazendo uma transformação na função que delimita cada um dos pontos do gráfico, ao passar de A) para B) como mostrado na Figura 17. Em B), temos o hiperplano que é a reta que delimita a divisão desse conjunto de dados e as retas de contorno que são os dois tracejados no gráfico da direita. O objetivo do SVM é obter duas destas retas de contorno que delimitem uma maior quantidade de dados possível dentro dessas retas.

Na regressão linear ou polinomial, em A na Figura, temos como objetivo delimitar uma reta se aproximando de pontos de tal forma que o erro, ou seja, a diferença entre a distância do ponto que está na reta e o ponto que está fora da reta seja a menor possível. Já o SVR, tenta controlar o erro de uma outra maneira, de forma que ponto do hiperplano

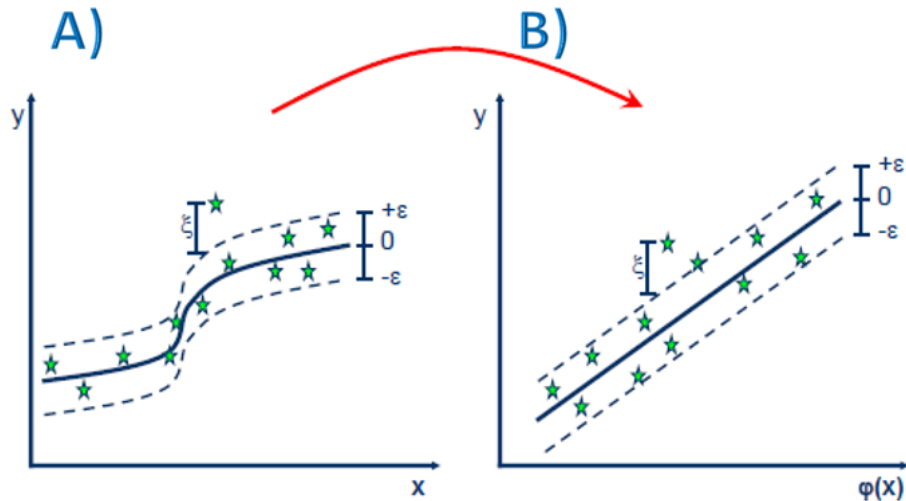


Figura 17: Visão geral do SVR. Fonte: Adaptado de (Sayad, 2019)

zero até o ponto onde se encontra a linha de contorno tenha o menor ε possível.

Em relação a regressão há muitas coisas em comum, mas matematicamente são representadas e computadas de maneiras diferentes. O SVM vai calcular tanto o hiperplano quanto o menor espaçamento possível entre as linhas de contorno, construindo uma curva ou uma reta (isso depende do tipo de Kernel). A tendência é que os pontos entre o hiperplano e a reta de contorno tenha o máximo de pontos dentro, esses pontos vão forçar a reta ou a curva ficar próxima dos pontos.

Existem várias funções de Kernel, mas neste trabalho, foi adotada a RFB - *Radial Basis Function*. Para esta função, é introduzido um parâmetro a mais no algoritmo que é a função γ que auxilia no ajuste dos pontos na reta. À medida que o valor de γ aumenta, o modelo fica super ajustado e a medida que o valor de γ diminui, o modelo começa a se tornar insuficiente.

5.3.7 Automatic Relevance Determination Regression

A Determinação Automática de Relevância (Automatic Relevance Determination - ARD) como apresenta Qi et al. (2004) é um método derivado de técnicas Bayesianas que utilizam redes neurais. A diferença é que o ARD, utiliza densidades de probabilidades. A densidade de probabilidade é modelada por função de densidade $f(x)$ com as seguintes propriedades:

- (a) o valor de $f(x)$ é sempre igual ou superior a zero;
- (b) a área sob a curva estabelecida pela densidade e delimitada pelo eixo da abscissa é igual à uma unidade, se for considerado o domínio da variável X . Essa área representa a probabilidade de X estar no intervalo $[a, b]$.

Esse tipo de função, pode ser representada graficamente por uma curva em formato de sino, como mostra a Figura 18 a seguir.

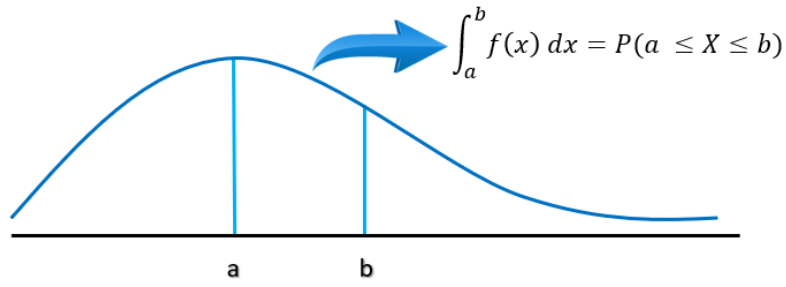


Figura 18: Distribuição de probabilidade

O ARD consiste em inicialmente treinar uma rede neural com todas as variáveis de entrada usando o método Bayesiano. Esse método, tenta prever um determinado atributo alvo desconsiderando completamente a correlação entre os demais atributos do conjunto de dados. O algoritmo então, toma os valores $(x_1, x_2, \dots, x_n)$ dos atributos e calcula a probabilidade $P(Y = y_m | X = x_1, x_2, \dots, x_n)$. Ao final, irá selecionar o y_k , $0 < k < m + 1$, cuja probabilidade for maior. Como o algoritmo trata cada atributo de forma independente, pode-se dizer que $P(x_1, \dots, x_n | y) = P(x_1 | y) * P(x_2 | y) * \dots * P(x_n | y)$, resultando na Equação 41.

$$P(y | x_i, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)} \quad (41)$$

Como $P(x_1, \dots, x_n)$ é constante, de acordo com a entrada do conjunto de atributos, obtém-se a Equação 42.

$$P(y | x_i, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y) \quad (42)$$

O algoritmo escolherá o y cuja probabilidade de $P(y | x_i, \dots, x_n)$ for maior. Assim, chegamos na Equação 43 ao qual representa o Classificador de Naive Bayes.

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^n P(x_i | y) \quad (43)$$

Com essa equação, todas as probabilidades são calculadas e um vetor de combinações com todas as possibilidades do treinamento é montado. Essa distribuição é parametrizada por vetores $\Theta_{y_i} = \Theta_{y_1}, \dots, \Theta_{y_n}$ é a probabilidade do evento i ocorrer, dado o atributo alvo que é y . Podemos dizer que cada vetor Θ_y representa uma observação e n é o número de *features*. Assim, $\Theta_{y_i} = P(x_i | y)$ e é estimado pela Equação 44.

$$\hat{\Theta}_{y_i} = \frac{N_{y_i} + \alpha}{N_y + \alpha n} \quad (44)$$

N_{y_i} é o número de vezes que o atributo x_i aparece no conjunto de treinamento, N_y é o número de observações em relação ao atributo alvo y , n é o número de atributos e alfa

é um parâmetro de correção que consiste em adicionar dados à base de treinamento para que não haja probabilidades iguais a 0.

O ARD segue esta metodologia e para cada *feature* atribui um peso associado a um parâmetro α distinto de acordo com a sua relevância em prever o atributo alvo. O cálculo da relevância de cada uma das *features* para treinar o modelo é dada pela Equação 45 (PUC-Rio, 2019).

$$ri(\%) = 100 \left(\frac{\frac{1}{\alpha_i}}{\sum_{j=1}^k \frac{1}{\alpha_j}} \right) \quad (45)$$

O ARD considera inicialmente que a soma das variáveis de entrada seja 100%. Então, após treinar o modelo com uma rede neural utilizando o método Bayesiano, ele ordena ambos os atributos e aqueles com uma medida de importância menor que 1% são excluídos. Esse processo é repetido até que não restem atributos com importâncias menores que 1% (PUC-Rio, 2019). Como o processo é repetido muitas vezes, isso faz com que este método tenha um maior custo computacional quando comparado aos demais métodos apresentados nesta seção.

5.4 *Hyperparameter Tuning*: Técnicas de otimização de hiper-parâmetros

Parâmetros que são definidos antes do treino são chamados de hiperparâmetros, e são diferentes de valores internos do modelo que vão sendo alterados de acordo com o que o modelo está aprendendo. Para saber quais são os melhores parâmetros (quantos e quais dimensões), existe no Skicit-learn algumas formas de buscar a otimização de parâmetros em *grid*. Uma otimização significa que o modelo vai ser melhor e assim as importâncias estarão mais próximas da realidade.

No Skicit-learn, de acordo com Dei (2019), as técnicas mais conhecidas são: o *Grid Search*, o *Random Search* e o *Bayes Search*. O *Grid Search* realiza a busca dos melhores parâmetros a partir de uma tabela de valores. Esse algoritmo testa todas as combinações e verifica quais possuem um melhor desempenho. Ao final, são selecionados aqueles valores que tiveram uma maior performance. O *Random Search* assemelha-se ao funcionamento do *Grid Search*, porém, os testes são realizados a partir de combinações aleatórias, ou seja, o algoritmo não testa todas as combinações possíveis como o *Grid Search*. Por isso, ele é utilizado quando temos um conjunto de dados ou atributos muito grande, pois nesse caso, é uma alternativa computacionalmente mais rápida.

Já o *Bayes Search* funciona de forma bem diferente dos métodos anteriormente citados. Esse método realiza estimativas para estimar qual é a melhor combinação de parâmetros para o modelo. Ele procura as regiões onde há menor confiança na distribuição levantada e, dentro dessas regiões, qual é a que pode ter um valor mais elevado para a performance. Embora o ganho de desempenho em relação aos demais algoritmos apre-

sentados, seja muito maior, o seu tempo de execução é bem mais demorado. Esse método funciona muito bem para treinamentos mais complexos e com espaços de busca muito grandes.

6 TRABALHOS RELACIONADOS

Nesta seção serão apresentados alguns estudos relacionados à utilização de técnicas de aprendizado de máquina, seleção de atributos e *Ensemble* relacionados ao desenvolvimento da proposta apresentada e que mostram exemplos de como o problema é atualmente tratado.

Um dos trabalhos ao qual assemelha-se aos métodos utilizados neste estudo é Lopez-Rincon et al. (2019). Porém, o foco deste trabalho diferentemente da proposta aqui apresentada, busca investigar o uso de técnicas de *Feature Selection* e *Ensemble* para descobrir e validar uma lista reduzida de miRNAs a serem usados como assinatura para classificação de tumores e distinguir entre tecido normal e tecido tumoral.

Neste sentido, Lopez-Rincon et al. (2019) apresentam uma metodologia que visa combinar o uso de múltiplos algoritmos para obter um desempenho preditivo mais robusto e geral para predição de miRNAs com melhor acurácia. Os resultados mostraram que o modelo proposto melhora a precisão da classificação em comparação com a simples seleção do melhor classificador da combinação. Finalmente, o método proposto foi combinado com os resultados do uso de *Bagging* e *Boosting*, no qual obteve resultados que foram significativamente melhores do que os classificadores *ensemble* individuais.

As técnicas utilizadas por Lopez-Rincon et al. (2019) para seleção de atributos incluíram: *Univariate Feature Selection* (UFS), *Recursive Feature Elimination* (RFE), *Elastic Net* (EN), *Genetic Algorithms* (GALGO), *Least Absolute Shrinkage and Selection Operator* (LASSO) e *Ensemble Feature Selection - Complete Linear Aggregation* (EFSCLA). Todos os métodos considerados foram implementados na linguagem de programação Python utilizando o pacote de aprendizado de máquina *scikit-learn*, assim como neste trabalho.

Com estes classificadores, os miRNAs mais relevantes foram extraídos para serem usados como recursos para a classificação do câncer. Uma abordagem *ensemble* foi então aplicada, porém, nem todos os classificadores usaram os mesmos recursos para separar os tipos de câncer.

Para esse fim, um conjunto de classificadores foi treinado para extrair uma lista classificada dos recursos mais relevantes de cada um. A partir de uma comparação de 22

classificadores que foram treinados utilizando o conjunto de dados considerado, um subconjunto desses classificadores foi selecionado.

Foram escolhidos 8 classificadores, todos com uma precisão média superior a 90% em uma validação cruzada de 10 vezes: *Bagging*, *Gradient Boosting*, *Logistic Regression*, *Passive Aggressive*, *Random Forest*, *Ridge*, *Stochastic Gradient Descent on linear models* e *Support Vector Machines Classifier with a linear kernel*. Dentre os quais, implementados na linguagem de programação Python, utilizando a biblioteca *scikit-learn*.

Foi demonstrado por Lopez-Rincon et al. (2019) que dados acumulados sobre os níveis de expressão de miRNA em tumores e seu alto grau de envolvimento no desenvolvimento e no progresso do câncer identificam essas moléculas como candidatas promissoras para orientar o tratamento do câncer, e também para a previsão dos resultados da quimioterapia. Essa análise de miRNA, utilizando técnicas de *Ensemble* e *Feature Selection* pode ser uma ferramenta promissora para determinar a sensibilidade de tumores à quimioterapia, tornando os biomarcadores de miRNA de grande importância na prática clínica.

Uma outra pesquisa que relata a aplicação de técnicas de *Feature Selection* e *Ensemble* alinhada a metodologia proposta neste estudo é Gao et al. (2017). Os pesquisadores descrevem uma abordagem para identificar as características gerais que definem a diferença entre um coração saudável e um que sofreu um infarto agudo do miocárdio.

Neste estudo, é proposta uma metodologia que visa encontrar os fatores biomecânicos que possuem o maior poder explicativo para a diferenciação da função contrátil do miocárdio de um coração saudável ou um que sofreu um infarto agudo. Para isso, foram utilizados os métodos: *Lasso CV*, *Decision Tree*, *Random Forest* e *Gaussian Process with Automatic Relevance Determination* (GP-ARD).

A medida de importância dos métodos citados anteriormente, foram combinados em uma pontuação de relevância de classificação única para identificar os fatores biomecânicos com o maior poder explicativo. Para cada um dos métodos utilizados, classificou-se as variáveis explicativas e combinou-se os resultados em uma pontuação acumulada, que serviu como um indicador para a relevância geral do fator. Todos os métodos utilizados forneceram uma medida da importância destes fatores.

De acordo com os resultados de Gao et al. (2017), é possível desenvolver um sistema de suporte à decisão clínica usando os algoritmos avaliados. Esse sistema ajudaria especialistas em cardiologia no diagnóstico automático de patologias cardíacas. A relevância do estudo está em apresentar meios que suscitem maiores estudos no futuro que possam levar ao desenvolvimento de uma ferramenta de diagnóstico inteligente. Através dessa modelagem, a metodologia proposta conseguiu alcançar mais de 90% de precisão na classificação com alguns classificadores.

Além destes estudos, também damos destaque a revisão de Shen et al. (2020), a qual aborda alguns métodos de aprendizado de máquina amplamente empregados para

predição da afinidade de ligação e que também foram utilizados em nosso trabalho.

De acordo com os pesquisadores, existe uma variedade de funções de pontuação que são baseadas em métodos como o *Random Forest*, contudo eles destacam: RF-Score, B2BScore e SFCscore por terem superado uma ampla gama de funções clássicas.

Os autores também destacam as funções de pontuação que utilizam o método *Support Vector Machine*, como a SVR-SFs e ID-Score para predição da afinidade de ligação aos quais obtiveram os melhores resultados para o poder de pontuação utilizando este método.

Além destes métodos, salienta-se o *Gradient Boosting* ao qual tem sido amplamente empregado no campo da descoberta de novos medicamentos e recentemente também envolvido no desenvolvimento de funções de pontuação, como a XGB-Score, que até mesmo superou sua função de pontuação original, a RF-Score. Os pesquisadores também fazem citação as funções: BT-Score, BT-Dock e BT-Screen e afirmam que estas poderiam ter o maior poder de pontuação, encaixe e triagem do que outras funções de pontuações convencionais, apesar do fato de que o BT-Score não teve um bom desempenho em termos de poder de ancoragem. Os autores por último citam: T-Bind, TopBP-ML e EIC-Score as quais obtiveram escores consideravelmente altos para o poder de pontuação.

Com esse mesmo enfoque, o estudo de Wang and Zhang (2017) abordaram a avaliação de uma nova função de pontuação, denominada $\Delta VinaRF_{20}$, que emprega vinte descritores, além da pontuação de um programa utilizado para acoplamento molecular, denominado AutoDock Vina. Essa função de pontuação alcançou um desempenho superior em todos os testes de potência dos benchmarks CASF-2013 e CASF-2007. Isso permitiu a definição do conjunto de treinamento utilizado neste trabalho.

Para o desenvolvimento da função $\Delta VinaRF_{20}$, os autores fizeram uma pré-seleção das características físico químicas a serem utilizadas, ao tomar todas as 58 atributos do código fonte do programa AutoDock Vina junto com 10 atributos relacionados a área superficial acessível ao solvente (bSASA). Os pesquisadores testaram o desempenho da função de pontuação em diferentes conjuntos de atributos, de diferentes tamanhos e chegaram a conclusão que 10 atributos de Vina e 10 de bSASA melhoraram significativamente os resultados. Wang and Zhang (2017) afirmam que $\Delta VinaRF_{20}$ foi capaz de alcançar um desempenho superior em comparação com as funções de pontuação clássicas em todos os testes.

No geral este capítulo, apresentou trabalhos relacionados a utilização de técnicas de aprendizado de máquina implementadas na linguagem de programação Python, utilizando a biblioteca *scikit-learn*, que provou ser de alto desempenho e robusta (Pedregosa et al., 2011).

Na Tabela 3 é apresentado um resumo dos trabalhos que foram citados ao longo deste capítulo.

Tabela 3: Trabalhos Correlatos - Resumo

Pesquisadores	Métodos utilizados	Descrição
Lopez-Rincon et al. (2019)	Técnicas de feature selection como Lasso e Ensemble	Métodos computacionais usam Python utilizando o pacote de aprendizado de máquina Scikit-learn.
Gao et al. (2017)	Lasso, DT, RF, GP-ARD	Medidas de importância combinadas (<i>ranking</i>) para identificar atributos com maior relevância.
WANG; ZHANG (2017)	<i>Random Forest</i>	Utiliza o algoritmo de aprendizado de máquina <i>Random Forest</i> para <i>reescore</i> da função do AutoDock Vina.
Shen et al. (2020)	<i>Random Forest, Suport Vector Machine, Gradient Boosting</i>	Revisão da literatura sobre o progresso das funções de pontuação baseadas em ML nos últimos anos.

É importante destacar que a correlação observada entre as metodologias levantas nesta seção subsidiaram a proposta deste trabalho, servindo como base para modelagem e implementação deste estudo.

7 METODOLOGIA

Este estudo busca implementar e analisar uma metodologia baseada em métodos de *Machine Learning*, para auxiliar na redução da dimensionalidade dos dados que caracterizam as interações entre complexos proteína-ligante, prevenindo a possibilidade de *overfitting*, através da seleção das propriedades físico químicas com maior grau de importância e da utilização de métodos de *Ensemble* para melhorar o desempenho geral das predições.

Tendo em vista a metodologia a ser adotada, propõe-se o conjunto de etapas apresentado na Figura 19, que servirá de base para o desenvolvimento deste trabalho.

- Selecionar métodos de *Machine Learning* para a identificação das características físico-químicas e geométricas de complexos proteína-ligante por ordem de importância;
- Ranquear a importância desses atributos para cada método (*feature importance*);
- Combinar a medida de importância em um *ranking* acumulativo;
- Determinar os subconjuntos de *features* (*feature subset selection*) que possuem a maior importância;
- Realizar a predição de Δp_{k_d} e p_{k_d} para calcular a energia livre de ligação entre complexos proteína-ligante;
- Gerar um *ensemble score* para combinar em uma única abordagem os resultados obtidos através da utilização de diferentes modelos preditivos;
- Realizar a otimização dos parâmetros para melhorar a precisão na etapa de *feature importance* e no *ensemble*;
- Avaliar as predições do *ensemble* e de cada um dos métodos individualmente em relação as métricas de poder de classificação, poder de pontuação e poder de ancoragem em referência ao CASF-2016.

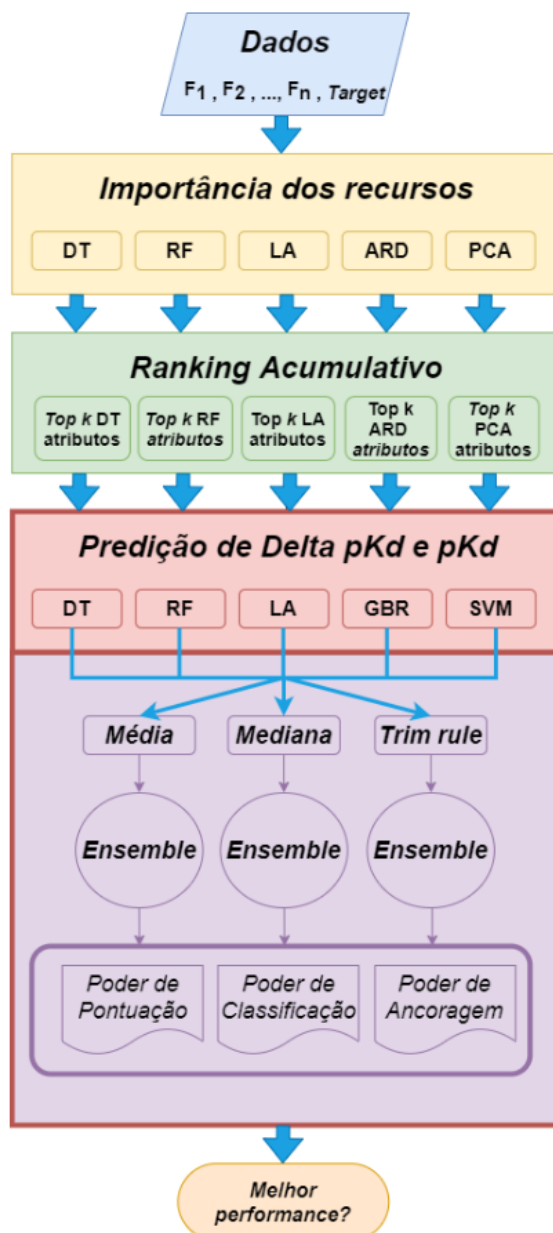


Figura 19: Metodologia aplicada

Primeiramente, foram calculadas as medidas de importância de cada um dos atributos utilizando os métodos *Decision Tree*, *Random Forest*, Lasso, ARD e PCA. Essas medidas foram combinadas em um ranking acumulativo. Utilizamos essa classificação para realizar uma sub seleção do conjunto de *features* e melhorar a predição da afinidade de ligação, ao combinar os resultados em um sistema *Ensemble*.

Em mais detalhes, depois de escolher os “ k ” atributos com o maior importância, foram aplicados métodos de *Ensemble*, mas para diferentes tamanhos k . Como resultado da combinação do *Ensemble* foi obtido três *scores* finais (média, mediana e *trim rule*), em relação ao poder de pontuação e ao poder de classificação.

Além destas etapas, foi realizada uma otimização dos parâmetros do modelo desen-

volvido. Esse processo teve por objetivo tentar melhorar o desempenho das predições em cada uma das etapas da metodologia que foi aplicada neste trabalho. Após, foi realizado um levantamento para verificar quais métodos, combinados a quais conjuntos de atributos obtiveram os melhores resultados. Por fim, avaliamos o modelo preditivo proposto com as outras funções de escore de acordo com as avaliações do CASF-2016.

8 DADOS E SIMULAÇÕES

Neste capítulo, serão apresentados os experimentos que foram realizados durante este estudo, bem como as bases para treinamento e teste e o conjunto de descritores que foram utilizados. Será mostrado os resultados para as etapas de *Feature Importance* e a agregação das medidas de importância através de um *ranking* acumulativo. A seguir, apresentamos o *Ensemble*

8.1 Base de dados para treinamento e teste

Neste trabalho foi utilizada a base de treinamento do PDBbind *refined set* de 2018 e para teste o conjunto de dados do CASF-2016. Foram utilizados os conjuntos de atributos do DeltaVinaRF20, AutoDock Vina e os descritores gerados com o RDKit 2D. Ambos os conjuntos de dados utilizados são descritos a seguir.

8.2 PDBbind refined set 2018

O banco de dados PDBbind fornece uma coleção abrangente dos dados de afinidade de ligação medidos experimentalmente para todos os tipos de complexos biomoleculares depositados no Protein Data Bank (PDB). Cada versão do PDBbind inclui um “conjunto geral”, que contém todas as estruturas de ligantes de proteínas no banco de dados; e um “conjunto refinado”, um subconjunto de complexos proteína-ligante que satisfaz critérios rígidos em relação à qualidade da estrutura, confiabilidade dos dados de afinidade e natureza do complexo. A versão 2018 do PDBbind contém 16.151 complexos proteína-ligante no conjunto geral, com 4.463 complexos no conjunto refinado. Neste trabalho, usamos o conjunto refinado como nossa principal fonte de dados de treinamento.

8.3 CASF-2016

O CASF foi criado como uma referência de acesso aberto para que outros pesquisadores possam utilizá-lo para testar uma gama mais ampla de funções de pontuação, é uma referência de acesso aberto para que outros pesquisadores possam utilizá-lo para

testar uma gama mais ampla de funções de pontuação (Su et al., 2018). A versão mais recente, de 2016 consiste em 285 complexos proteína-ligante com estruturas cristalinas de alta qualidade e constantes de ligação confiáveis, extraídas do conjunto refinado de dados do PDBbind. Em nosso trabalho, esse conjunto de dados foi utilizado para testar nosso modelo e avaliar nossos resultados, permitindo que fossem comparados diretamente com referências de funções de pontuação publicadas recentemente. Um painel de 25 funções de pontuação é testado no CASF-2016 como uma demonstração e foram comparadas aos nossos melhores resultados.

8.4 Conjunto de atributos utilizados

As funções de pontuação baseadas em aprendizado de máquina usualmente propostas dependem de uma ampla gama de características (químicas, físicas, físico-químicas, estruturais, geométricas ou de conservação de estrutura primária) que caracterizam inteiramente os complexos proteína-ligante. A seguir, são apresentados os conjuntos de atributos que foram aplicados aos fatores de correção da função de pontuação do DeltaVinaRF20.

8.5 Atributos do DeltaVinaRF20

Nesse trabalho, consideramos os vinte atributos utilizados por Wang and Zhang (2017) no desenvolvimento da função de pontuação $\Delta V_{\text{VinaRF20}}$, dentre os quais estão listados no Anexo A.1 (Tabela 7). Esse conjunto é composto de 10 termos do código fonte do AutoDock Vina e 10 termos relacionados a energia de solvatação, que estima a energia da interação soluto-solvente determinada com base no cálculo da área de superfície de acessibilidade (SASA, do inglês *Solvent Accessible Surface Area*). Cada atributo foi mapeado a partir de um identificador para melhor representação dos resultados. Além deste conjunto de atributos, foram utilizados também os conjuntos de descritores do AutoDock Vina e do RDkit 2D, aos quais são detalhados a seguir.

8.6 Atributos do AutoDock Vina

Um dos mecanismos de acoplamento de código aberto mais citados é o AutoDock Vina. Ele usa algoritmo genético para procurar a pose mais favorável em termos de energia de uma molécula pequena flexível em um local de ligação rígido ou flexível de uma proteína (Shamsara, 2018).

No código fonte do AutoDock Vina, há um total de 58 atributos implementados, como mostra a Tabela 8 no Anexo A.2 aos quais foram utilizados neste trabalho e que descrevem termos gaussianos, termos hidrofóbicos e não hidrofóbico, ligações de hidrogênio, contagem de propriedades simples, termos eletrostáticos, termos de dessolvatação e um

termo Lennard-Jones van der (Wang and Zhang, 2017).

8.7 Atributos gerados com RDKit 2D

O RDKit¹ é um software de quiminformática de código aberto e gerador de descritor para aprendizado de máquina. O site pode calcular vários recursos, incluindo SMILES canônicos, representação 2D, impressões digitais, reações químicas, serialização molecular, seleção de similaridade/diversidade e descrições 2D e 3D para moléculas de drogas. O software está sendo atualizado continuamente a partir de 2012. Neste trabalho, utilizamos os atributos 2D das moléculas geradas com o RDkit. Um total de 147 atributos físico-químicos foram aplicadas em nossos experimentos e podem ser observadas em mais detalhes no Anexo A.3 (Tabela 9).

8.8 Atributos alvo

Neste estudo, foi utilizado dois tipos de atributos alvo: p_{k_d} que como anteriormente já mencionado, consiste nos valores das afinidades de ligação experimentais e o termo de correção Δp_{k_d} utilizado na função de pontuação proposta por Wang and Zhang (2017), para estimar a afinidade de ligação entre os complexos proteína-ligante.

Na equação (46), $p_{k_d(Vina)}$ é a afinidade de ligação calculada pelo *software* AutoDock Vina e $\Delta p_{k_d}(RF)$ é um termo de correção que tem como objetivo realizar um *rescoring*, para tentar melhorar os pontos fracos da função utilizada no Autodock Vina. Os pontos fracos são sua predição ao valor de afinidade de ligação principalmente.

$$p_{k_d}(\Delta Vina RF) = p_{k_d(Vina)} + \Delta p_{k_d}(RF) \quad (46)$$

Isolando os termos da equação (46), pode-se determinar o fator de correção, a partir da equação (47). O termo de correção $\Delta p_{k_d}(RF)$ é treinado pelo algoritmo de floresta aleatória (RF) utilizando-se um conjunto de treinamento de complexos proteína-ligante, isto é, $p_{k_d(train)} - p_{k_d(Vina)}$.

$$\Delta p_{k_d}(RF) = p_{k_d(train)} - p_{k_d(Vina)} \quad (47)$$

¹<http://www.rdkit.org/>

9 RESULTADOS

9.1 *Feature Importance e Cumulative Ranking*

Na etapa de *feature importance*, utilizamos os algoritmos: *Decision Tree*, *Random Forest*, *Lasso CV*, *PCA* e *ARD* que foram implementados na linguagem de programação Python, utilizando o pacote de aprendizado de máquina *scikit-learn*. Realizamos este experimento para analisar os atributos mais importantes ao aplicar os fatores de correção p_{k_d} e Δp_{k_d} da função DeltaVina RF20.

Os gráficos da representação da realização do *feature importance* no Anexo A.4, apresentam os atributos mais importantes em relação a Δp_{k_d} e p_{k_d} . Devido a seus valores muito pequenos, nem todos os atributos não foram incluídos nas figuras. A partir dessa representação, é possível verificar a relação entre os valores de ambos atributos alvo, aos quais foram treinados pelos algoritmos: *Decision Tree*, *Random Forest*, *Lasso* e *ARD* para cada um dos 205 atributos. Nestes gráficos, é possível observar as características físico-químicas mais importantes para cada um dos fatores de correção.

Nos gráficos, o comprimento das barras representam o valor medido: quanto maior a barra, maior o grau de relação entre as variáveis. Para isso, a função apresentada através da equação (36) teve como finalidade verificar se $p_{k_d(\text{train})}$ se aproxima de $p_{k_d(\text{Vina})}$. Como até agora foi realizado um estudo de atributos, é possível verificar de acordo com essa análise quais atributos são mais relevantes para calcular os fatores de correção utilizados.

Após essa etapa, o conjunto de atributos foi então representado através de um *ranking* acumulativo com objetivo de identificar as características físico químicas mais relevantes. Na sequência, são apresentados os resultados para Δp_{k_d} para a combinação das medidas de importância de acordo com os métodos de ML aplicados neste estudo. O gráfico original, foi dividido em cinco partes: Figura 20 (apresenta a importância dos atributos mais relevantes de 1 a 41), Figura 21 (apresenta a importância dos atributos mais relevantes de 42 a 82), Figura 22 (apresenta a importância dos atributos mais relevantes de 83 a 123), Figura 23 (apresenta a importância dos atributos mais relevantes de 124 a 164) e a Figura 24 (apresenta a importância dos atributos mais relevantes de 165 a 205).

O mesmo procedimento foi realizado utilizando p_{k_d} . Os resultados podem ser observa-

dos a partir dos gráficos das figuras a seguir, aos quais representam o *ranking* acumulativo para este fator de correção. Na Figura 25 (apresenta a importância dos atributos mais relevantes de 1 a 41), Figura 26 (apresenta a importância dos atributos mais relevantes de 42 a 82), Figura 27 (apresenta a importância dos atributos mais relevantes de 83 a 123), Figura 28 (apresenta a importância dos atributos mais relevantes de 124 a 164) e a Figura 29 (apresenta a importância dos atributos mais relevantes de 165 a 205).

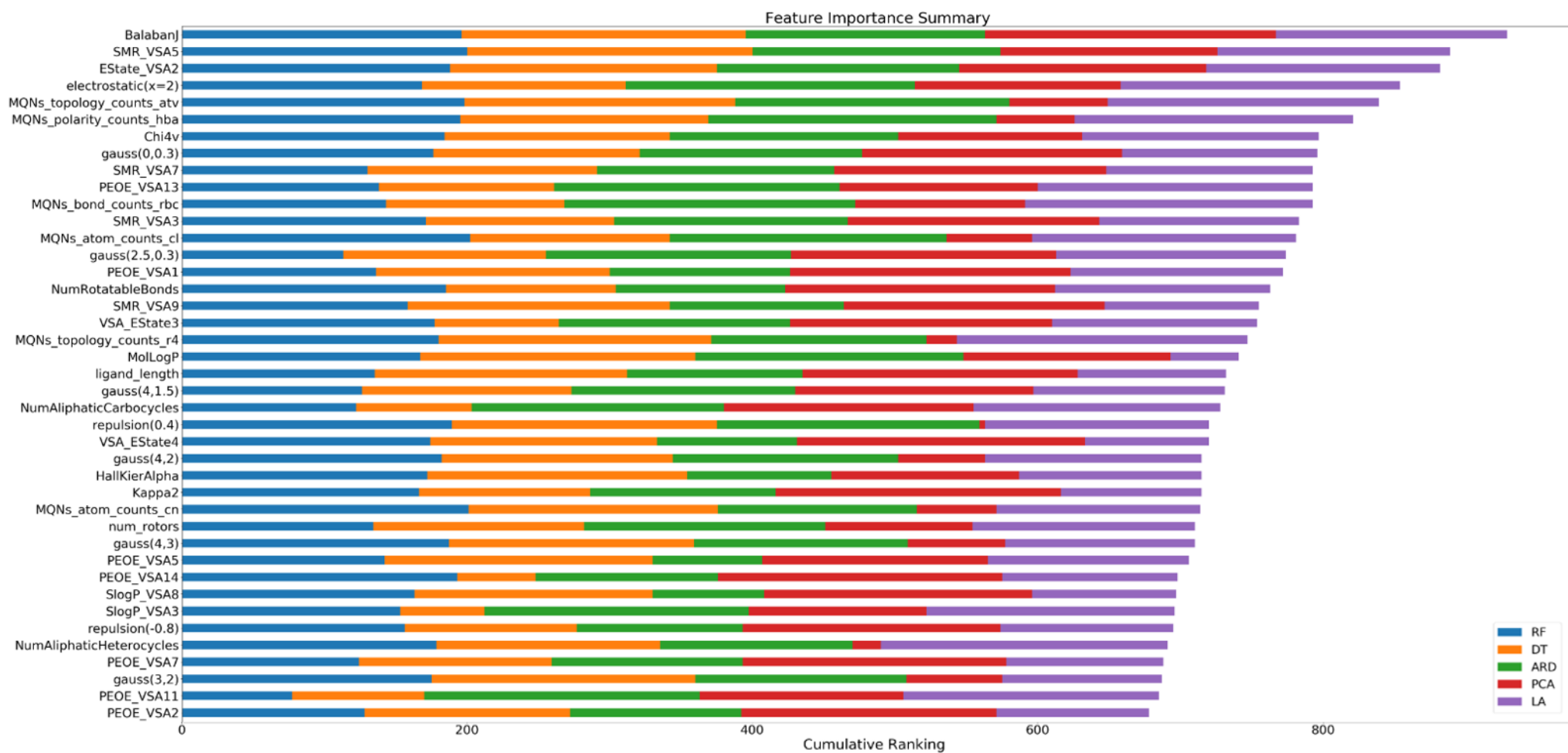


Figura 20: Ranking acumulativo para Δp_{k_d} (1 a 41)

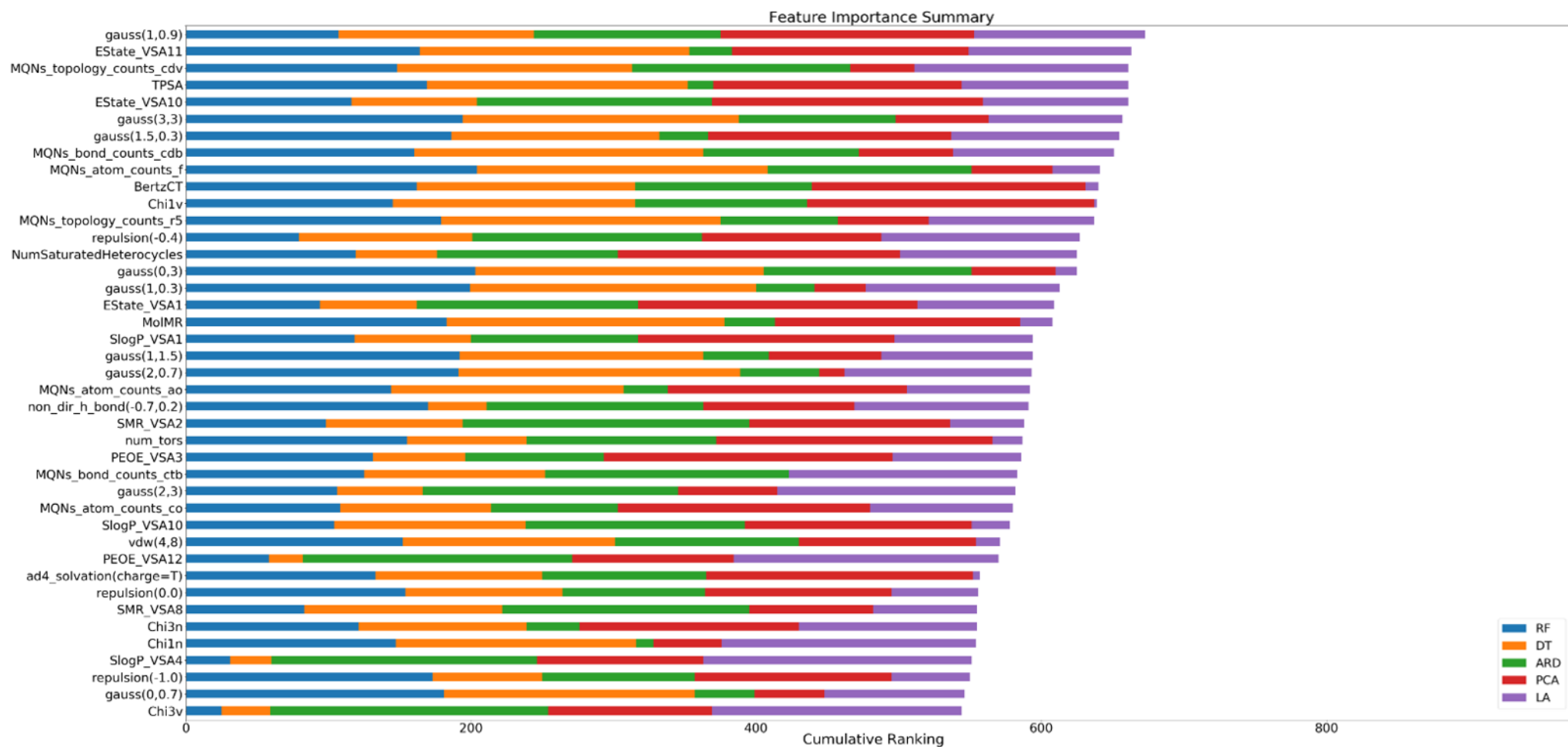


Figura 21: Ranking acumulativo para Δp_{k_d} (42 a 82)



Figura 22: Ranking acumulativo para Δp_{k_d} (83 a 123)

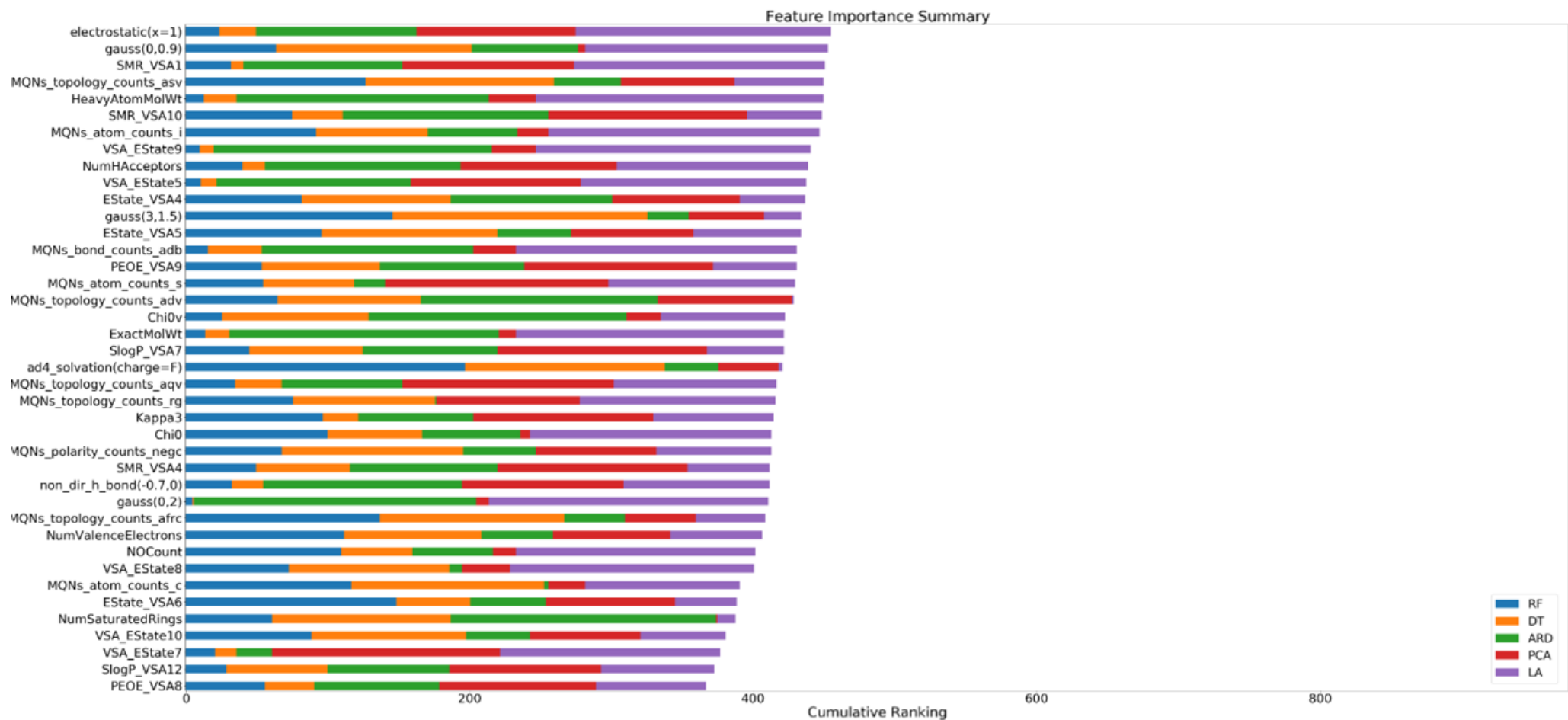


Figura 23: Ranking acumulativo para Δp_{k_d} (124 a 164)

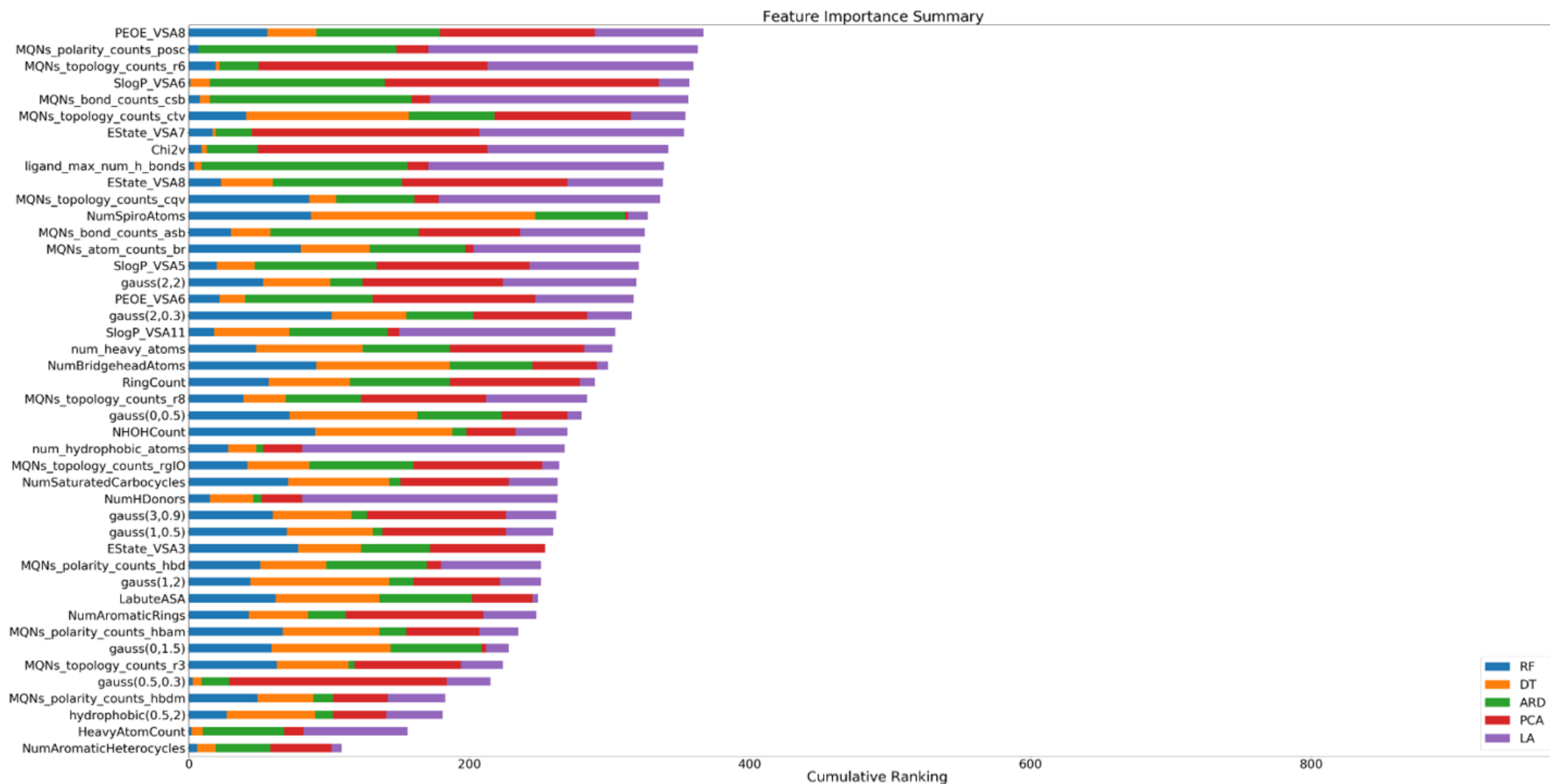


Figura 24: Ranking acumulativo para Δp_{k_d} (165 a 205)

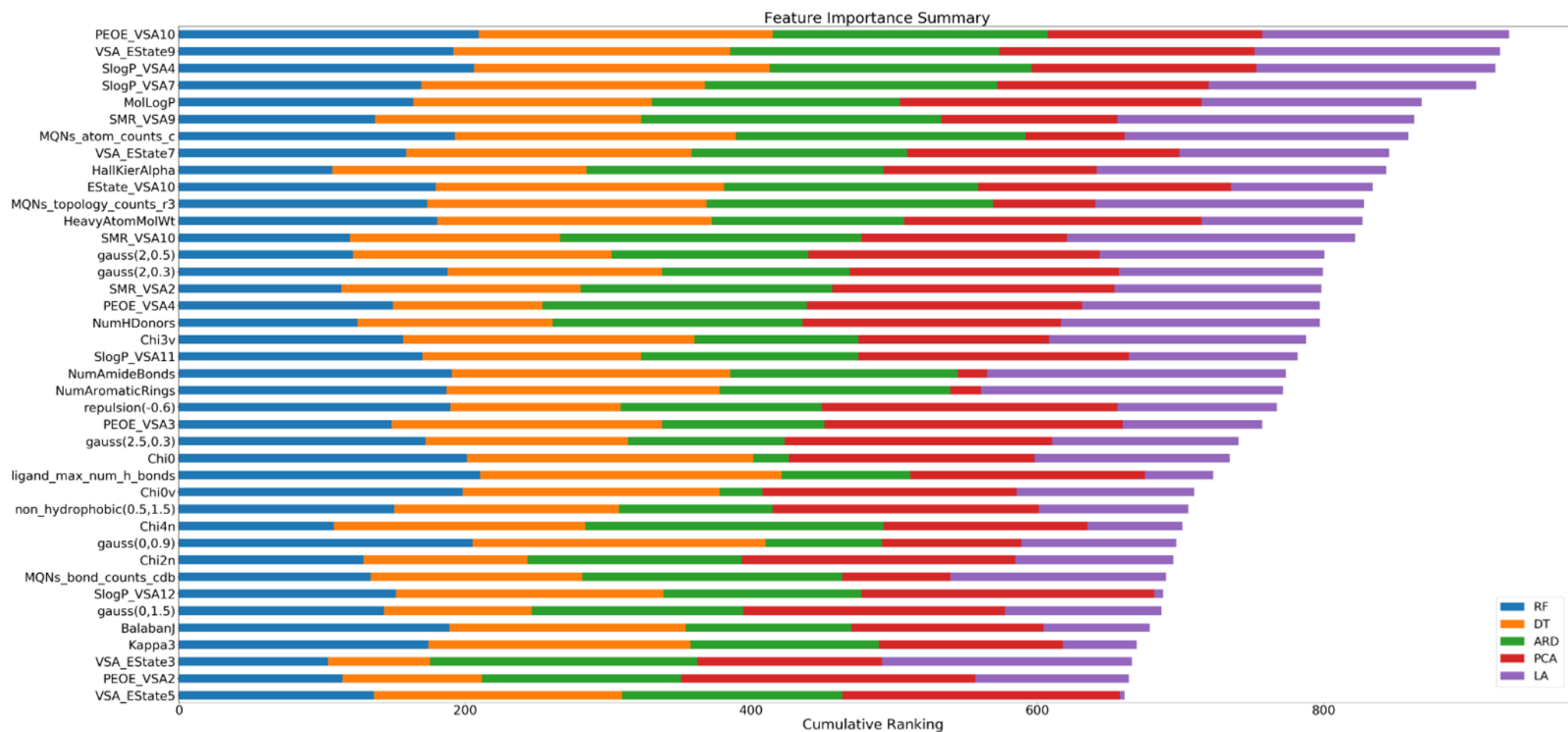


Figura 25: Ranking acumulativo para p_{k_d} (1 a 41)

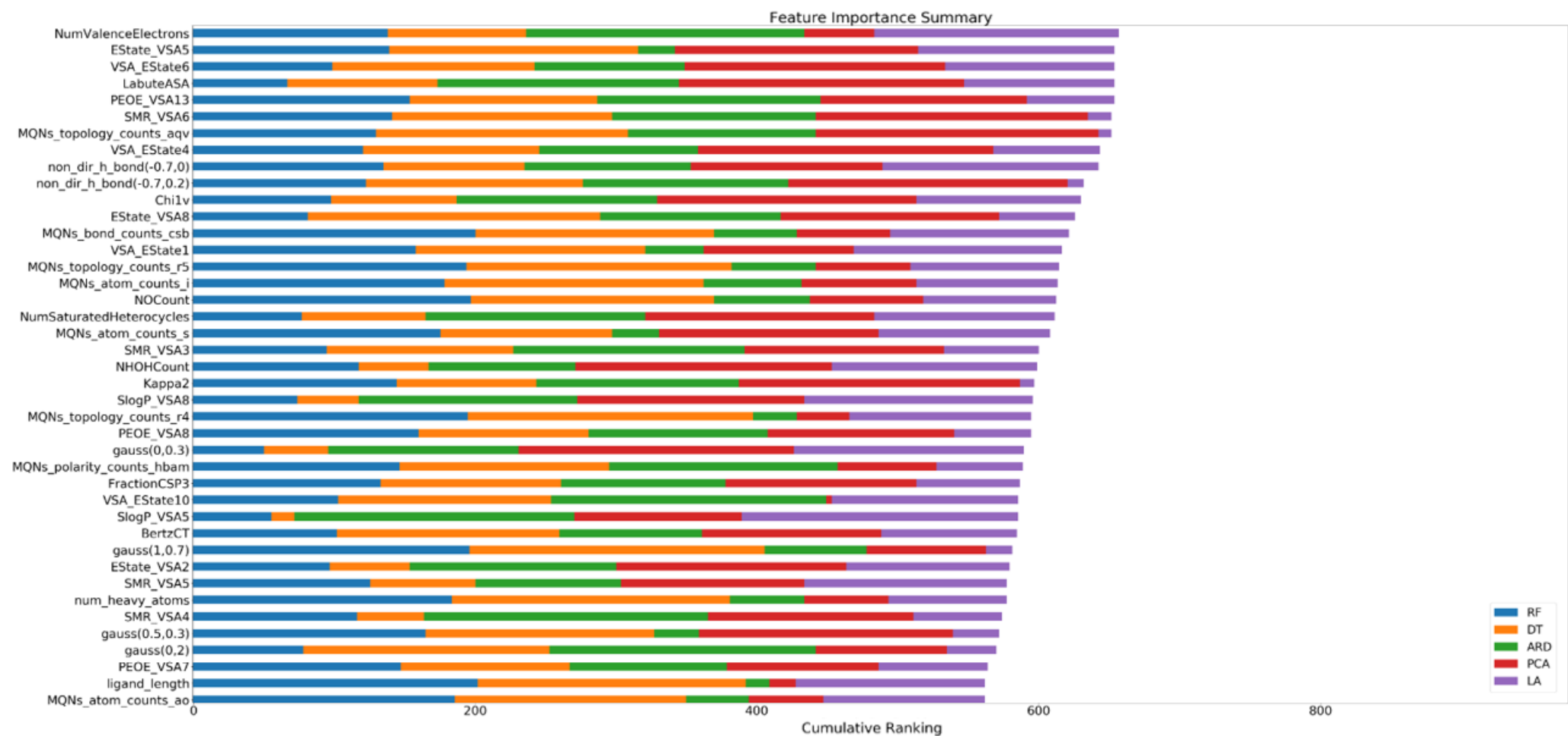


Figura 26: Ranking acumulativo para p_{k_d} (42 a 82)

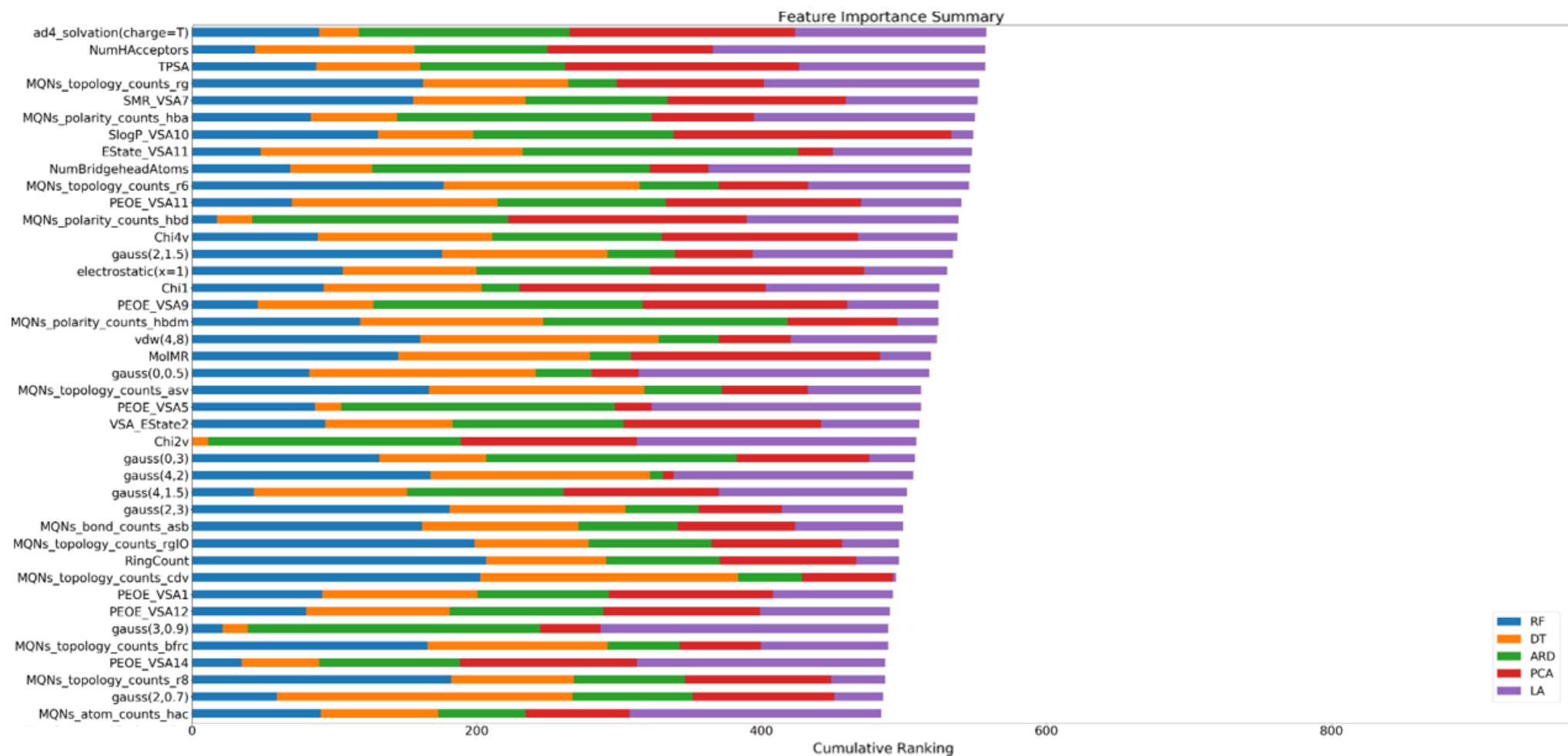


Figura 27: Ranking acumulativo para p_{k_d} (83 a 123)

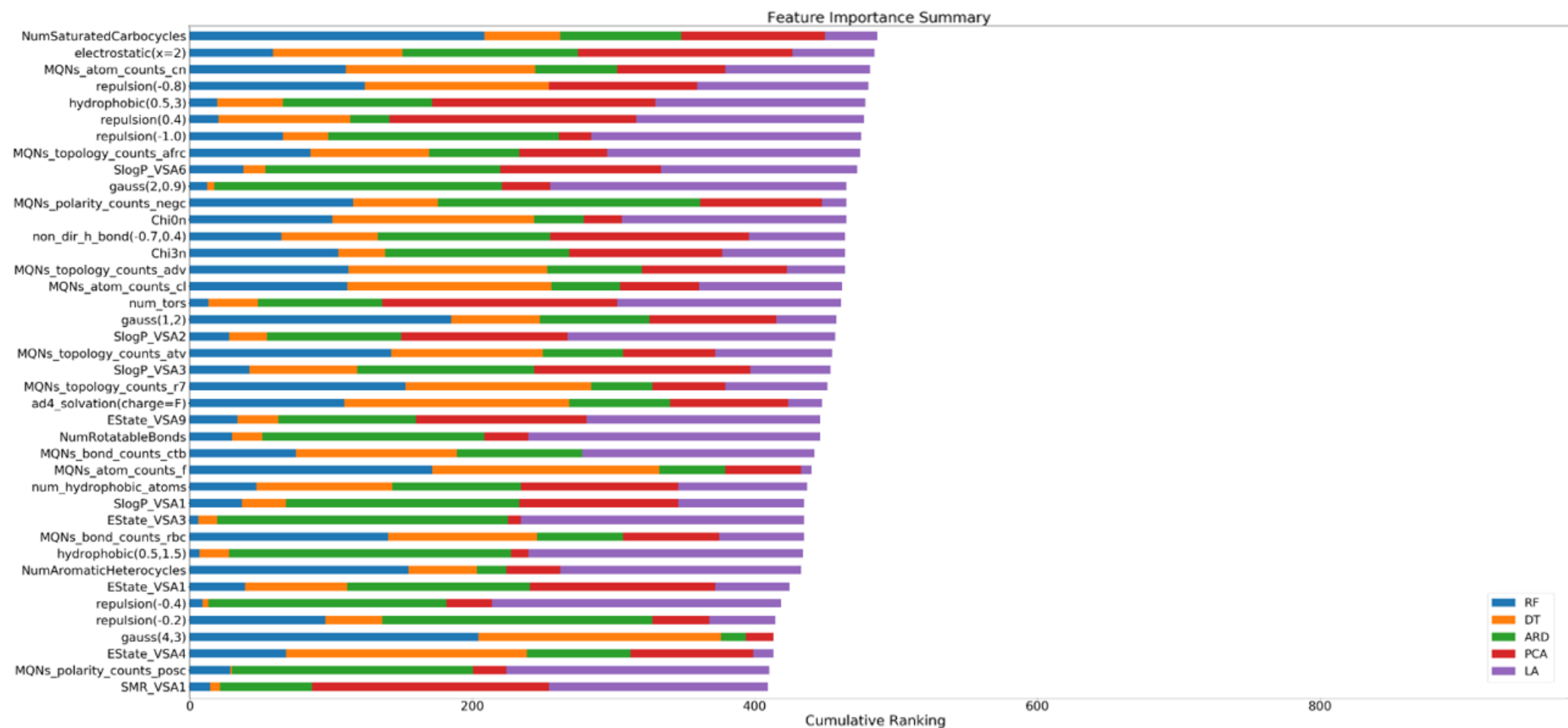


Figura 28: Ranking acumulativo para p_{k_d} (124 a 164)

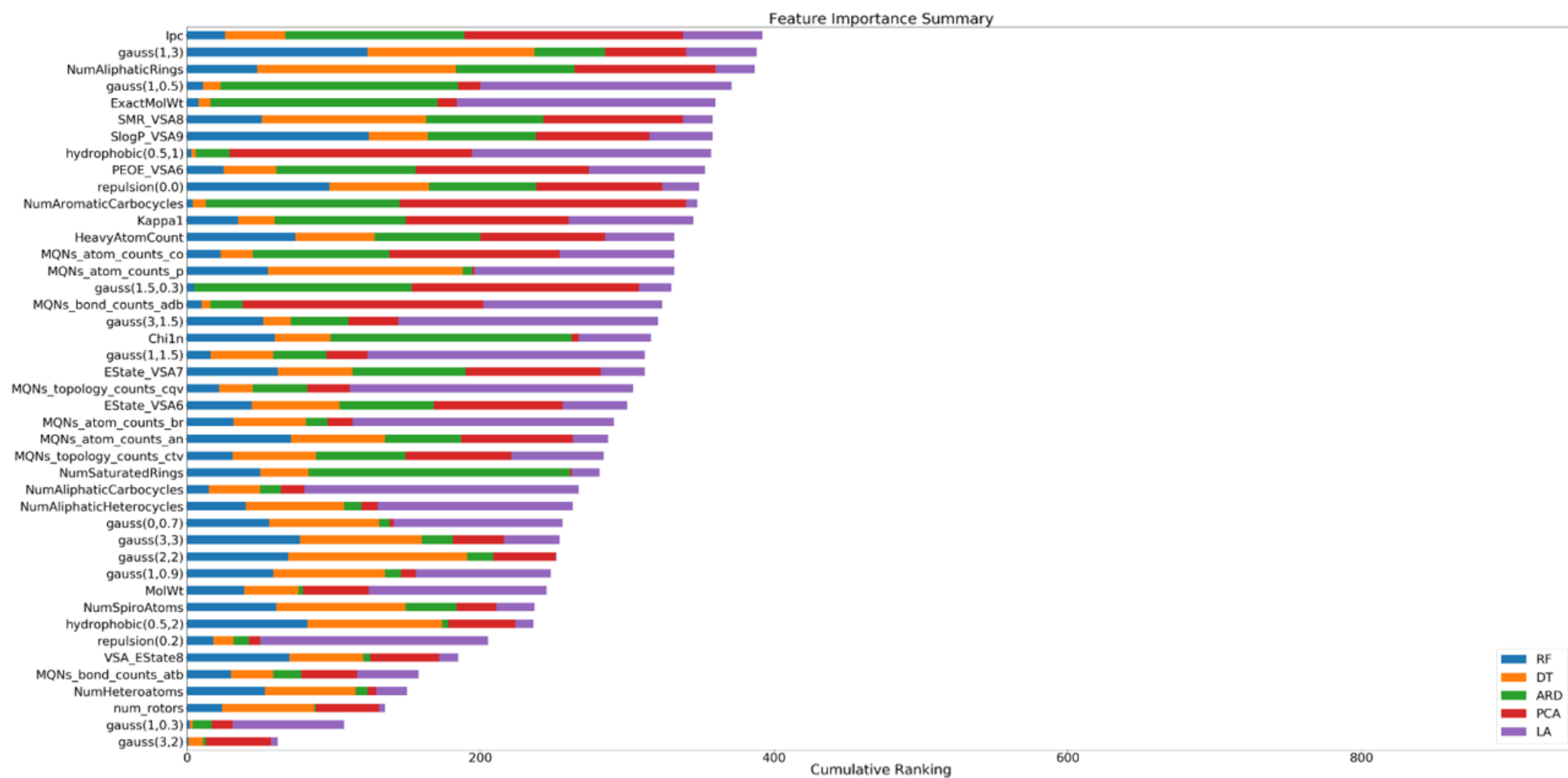


Figura 29: Ranking acumulativo para p_{k_d} (165 a 205)

Os valores apresentados pelos algoritmos de *feature importance* são de correção não de afinidade. Quanto mais altos, melhor é a correlação com atributo alvo, que neste caso são: Δp_{k_d} e p_{k_d} . Então, se Δp_{k_d} é alto, $p_{k_d(\text{train})}$ é bastante diferente de $p_{k_d(\text{Vina})}$, por isso as *features* que contribuem a manter um valor de Δp_{k_d} e p_{k_d} altos são importantes para estimar ambos os fatores de correção.

9.2 Resultados da seleção de atributos utilizando um sistema *Ensemble*

Após as etapas descritas anteriormente, selecionamos quatorze conjuntos diferentes que equivalem aos 5, 10, 15, 20, 25, 30, 40, 50, 70, 80, 90, 100, 105 e 205 atributos mais importantes de acordo com a utilização conjunta dos métodos de aprendizado de máquina no *ranking* acumulativo.

Nesta etapa, para a predição de Δp_{k_d} e p_{k_d} utilizamos os métodos: *Decision Tree*, *Random Forest*, *Lasso*, *Gradient Boosting*, e *Support Vector Machine*. Calculamos o poder de pontuação e o poder de classificação para cada um dos complexos proteína-ligante utilizando cada um dos conjuntos de atributos selecionados.

Com os métodos de ML aplicados, repetimos este experimento dez vezes e um sistema *Ensemble* foi desenvolvido para avaliar o desempenho do cálculo das predições de Δp_{k_d} e p_{k_d} ao utilizar cada um dos conjuntos de atributos selecionados.

Utilizamos a média, a mediana e o *trim rule*. Cada métrica foi analisada individualmente e, finalmente, comparada com o desempenho do método *Random Forest*, uma vez que ambos os fatores de correção utilizados foram treinados por (Wang and Zhang, 2017) com este algoritmo. Neste experimento, todos os atributos foram utilizados.

Assim, em nosso primeiro experimento para o teste do poder de classificação, utilizando-se a média e o fator de correção Δp_{k_d} , o melhor conjunto de atributos é aquele composto por 80 atributos. Isso pode ser observado na Figura 30.

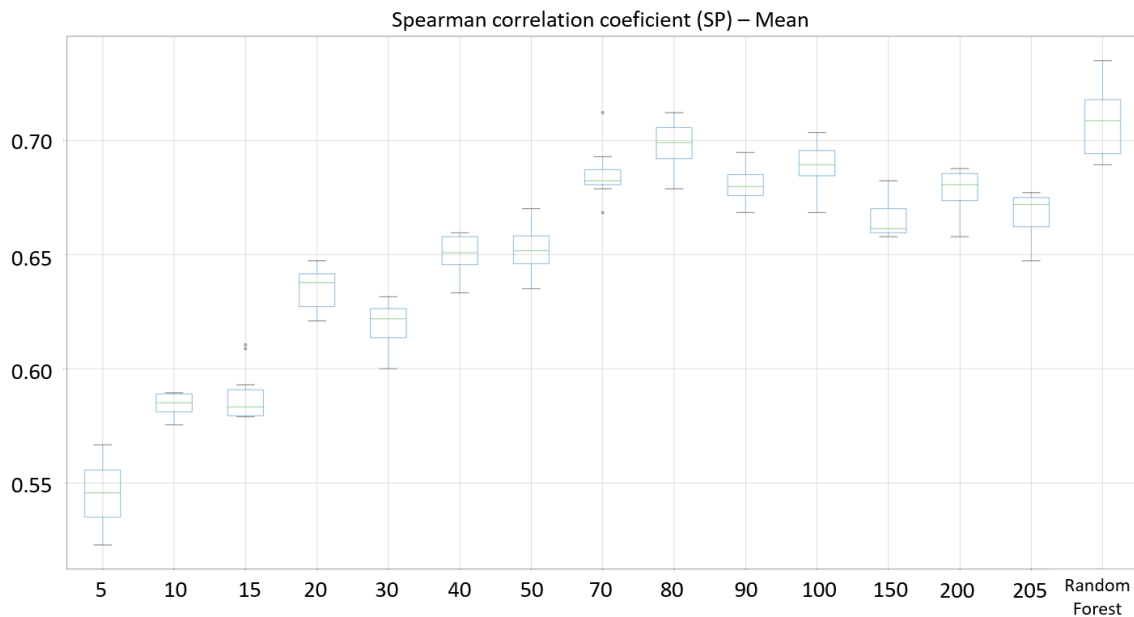


Figura 30: Poder de pontuação avaliado pelo sistema *Ensemble* utilizando a média para Δp_{k_d}

Como apresentado na Figura 31, quando utiliza-se a mediana, o melhor conjunto de atributos é aquele composto por 90 atributos. Neste caso, utilizando-se também, o fator de correção Δp_{k_d} .

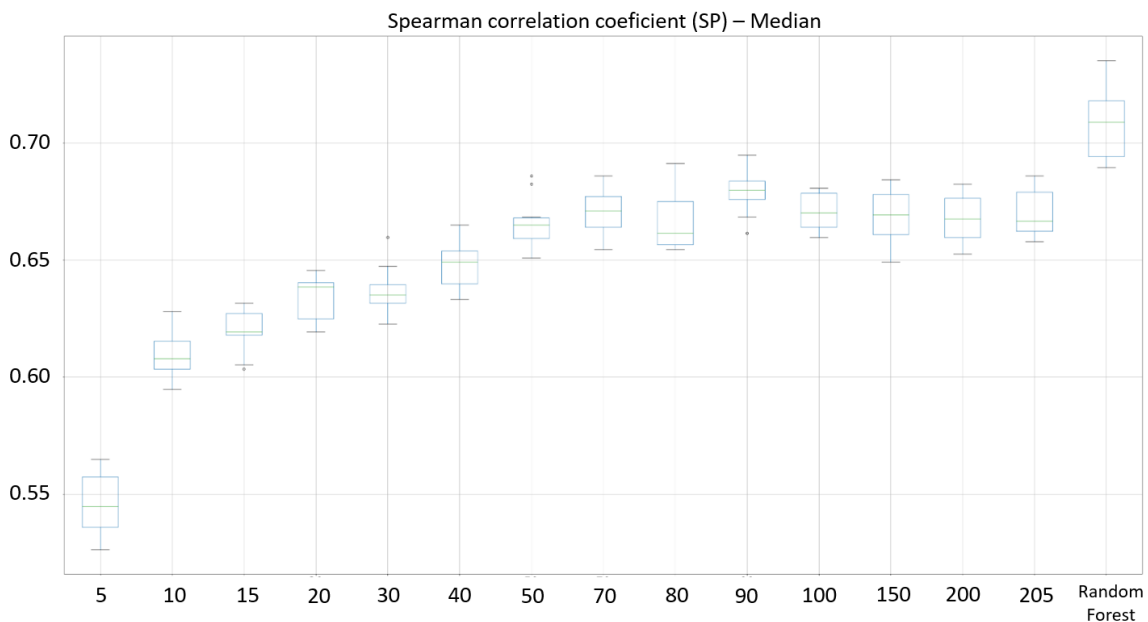


Figura 31: Poder de pontuação avaliado pelo sistema *Ensemble* utilizando a mediana para Δp_{k_d}

Com o *trim rule*, o melhor conjunto de atributo, utilizando Δp_{k_d} , é aquele composto por 90 recursos como mostra a Figura 32.

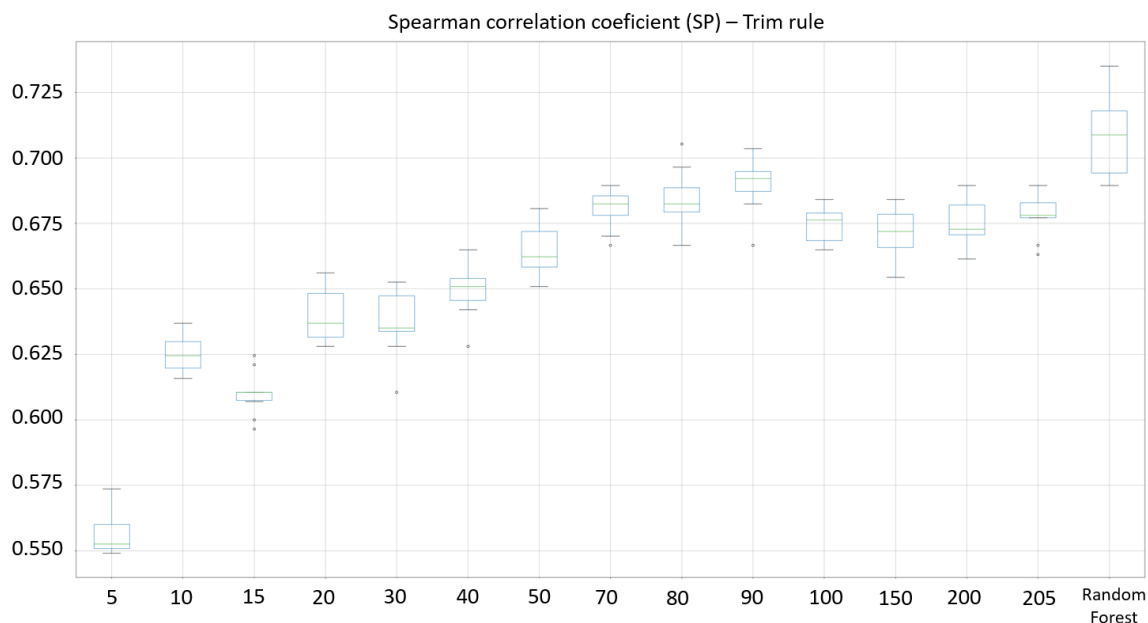


Figura 32: Poder de pontuação avaliado pelo sistema *Ensemble* utilizando o *trim rule* para Δp_{k_d}

Cabe ressaltar que o *trim rule* é uma medida a qual realiza uma média entre todas as predições, após ele ordena cada uma delas e retira os extremos (o maior valor e o menor valor).

Em relação ao poder de classificação, para Δp_{k_d} utilizando-se a média, o melhor conjunto de atributos é aquele composto por 80 atributos, como pode ser observado na Figura 33.



Figura 33: Poder de classificação avaliado pelo sistema *Ensemble* utilizando a média para Δp_{k_d}

Quando utilizamos a mediana para avaliar o poder de classificação, percebemos na

Figura 34 que o melhor conjunto de atributos para este caso foi aquele composto por 70 atributos para Δp_{k_d} .

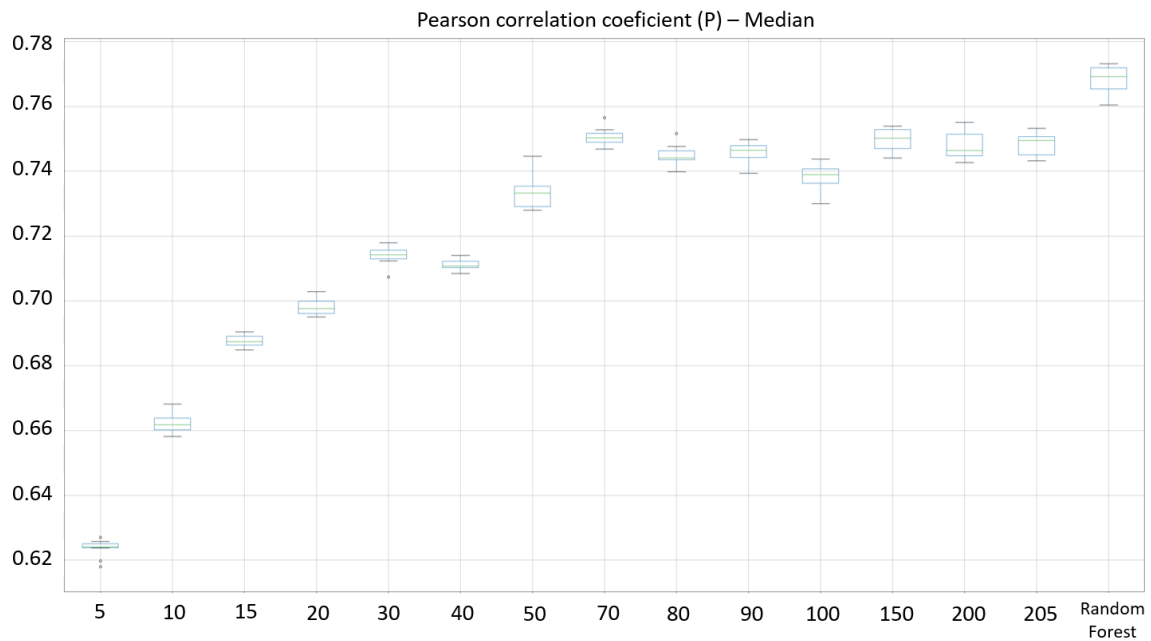


Figura 34: Poder de classificação avaliado pelo sistema *Ensemble* utilizando a mediana para Δp_{k_d}

Avaliando a predição de Δp_{k_d} com o *trim rule*, observamos conforme a Figura 35 que o melhor conjunto de atributos é aquele composto por 70 recursos.

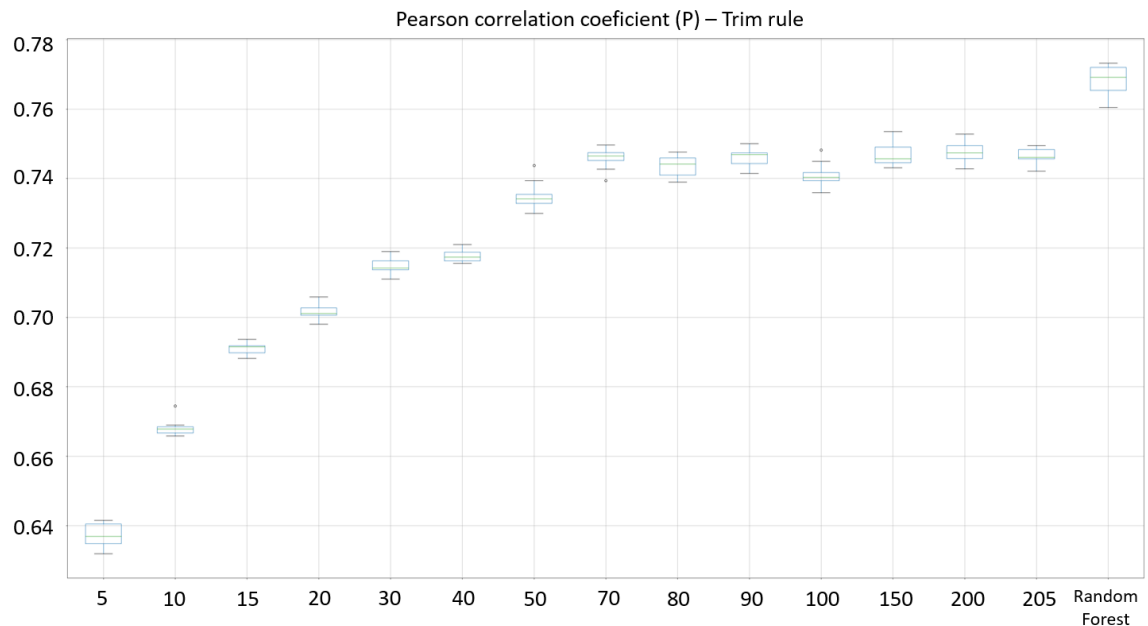


Figura 35: Poder de classificação avaliado pelo sistema *Ensemble* utilizando o *trim rule* para Δp_{k_d}

Para todos estes experimentos, constatamos que o método *Random Forest* quando aplicado separadamente, utilizando todo o conjunto de atributos alcança as melhores pontuações,

tanto para o poder de classificação quanto para o poder de pontuação.

Depois de avaliar as predições para Δp_{k_d} , seguimos a avaliação do fator de correção p_{k_d} . A seguir, na Figura 36 é apresentado os resultados em relação ao poder de classificação utilizando-se a média. Observa-se que, para este caso, utilizar todos os atributos de nosso conjunto apresenta uma melhor pontuação.

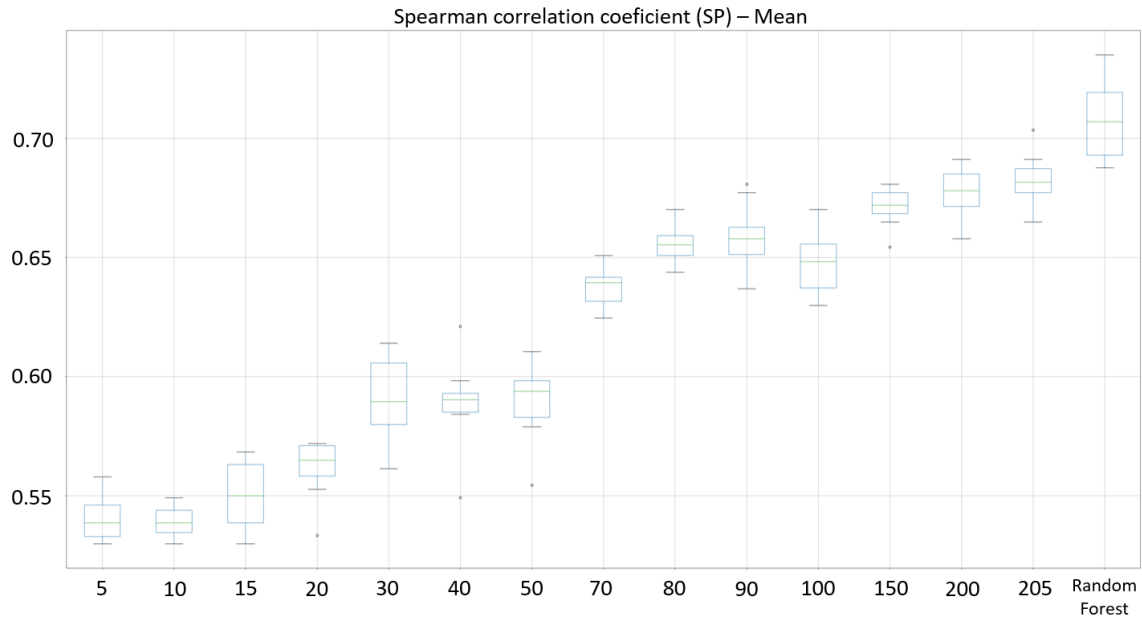


Figura 36: Poder de classificação avaliado pelo sistema *Ensemble* utilizando a média para p_{k_d}

Porém, quando utilizamos a mediana o melhor conjunto de atributos é aquele composto por 150 recursos para p_{k_d} , como pode ser observado na Figura 37.

Contudo, quando utilizamos o *trim rule*, o melhor conjunto de atributos é aquele composto por 200 recursos para p_{k_d} como mostra a Figura 38.

No teste do poder de pontuação, utilizando a média, o melhor conjunto de atributos, para a predição de p_{k_d} , foi aquele composto por 150 recursos, como é apresentado na Figura 39.

No entanto, conforme apresenta a Figura 40, ao utilizar a mediana percebemos que utilizar os 200 melhores atributos contribui para uma melhor predição de p_{k_d} .

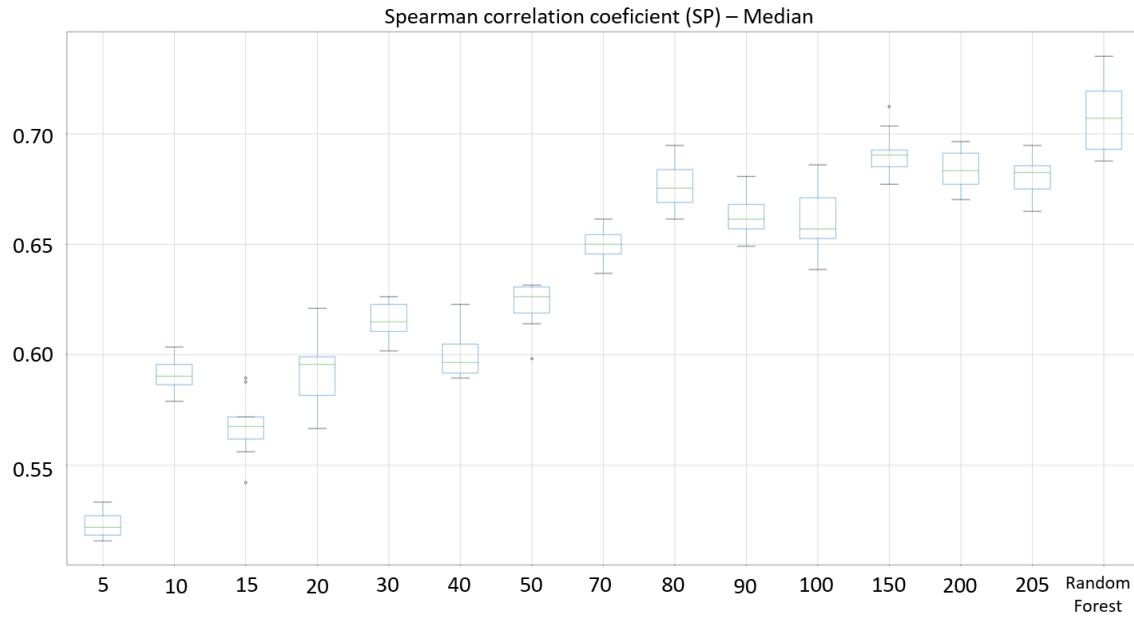


Figura 37: Poder de classificação avaliado pelo sistema *Ensemble* utilizando a mediana para p_{k_d}

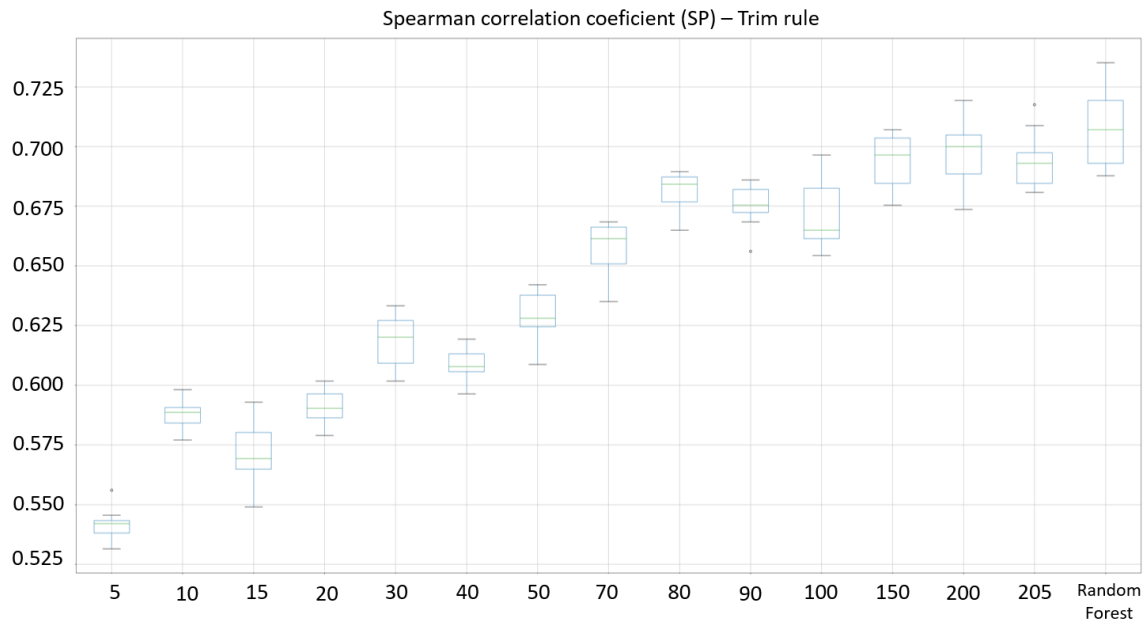


Figura 38: Poder de classificação avaliado pelo sistema *Ensemble* utilizando o *trim rule* para p_{k_d}

Ao utilizar o *trim rule* o melhor conjunto de atributos para a predição de p_{k_d} é aquele composto por 150 atributos, de acordo com a Figura 41.

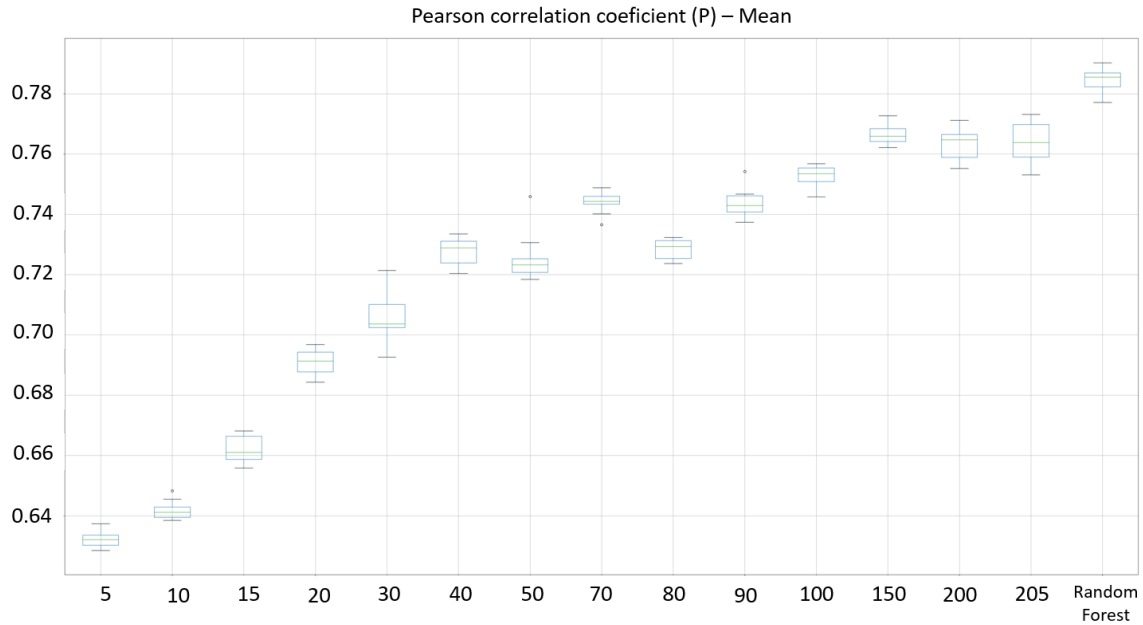


Figura 39: Poder de pontuação avaliado pelo sistema *Ensemble* utilizando a média para p_{k_d}

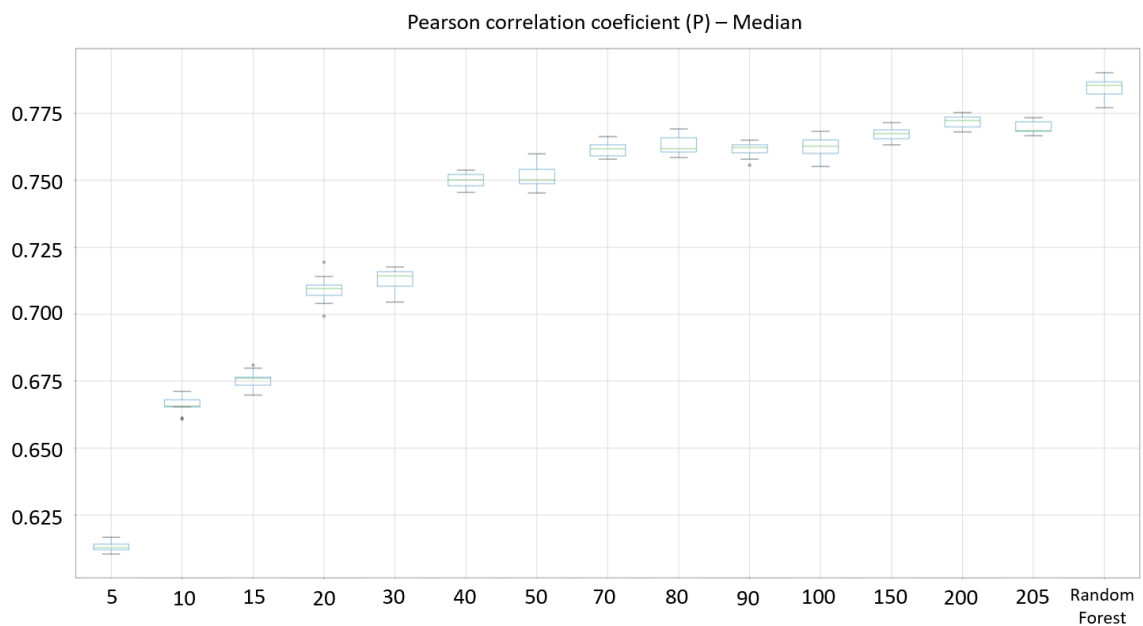


Figura 40: Poder de pontuação avaliado pelo sistema *Ensemble* utilizando a mediana para p_{k_d}

Ao analisar de um modo geral a utilização do *Ensemble*, verificamos que utilizar a métrica de *trim rule* e o conjunto dos nossos 90 melhores atributos é possível obter um maior poder de pontuação para a predição de Δp_{k_d} . Já para p_{k_d} , o maior poder de pontuação foi observado quando utilizamos os nossos 200 melhores atributos e a métrica de *trim rule*.

Um melhor poder de classificação para Δp_{k_d} é obtido quando utilizamos os nossos 70 atributos e a métrica da mediana. Com relação a p_{k_d} foi necessário utilizar os nossos 150

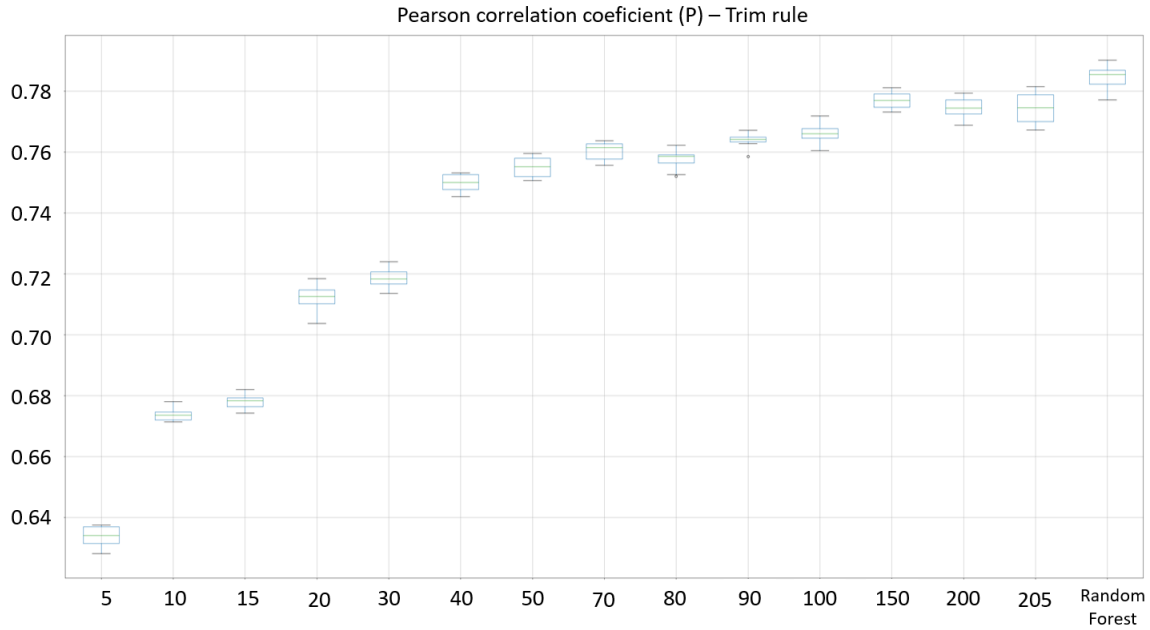


Figura 41: Poder de pontuação avaliado pelo sistema *Ensemble* utilizando o *trim rule* para p_{k_d}

atributos e a métrica do *trim rule* para obtenção do melhor poder de classificação.

Em ambos os gráficos anteriormente apresentados, é possível perceber que o método *Random Forest*, utilizando 205 atributos apresenta os melhores resultados, tanto para o poder de pontuação quanto para o poder de classificação. Isso porque alguns métodos que foram utilizados em nosso sistema *Ensemble* não oferecem um bom desempenho preditivo em relação aos atributos alvo que foram aplicados. A Figura 42 mostra o desempenho dos métodos de *Ensemble* em comparação com cada um dos algoritmos de ML individualmente para Δp_{k_d} .

O gráfico correlaciona o poder de pontuação (P) no eixo y, com o poder de classificação (SP) no eixo x. De forma geral, para os coeficientes P de Pearson e SP de Spearman, valores entre -0,3 e 0,3 são biologicamente desprezíveis; entre 0,31 e 0,5 (ou -0,31 e -0,5) são correlações fracas; entre 0,51 e 0,7 (ou -0,51 e -0,7) são moderadas; entre 0,71 e 0,9 (ou -0,71 e -0,9) são correlações fortes; e $> 0,9$ (ou $< -0,9$) são consideradas muito fortes.

Sendo assim, neste gráfico é possível observar que ambos os algoritmos de ML observados individualmente apresentam correlações moderadas e isto influencia no sistema *Ensemble* ao qual é aplicado estes métodos. Observamos que a única correlação forte entre os atributos utilizados para a predição de Δp_{k_d} é com o método *Random Forest*.

A seguir, na Figura 43 é apresentado uma comparação dos resultados de ambos os métodos utilizados para a predição do fator de correção p_{k_d} . Com este gráfico, pode-se verificar que o método *Support Vector Machine* apresenta uma correlação muito fraca para a predição deste atributo alvo.

Pearson correlation vs. Spearman rank correlation - 205 Features

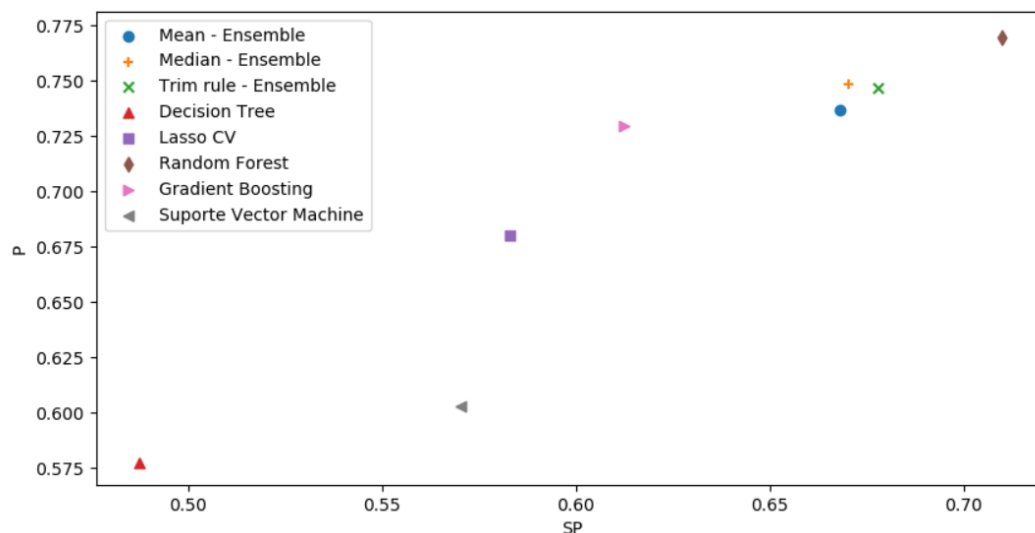


Figura 42: Comparação do *Ensemble* e dos métodos individualmente para a predição de Δp_{k_d}

Pearson correlation vs. Spearman rank correlation - 205 Features

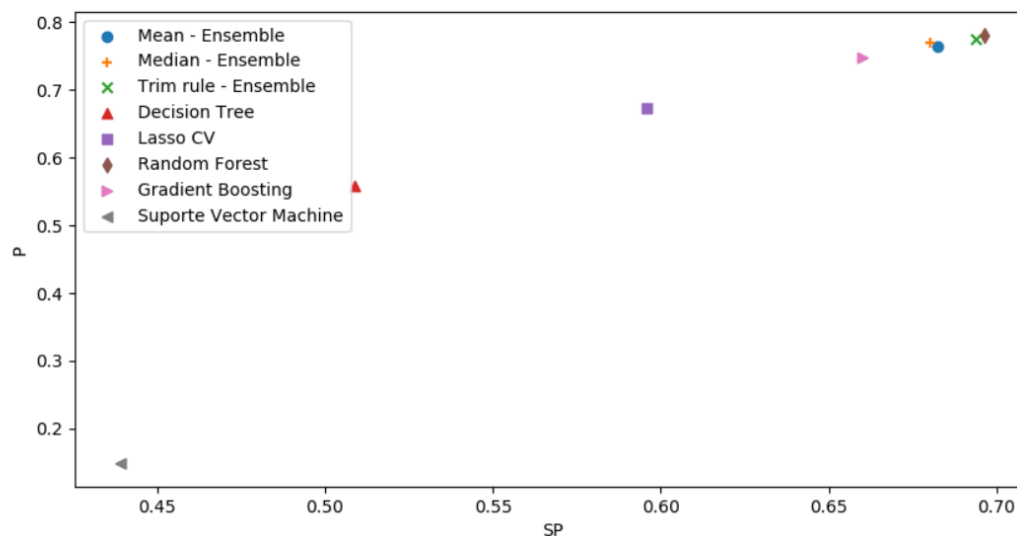


Figura 43: Comparação do *Ensemble* e dos métodos individualmente para a predição de p_{k_d}

A partir da análise destes resultados, percebemos que em nosso sistema *Ensemble* encontram-se métodos que possuem poderes preditivos ruins para a predição de ambos atributos alvos e, ficou evidente que o método *Random Forest* obteve melhores resultados sob os métodos *Ensemble* utilizados. Uma análise mais detalhada da comparação de ambos os métodos individualmente para a predição dos atributos alvo, pode ser encontrada no Anexo A.5.

9.3 Resultados da otimização do modelo proposto

Uma vez que uma otimização pode mudar a importância dos atributos e que conjuntos que tem menos atributos em comparação com o total de recursos utilizados pode gerar um modelo que é diferente (porque os dados são diferentes - tem menos atributos), quando estes conjuntos variam em relação a quantidade de dados, para cada conjunto ($k = 5, 10, \dots$) fizemos uma otimização. Além disso, também foi realizada uma otimização em relação aos parâmetros de cada um dos métodos utilizados para a predição da afinidade de ligação no processo de *Ensemble*.

Em mais detalhes, começamos realizando um ajuste automático de parâmetros para cada método. Aplicamos o método GridSearchCV com um conjunto de parâmetros. O método foi escolhido por garantir a confiança de que todas as combinações de parâmetros sejam testadas, de acordo com Gao and Sun (2010), ao qual afirma além disso que a segurança no resultado a ser obtido também é mais confiável, quando comparado aos outros métodos para otimização de parâmetros.

Utilizamos um gerador de validação cruzada e dividimos o conjunto de dados em k dobras a partir de kfold. Com a validação cruzada, o conjunto de dados é dividido em n divisões. Este processo envolve quebrar em n partes ou k partes e rodar k processos de treino e teste, validando os algoritmos que serão utilizados. Em nosso modelo, utilizamos kfold com $n_splits=3$ (número de dobras), que representa a quantidade na qual a base de dados foi dividida, para uma validação cruzada em 3 vezes. *KFoldCrossValidation* e *GridSearchCV* pertencem a biblioteca scikit-learn do Python.

Um dos principais objetivos da validação cruzada é estimar a capacidade de um modelo de aprendizagem de máquina, quando os dados não são aqueles utilizados durante o treinamento do modelo. O GridSearchCV foi usado para automatizar o processo de ajuste de parâmetros, criando combinações destes parâmetros e posteriormente avaliando aquela que traz os melhores resultados, aplicamos esta técnica para cada uma das etapas em nossa metodologia.

Na etapa da realização de *feature importance*, utilizamos três dos cinco métodos utilizados para ranquear a medida de importância dos atributos. Sendo estes, o *Decision Tree* e o *Random Forest* e o Lasso. ARD e PCA decidimos não otimizar, porque o método ARD fez com que o tempo para execução de nosso algoritmo fosse muito alto e o método PCA apresentou predições com valores muito altos, para Δp_{k_d} e p_{k_d} , quando estes valores entram no *Ensemble* os valores de p_{k_d} ficam muito ruins. Além disso, para prever ambos os atributos alvo é preciso uma regressão linear, o método PCA tem uma função linear - *Linear Regression()* que faz isso, porém o método Lasso já realiza este mesmo processo. O PCA também faz uma transformação nos dados, e já temos nosso conjunto de atributos sem transformação que é usado como entrada do modelo.

Após a seleção dos métodos para a otimização dos parâmetros, na etapa de *feature*

importance, verificamos quais as melhores configurações, para cada método. Detalhamos na sequência, cada uma das configurações de hiperparâmetros que foram utilizadas, em cada método. Estas configurações, foram escolhidas conforme a documentação do Scikit-learn em Pedregosa et al. (2011), a qual especifica os valores iniciais (mínimos) e os limites (valores máximos) de cada hiperparâmetro.

Para o método *Decision Tree*, foi utilizada a configuração de hiperparâmetros conforme é descrita na Tabela 4.

Tabela 4: Configuração para escolha dos parâmetros do método *Decision Tree* para seleção de atributos

Método	GridSearchCV	max_depth	min_samples_split	max_leaf_nodes
<i>Decision Tree</i>	cv = kfold verbose=1 n_jobs=-1	Index = [1,5,10,15]	Index = [2,5,10,15]	Index = [None]

Os parâmetros apresentados na Tabela 4, representam uma peculiaridade específica para o treinamento do modelo. O parâmetro *max_depth* indica a profundidade máxima da árvore. Ele é utilizado para controlar o sobreajuste, uma maior profundidade permitirá ao modelo aprender relações cada vez mais específicas da amostra.

O parâmetro *min_samples_split* avaliará o número de amostras no nó, e se o número for menor que o mínimo, a divisão será evitada e o nó será uma folha. Especificar o número mínimo de amostras para divisão do nó previne a criação de modelos complexos e evita *underfitting*.

Além deste parâmetro, também utilizamos *max_leaf_nodes* para limitar o número de nós folha. Esse parâmetro faz a árvore crescer da melhor maneira até que *max_leaf_nodes* seja alcançado. A melhor divisão é decidida com base na redução de impurezas. Se nenhum, então o algoritmo escolherá um número ilimitado de nós folha, com base no modelo.

Com relação ao método *Random Forest*, a configuração mostrada na Tabela 5 foi utilizada para escolha de seus parâmetros.

Tabela 5: Configuração para escolha dos parâmetros do método *Random Forest* para seleção de atributos

Método	GridSearchCV	n_estimators	max_features	min_samples_leaf
<i>Random Forest</i>	cv=kfold verbose=1 n_jobs=-1	Index = [25,50,75,100]	Index = [0.2, 0.7, 0.5, 1.0]	Index = [1, 2, 5, 10]

Assim como no algoritmo *Decision Tree*, cada parâmetro também representa sua peculiaridade para o treinamento do modelo utilizando o método *Random Forest*. Para este método, *n_estimators* representa o número de árvores ao qual pretendemos construir antes de obter a votação máxima ou as médias das previsões.

O parâmetro `max_features`, determinará o tamanho dos subconjuntos aleatórios de recursos a serem considerados ao dividir cada nó. Em nosso modelo, `max_features` é uma fração e os recursos são considerados em cada divisão.

O parâmetro `min_samples_leaf` representa o número de recursos considerados para divisão em cada nó folha. O valor escolhido irá determinar quantas amostras cada folha terá para classificar.

Como resultado, obtivemos nesta etapa: `max_depth: 5`, `max_leaf_nodes: None`, `min_samples_split: 5` para o algoritmo *Decision Tree*. E para o algoritmo *Random Forest*, `max_features: 0.2`, `min_samples_leaf: 1`, `n_estimators: 100`.

Após descrevermos sobre a importância de cada um dos parâmetros e sobre os passos da etapa para a realização do *feature importance*. Damos sequência ao próximo passo, em relação a otimização de parâmetros, a qual fizemos uma combinação das medidas de importância calculadas anteriormente em um *ranking* acumulativo e também aplicamos uma otimização de parâmetros para cada método.

Na Tabela 6 é possível verificar a parametrização utilizada para escolha dos melhores parâmetros ao utilizar métodos: *Decision Tree*, *Random Forest*, *Gradient Boosting Regression* e *Support Vector Machine* para ranquear o conjunto de atributos utilizados.

Tabela 6: Configuração para escolha dos parâmetros no *ranking* acumulativo

Método	GridSearchVC	Parâmetros
<i>Decision Tree</i>	<code>cv = kfold</code> <code>verbose = 1</code> <code>n_jobs = -1</code>	<code>max_depth = [1, 5, 10, 15];</code> <code>min_samples_split = [2, 5, 10, 15];</code> <code>max_leaf_nodes = [None]</code>
<i>Random Forest</i>	<code>cv = kfold</code> <code>verbose = 1</code> <code>n_jobs = -1</code>	<code>n_estimators = [25, 50, 75, 100];</code> <code>max_features = [0.2, 0.7, 0.5, 1.0];</code> <code>min_samples_leaf = [1, 2, 5, 10]</code>
<i>Gradient Boosting Regression</i>	<code>cv = kfold</code> <code>verbose = 1</code> <code>n_jobs = -1</code>	<code>max_features = [0.2, 0.7, 0.5, 1.0];</code> <code>min_samples_leaf = [1, 2, 5, 10];</code> <code>min_samples_split = [2, 4, 6, 8]</code>
<i>Support Vector Machine</i>	<code>cv = kfold</code> <code>verbose = 1</code> <code>n_jobs = -1</code>	<code>gamma = [1e-4, 1e-3, 0.01, 0.1, 0.2, 0.5, 0.6, 0.9];</code> <code>C = [1e-2, 1e-1, 1e0, 1e1, 1e2]</code>

Nos métodos *Decision Tree* e *Random Forest* aplicamos as mesmas configurações apresentadas anteriormente. Para o método *Gradient Boosting Regression* utilizamos `max_features`, ao qual representa uma fração em relação ao conjunto total de atributos, para a construção de cada árvore, uma parcela aleatória de recursos é escolhida pelo algoritmo em cada divisão.

Também utilizamos o parâmetro `min_samples_leaf` que define as amostras mínimas (ou observações) necessárias em um nó terminal ou folha e o parâmetro `min_samples_split` ao qual representará o número mínimo de amostras (ou observações) que serão necessárias em um nó para cada divisão.

Em relação ao método *Support Vector Machine*, sabemos que este algoritmo separa os

pontos de dados que pertencem a diferentes classes com um limite de decisão. Sendo assim, o parâmetro C utilizado na configuração de nosso modelo, determina o equilíbrio entre permitir erros de treinamento e forçar margens rígidas para permitir flexibilidade na separação das classes. E o parâmetro gama auxilia a ajustar os pontos dentro das classes de contorno e a delimitar a maior quantidade de dados possível dentro delas, evitando sobreajuste.

Após a etapa do *ranking acumulativo*, selecionamos os melhores conjuntos e treinamos ambos os algoritmos para cada grupo de atributos selecionados. A configuração final de cada um dos parâmetros selecionados para esta etapa pode ser visualizado no Anexo A.8, onde foi mapeado nossos resultados, dos melhores parâmetros para cada método, em relação a cada grupo de atributos. Em seguida, para cada um destes conjuntos de atributos foi aplicado o *Ensemble*, nesta etapa realizamos a predição de Δp_{k_d} e p_{k_d} utilizando para cada método os parâmetros que foram pré-selecionados anteriormente. Realizamos 10 repetições deste experimento para cada um dos conjuntos de atributos utilizados. Calculamos a média, a mediana e *trim rule*.

Para fins de comparação, todos os métodos anteriormente utilizados, sem otimização de parâmetros, foram treinados mantendo a configuração padrão de parâmetros conforme a documentação do Scikit-learn de acordo com Pedregosa et al. (2011) indicados em nota: *Decision Tree*¹, *Random Forest*², *Lasso*³, *ARD*⁴, *PCA*⁵, *Gradient Boosting*⁶ e *Support Vector Machine*⁷.

Analisando nossos resultados com a otimização de parâmetros, as medidas de importância mudaram em relação aos dois atributos alvos quando comparamos ao modelo sem parametrização. Isto pode ser melhor observado através dos gráficos que são apresentados no Anexo A.5. Nas figuras 55 a 61 são apresentadas as medidas de importância para o atributo alvo p_{k_d} , valores muito próximos a zero não foram incluídos nos gráficos para os métodos *Lasso CV* e *Decision Tree*.

Contudo, o *Random Forest* classificou vários outros parâmetros com importâncias maiores. O método classificou parcialmente, quase todo o conjunto de parâmetros com importâncias bem significativas e por este motivo, incluímos todos seus valores. Esse mesmo comportamento também foi observado para o atributo alvo Δp_{k_d} , apresentado nas Figuras 62 a 67 do Anexo A.5.

A representação combinada dos valores de importância para o conjunto de atributos utilizado, representada graficamente através das Figuras 44 a 48 para p_{k_d} sofreu alteração em seu ranqueamento quando comparada ao experimento anteriormente realizado na

¹<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>

²<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

³https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html

⁴https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ARDRRegression.html

⁵<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

⁶<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>

⁷<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>

seção 7.1.

Nas Figuras 49 a 53 são apresentados os resultados para Δp_{k_d} também para o *ranking acumulativo* das medidas de importância. Observou-se da mesma forma que, a classificação também se altera quando comparamos a classificação anterior, da seção 7.1 apresentada anteriormente.

Com estes resultados iniciais, para o *ranking acumulativo*, observamos que o método que foi mais sensível a otimização de parâmetros foi o método *Random Forest*, para ambos os atributos alvo p_{k_d} e Δp_{k_d} .

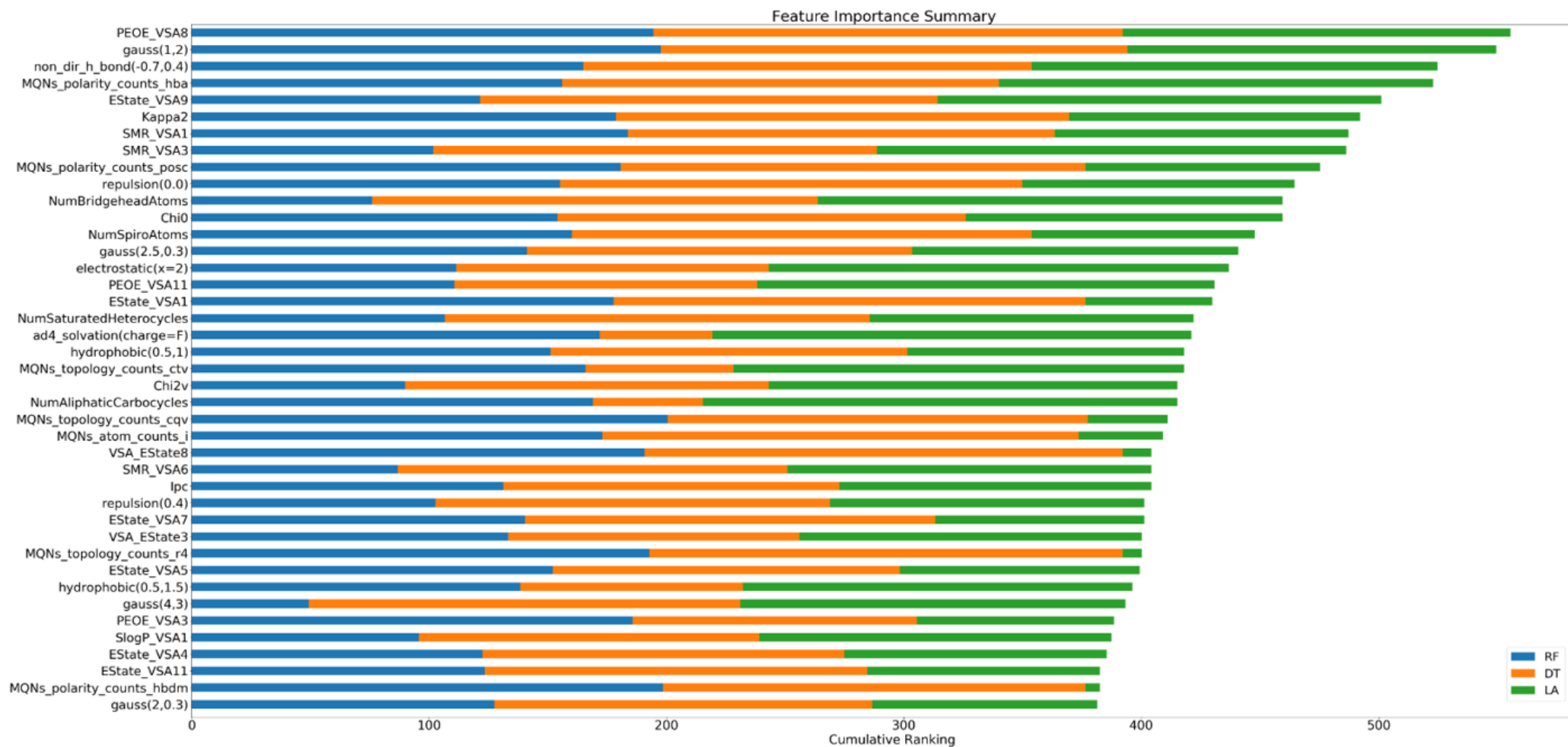


Figura 44: Ranking acumulativo para p_{k_d} (1 a 41)

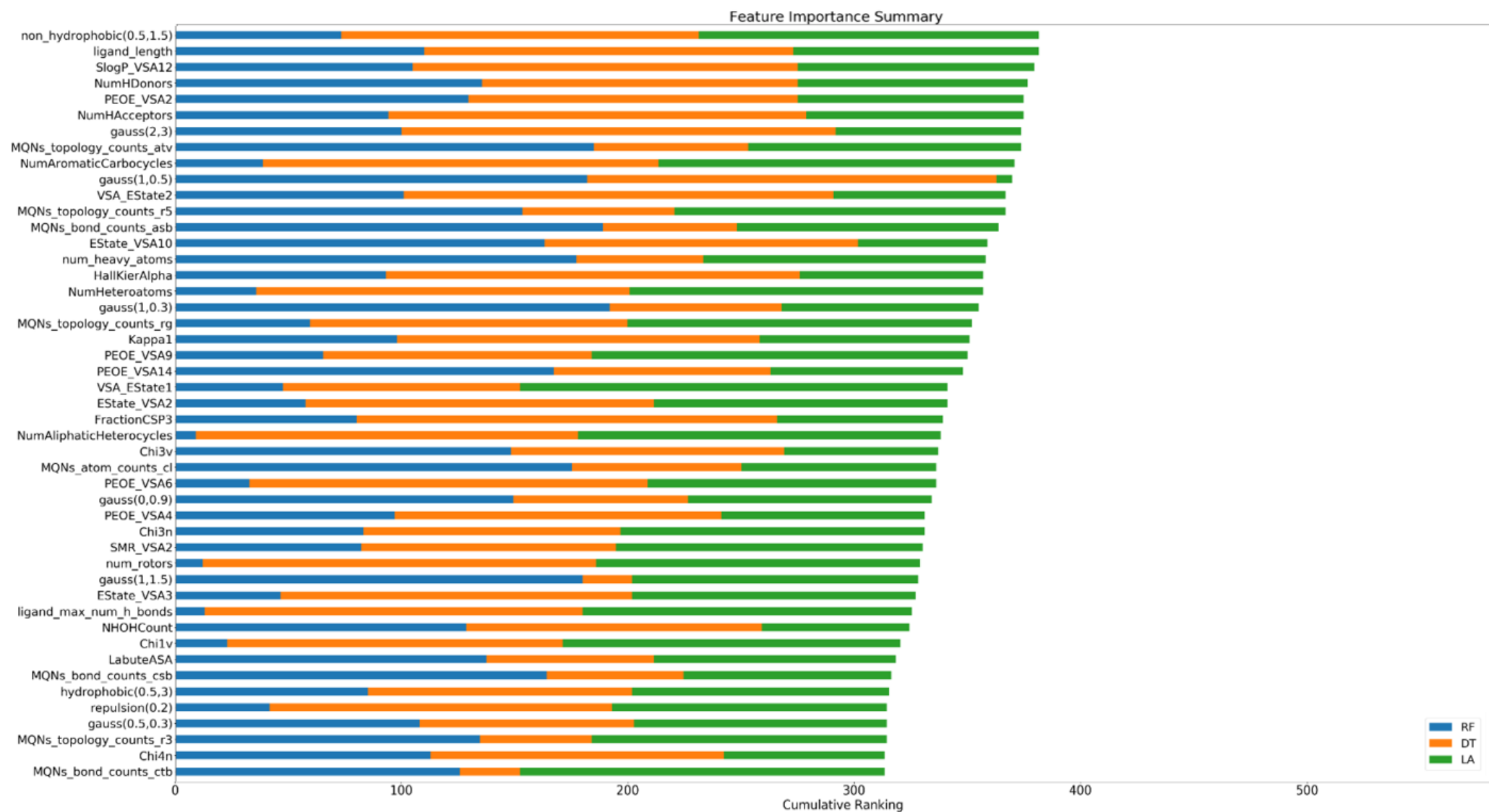


Figura 45: Ranking acumulativo para p_{k_d} (42 a 82)

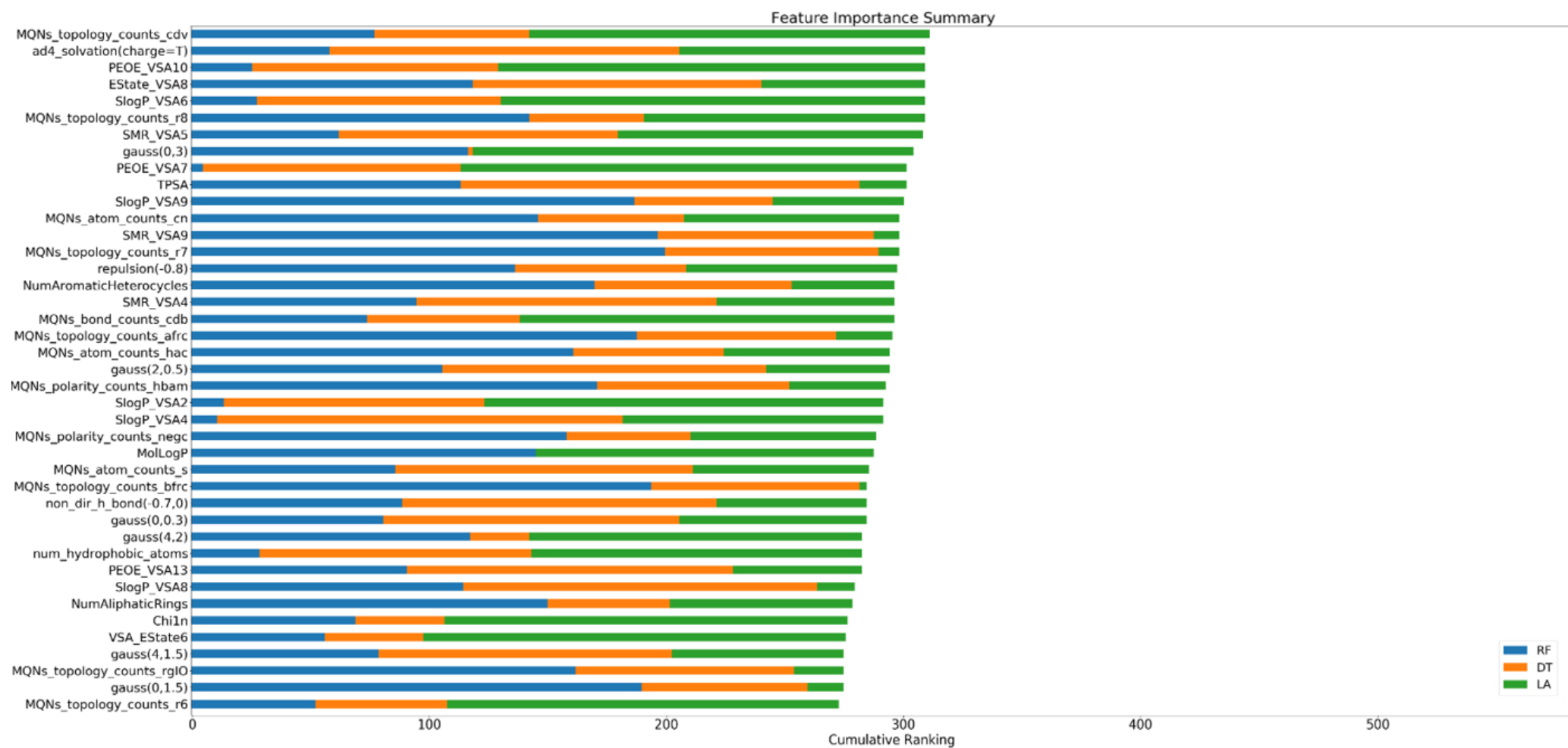


Figura 46: Ranking acumulativo para p_{k_d} (83 a 123)

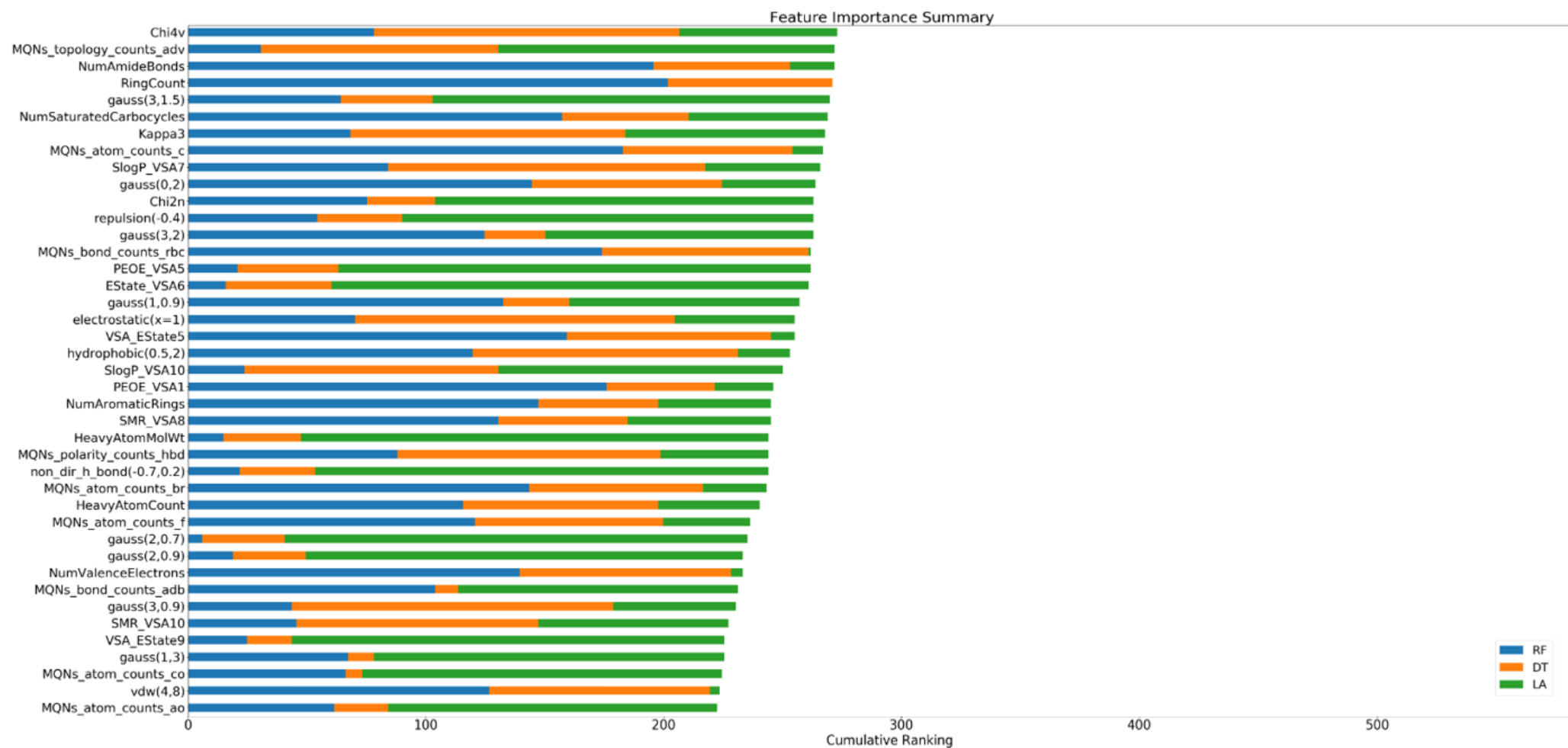


Figura 47: Ranking acumulativo para p_{k_d} (124 a 164)

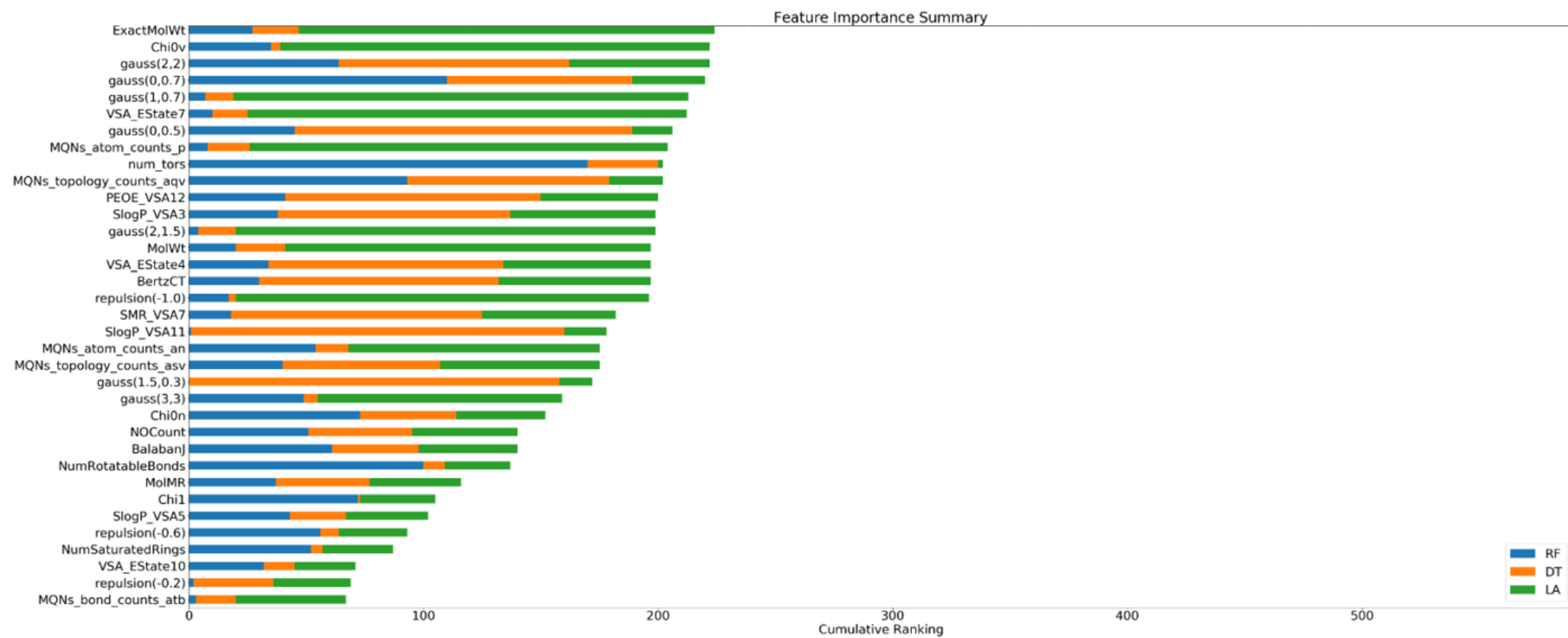


Figura 48: Ranking acumulativo para p_{k_d} (165 a 205)

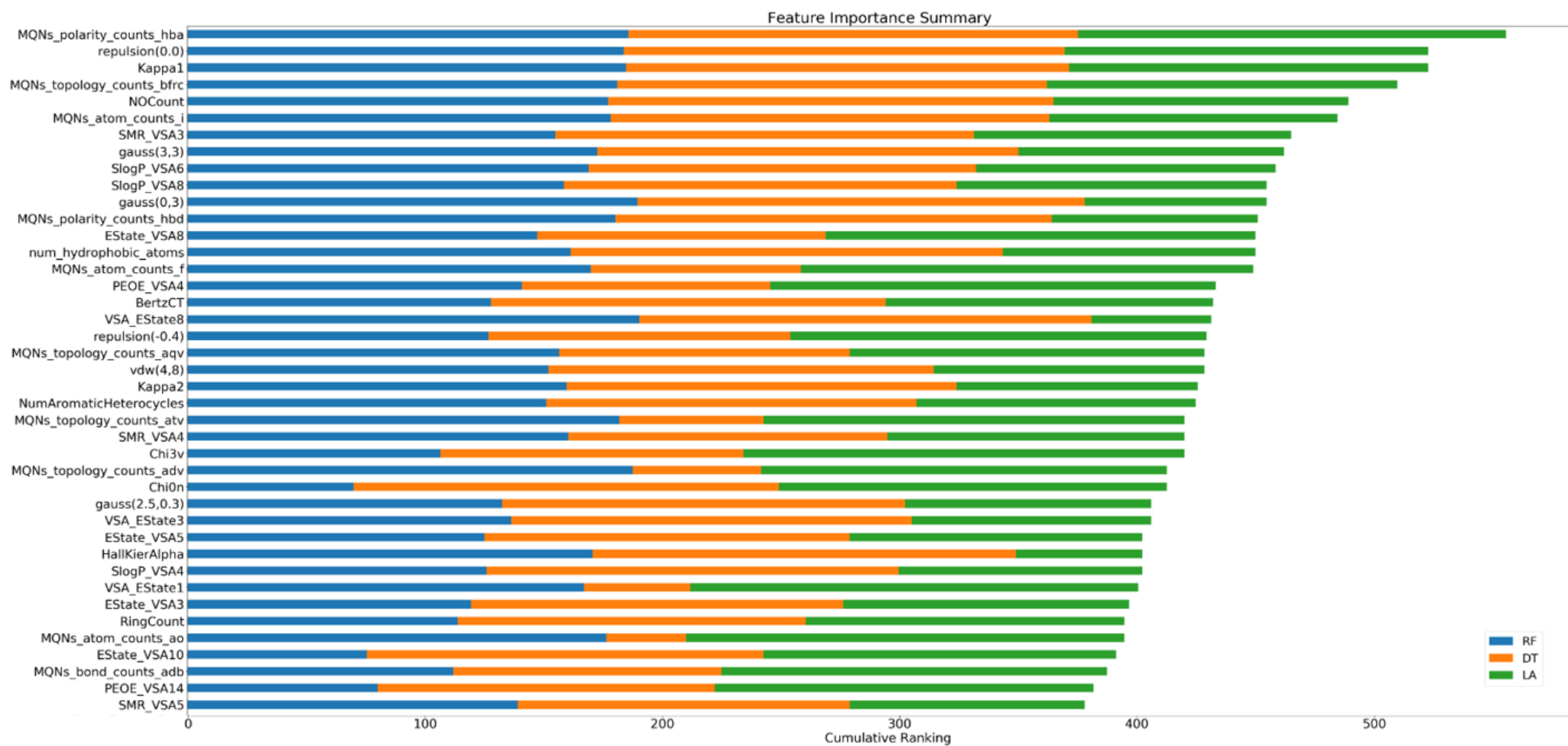


Figura 49: Ranking acumulativo para Δp_{k_d} (1 a 41)

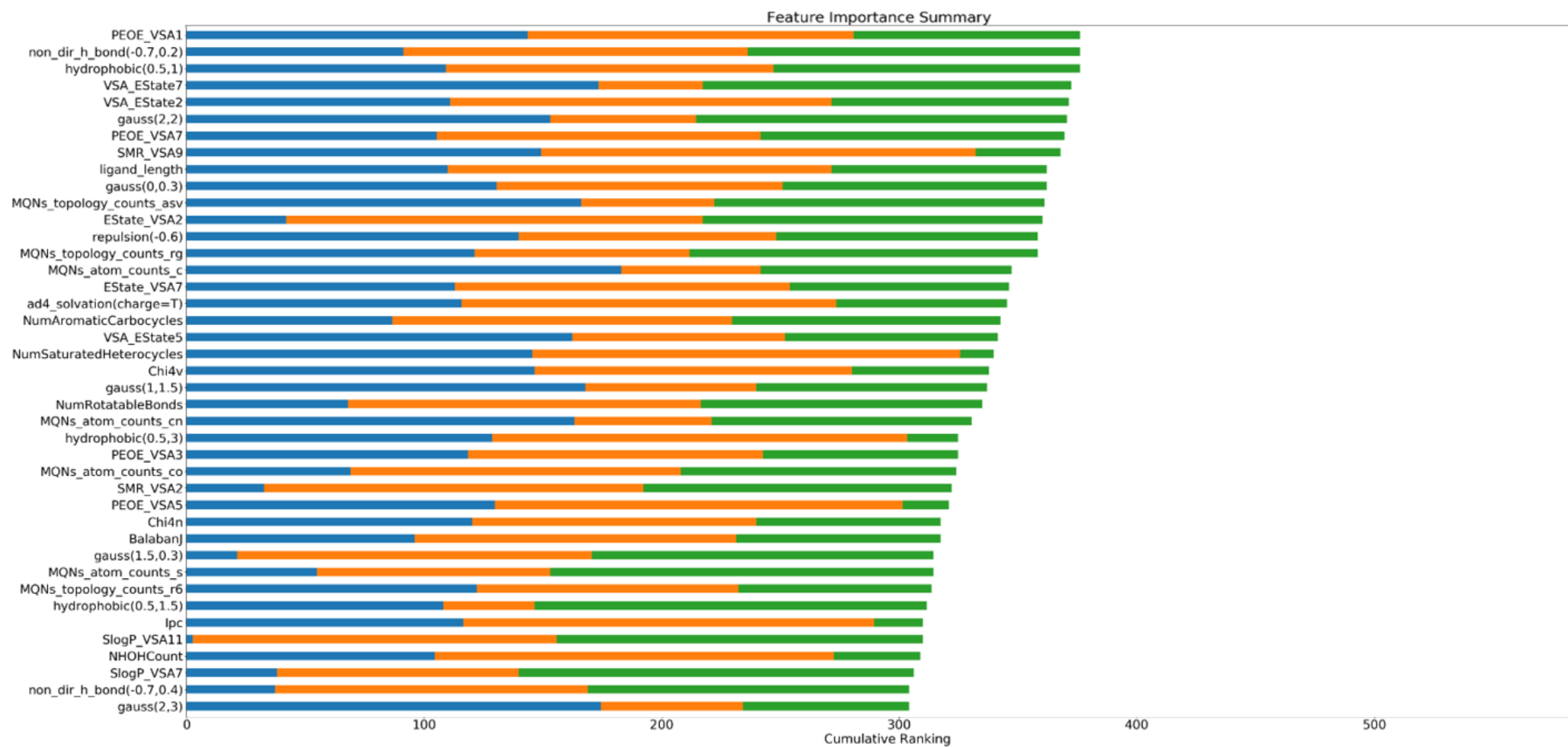


Figura 50: Ranking acumulativo para Δp_{k_d} (42 a 81)

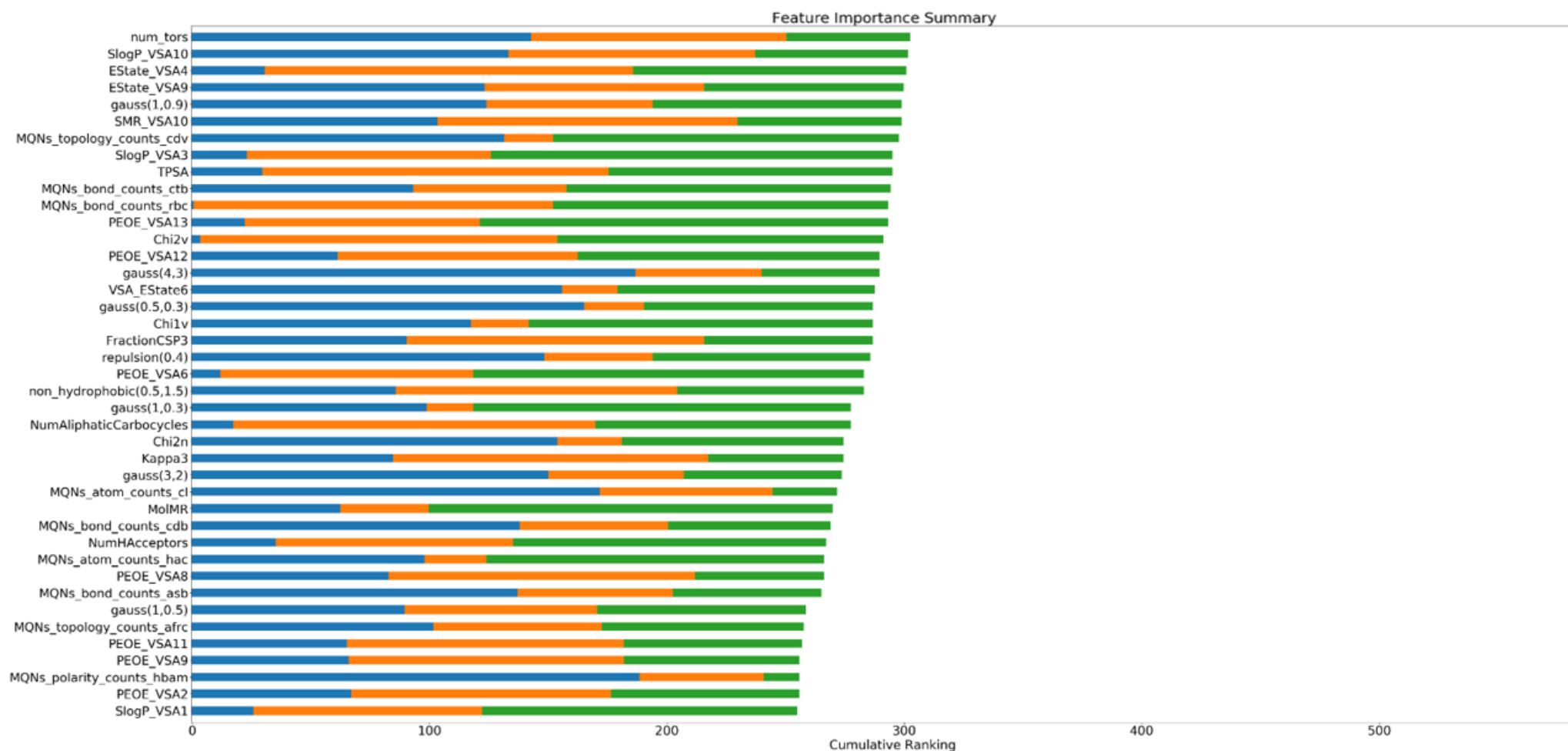


Figura 51: Ranking acumulativo para Δp_{k_d} (82 a 123)

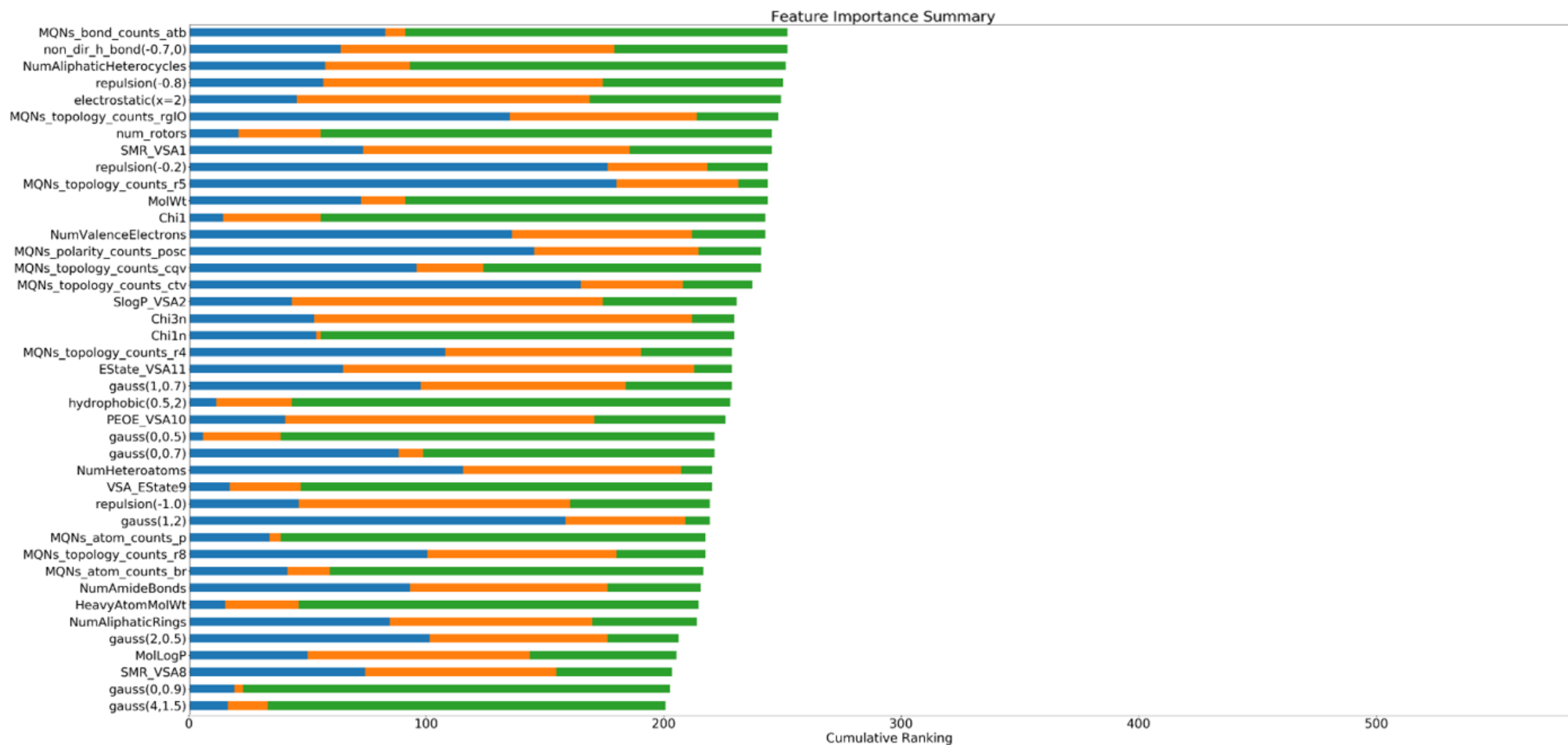


Figura 52: Ranking acumulativo para Δp_{k_d} (124 a 163)

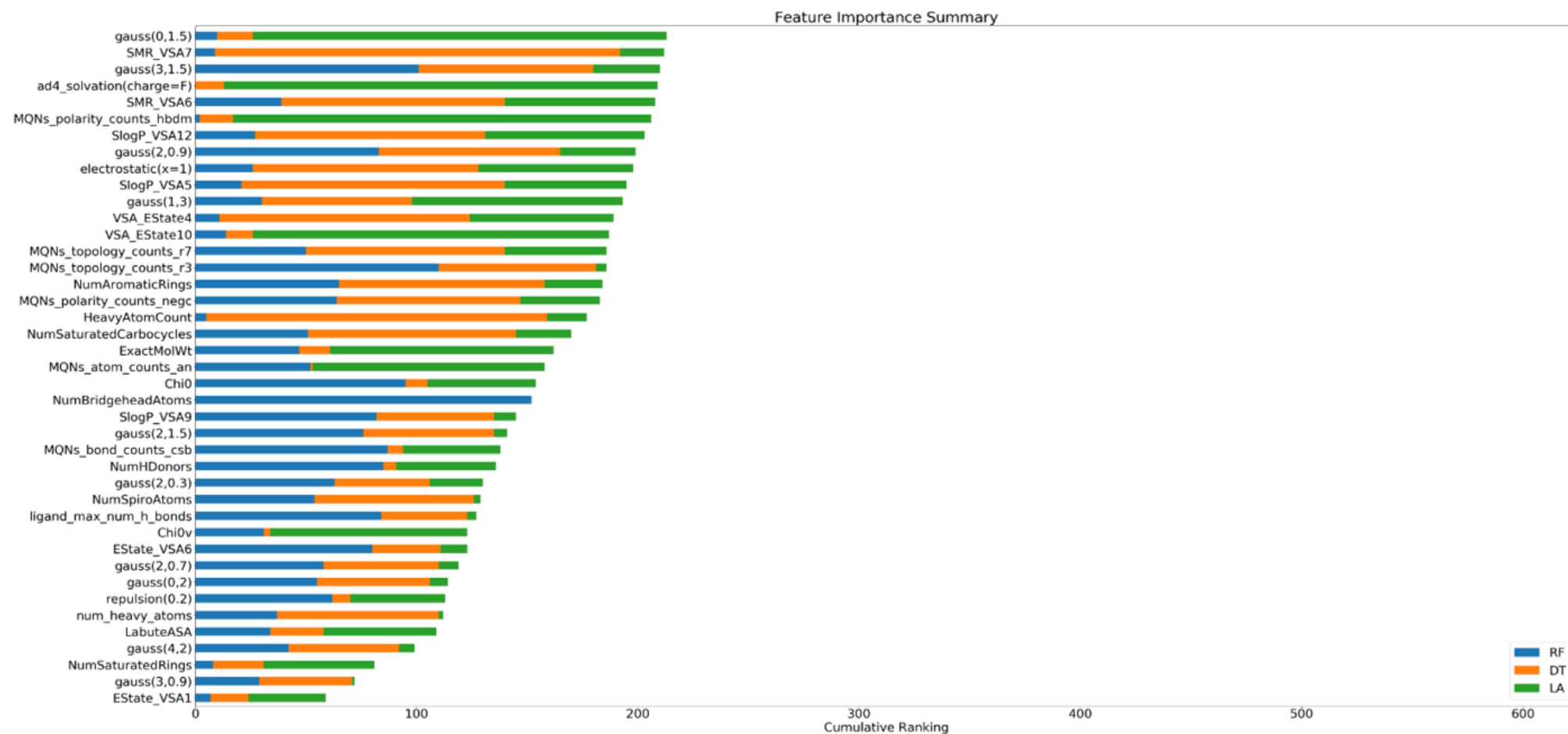


Figura 53: Ranking acumulativo para Δp_{k_d} (164 a 205)

Após combinarmos nossos resultados em um *ranking acumulativo*, fizemos a predição da afinidade de ligação utilizando além do *Random Forest*, *Decision Tree* e *Lasso* os métodos: *Gradient Boosting* e *Support Vector Machine* e fizemos a predição da afinidade de ligação utilizando os atributos alvos p_{k_d} e Δp_{k_d} . Realizamos uma comparação do modelo otimizado *versus* não otimizado. E verificamos que em relação as métricas do Poder de Classificação e Poder de Pontuação, nosso modelo permaneceu invariável com a parametrização que foi realizada.

Para identificação da combinação de cada método com cada grupo de atributos selecionados, rotulamos estas combinações por uma sigla, as quais foram utilizadas como referência nos gráficos das Figuras 54 a 59. Observamos que o método *Random Forest* rotulado como “RF_sep” combinado aos melhores atributos apresentou os melhores resultados, tanto para o modelo otimizado quanto não otimizado, para ambos atributos alvo.

Na Figura 54 observamos a avaliação do modelo para o poder de classificação, com a otimização e sem a otimização de parâmetros. Observamos que o modelo é quase insensível a parametrização.

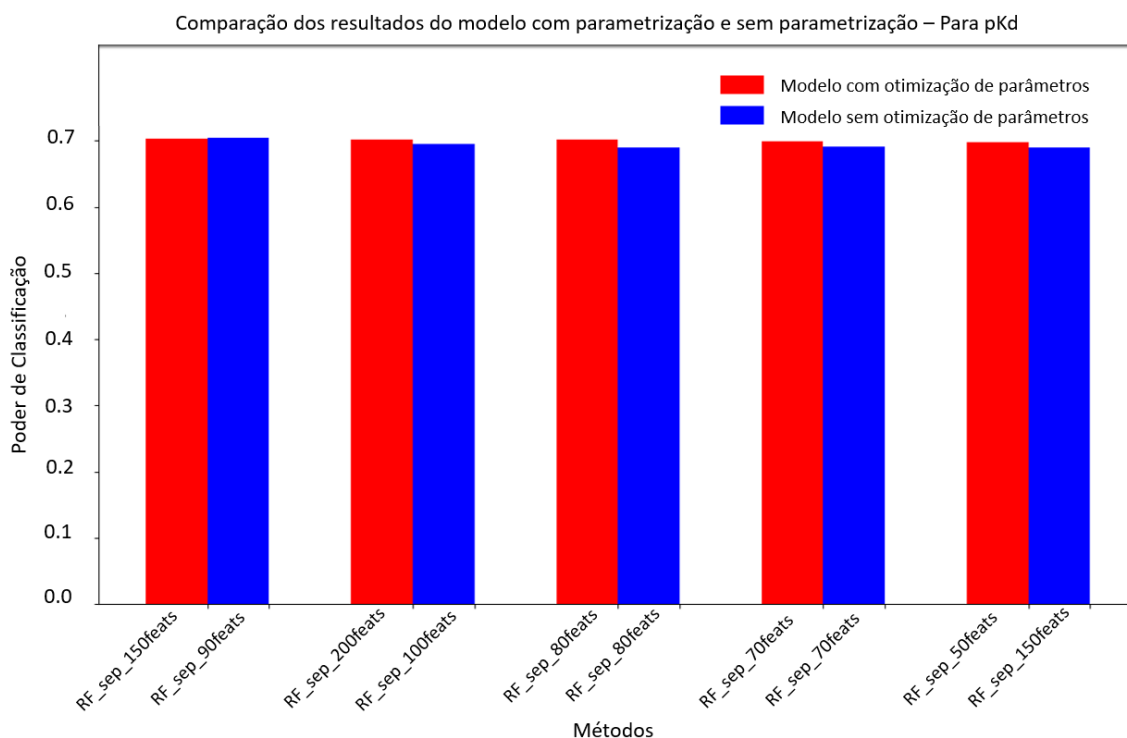


Figura 54: Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Classificação

Para o poder de classificação observamos na Figura 54 o mesmo comportamento, nosso modelo seguiu invariante a parametrização e o melhor método para a predição de p_{k_d} é o *Random Forest*.

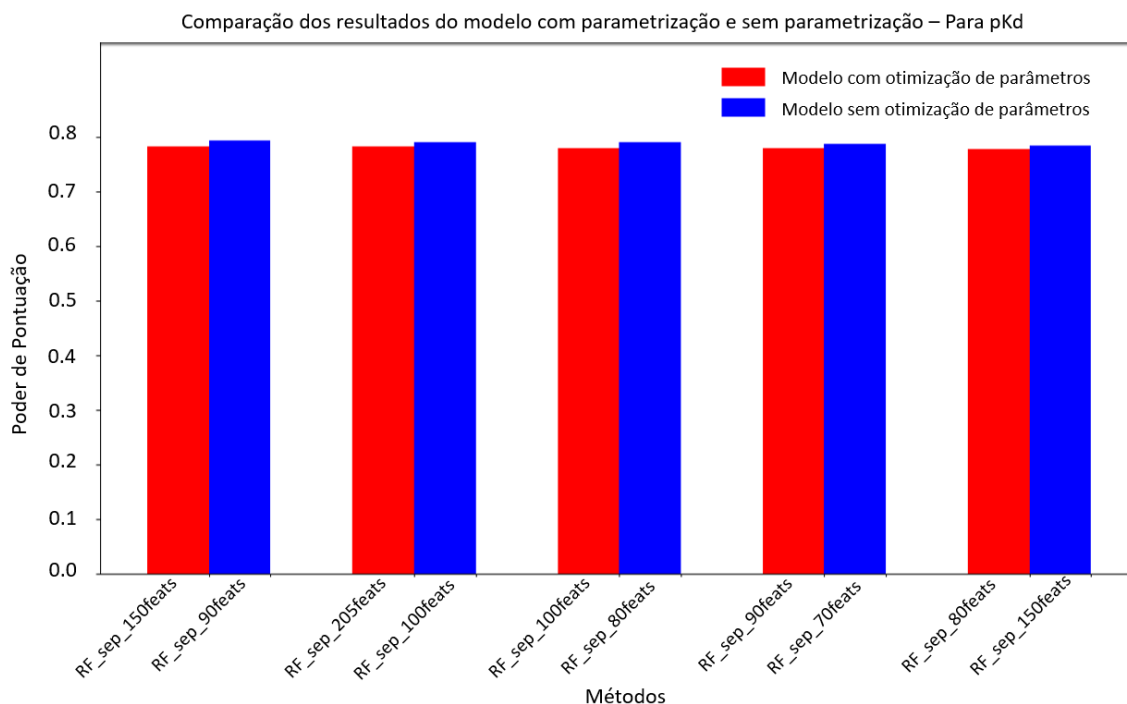


Figura 55: Comparação do modelo otimizado e não otimizado para p_{kd} - Avaliação do Poder de Pontuação

Na Figura 56, observamos que para o poder de pontuação, utilizando o atributo Δp_{kd} o modelo também é insensível a otimização de parâmetros, o método que resultou nas melhores valores para as predições foi o *Random Forest*.

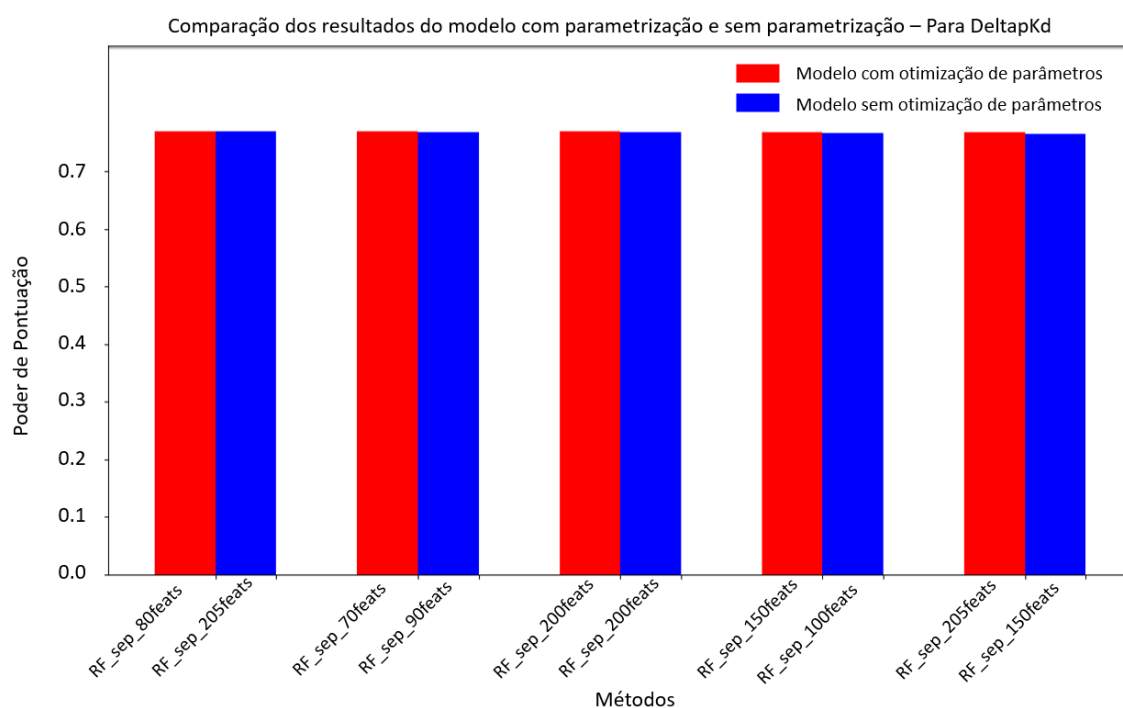


Figura 56: Comparação do modelo otimizado e não otimizado para Δp_{kd} - Avaliação do Poder de Pontuação

No poder de classificação, apresentado na Figura 57 para Δp_{k_d} também observamos que não há variação entre os valores, tanto utilizando parametrização, quanto não utilizando, os valores das predições para este atributo alvo são muito semelhantes.

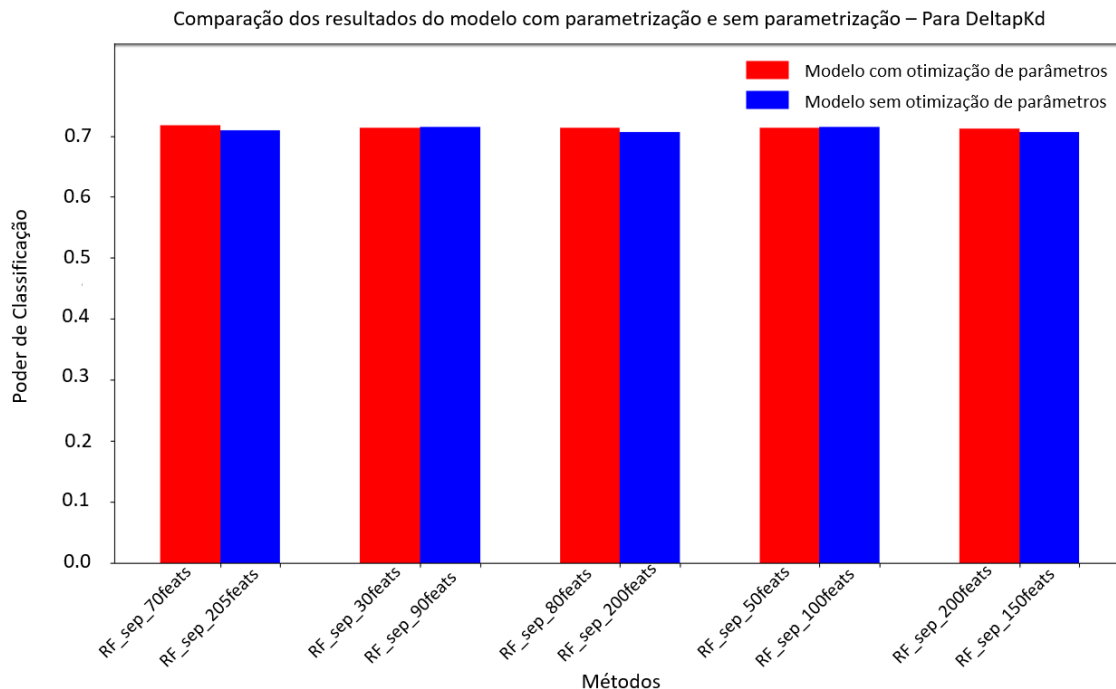


Figura 57: Comparação do modelo otimizado e não otimizado para Δp_{k_d} - Avaliação do Poder de Classificação

Embora nosso modelo tenha demonstrado escores muito próximos quando avaliado de acordo com as métricas do Poder de Classificação e do Poder de Pontuação, nossa motivação é que com a biblioteca pronta do Python e seguindo a documentação do Sckiy-learn, conseguimos obter um modelo que não é sensível a escolha de parâmetros quando utilizamos qualquer uma dessas duas métricas, tanto utilizando Δp_{k_d} ou p_{k_d} como atributo alvo.

Contudo, além destas avaliações, também analisamos nosso modelo em relação ao Poder de Ancoragem e observamos que alguns métodos, combinados a determinados conjuntos de atributos, obtiveram um escore um pouco mais alto quando não utilizamos a otimização de parâmetros.

Analisando os resultados das nossas cinco melhores pontuações e comparando nosso modelo otimizado *versus* não otimizado, observamos na Figura 58 que nosso modelo quando não é otimizado, utilizando como atributo alvo: Δp_{k_d} , apresenta escores mais elevados, quando utilizamos conjuntos compostos pelos nossos 15, 40 e 50 melhores atributos, que foram selecionados em etapa anterior, através de nosso *ranking acumulativo*, combinado ao método *Support Vector Machine*.

Para p_{k_d} , na Figura 59 utilizando o mesmo método, também observamos este tipo de comportamento. Porém, os escores mais altos foram aqueles em que utilizou-se os 80

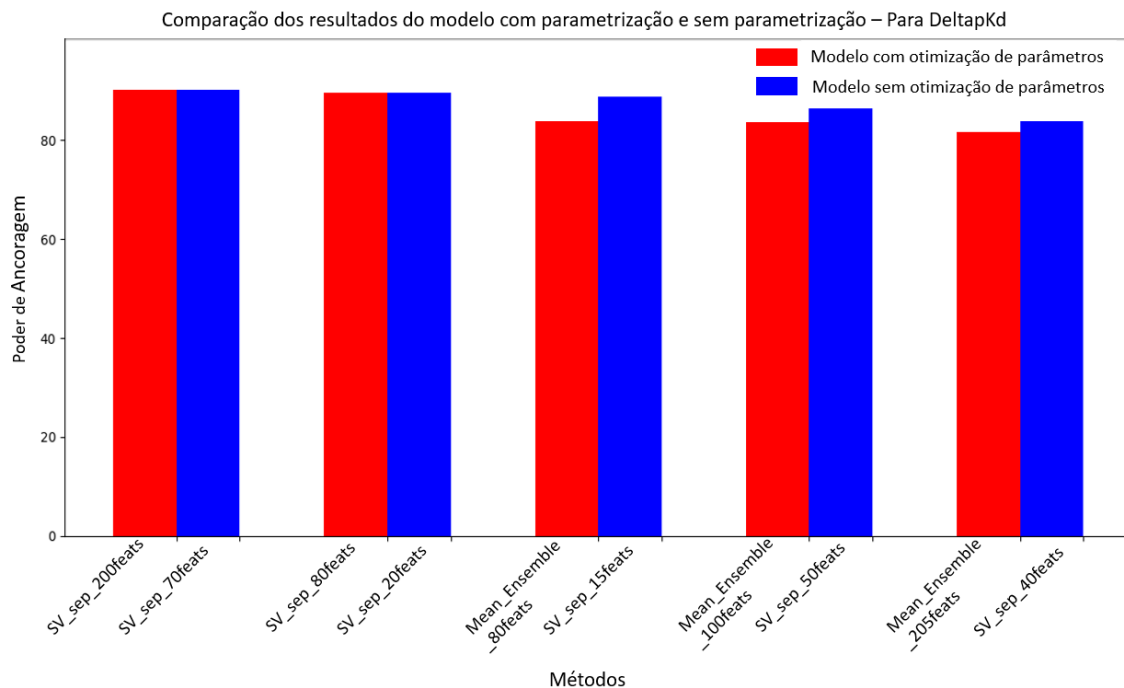


Figura 58: Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Ancoragem

melhores atributos. Para o método *Lasso* também constamos este mesmo padrão em nosso modelo, quando não otimizado. O método combinado aos 90, 80 e 70 melhores atributos, sem a otimização de parâmetros, retornou escores mais altos após a comparação de nossos cinco melhores escores na avaliação da métrica do Poder de Ancoragem.

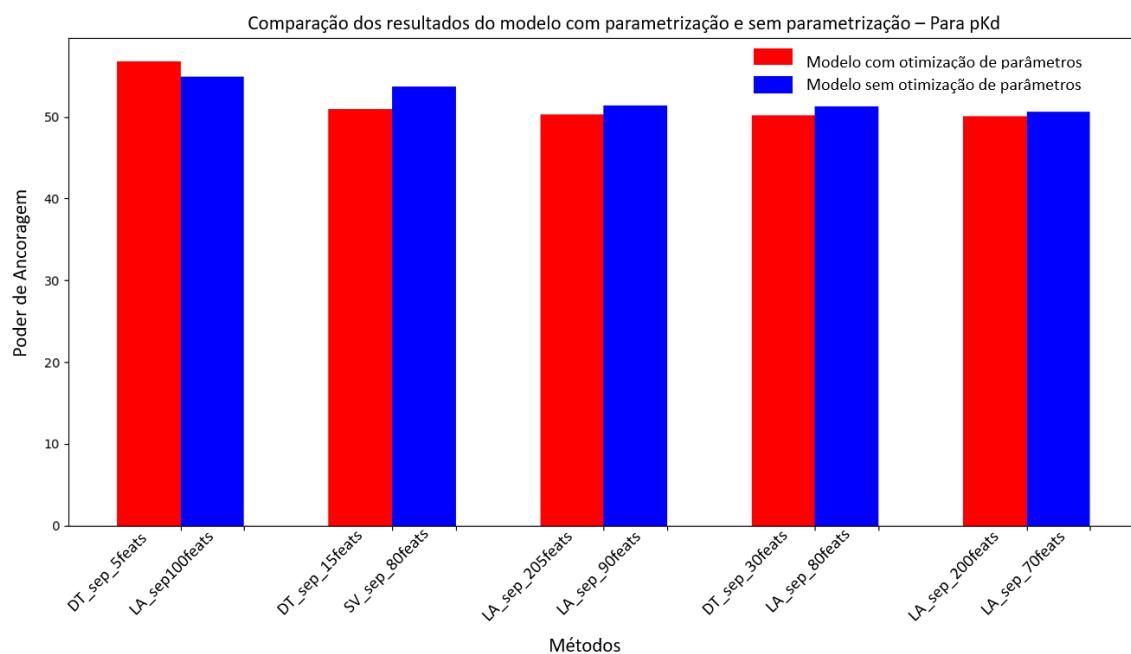


Figura 59: Comparação do modelo otimizado e não otimizado para p_{k_d} - Avaliação do Poder de Ancoragem

Contudo, cabe ressaltar que para o maior valor observado, ou seja, nosso maior escore, na avaliação dessa métrica, não houveram diferenças, pois o escore retornado foi o mesmo.

Após esta análise, cada combinação foi analisada individualmente e, finalmente, o desempenho geral foi avaliado com as outras funções de escore, de acordo com o guia comparativo de funções de pontuação CASF-2016.

Na avaliação do poder de classificação (*Ranking Power*), apresentado na Figura 60, para p_{k_d} observou-se que o modelo atingiu um coeficiente de *Spearman* de 0,703, utilizando nossos 150 melhores atributos, que foram classificados de acordo com o nosso *ranking acumulativo* e o método *Random Forest* ocupando o segundo lugar da classificação do CASF-2016. Para o poder de pontuação (*Scoring Power*) na Figura 61, também ocupamos a segunda colocação. Obtemos um coeficiente de *Pearson* de 0,782, utilizando o *Random Forest* e os nossos 150 melhores atributos.

Para Δp_{k_d} na Figura 62, também seguimos ocupando a segunda posição para o poder de classificação. Obtendo um coeficiente de *Spearman* de 0,718, utilizando o *Random Forest* e os 70 melhores atributos. Para o poder de pontuação na Figura 63, o coeficiente de *Pearson* de 0,769, o que fez com que permanecêssemos na segunda posição, utilizando o mesmo método e os 200 melhores atributos.

Em relação ao Poder de Ancoragem, verificamos que a capacidade do nosso modelo, em classificar o conjunto de dados de cada complexo proteína ligante e encontrar a pose de ligação melhor pontuada, quando comparado as funções de pontuação com base na avaliação do CASF, se igualou a pontuação do $\Delta VinaRF20$, tanto utilizando Δp_{k_d} quanto p_{k_d} como atributo alvo utilizando o método *Support Vector Machine*. Isso pode ser observado a partir dos gráficos apresentados nas Figuras 64 e 65.

Observamos também, a partir desses gráficos que o Δp_{k_d} combinado aos conjuntos melhores ranqueados, retornou escores mais elevados quando comparado à utilização do atributo p_{k_d} .

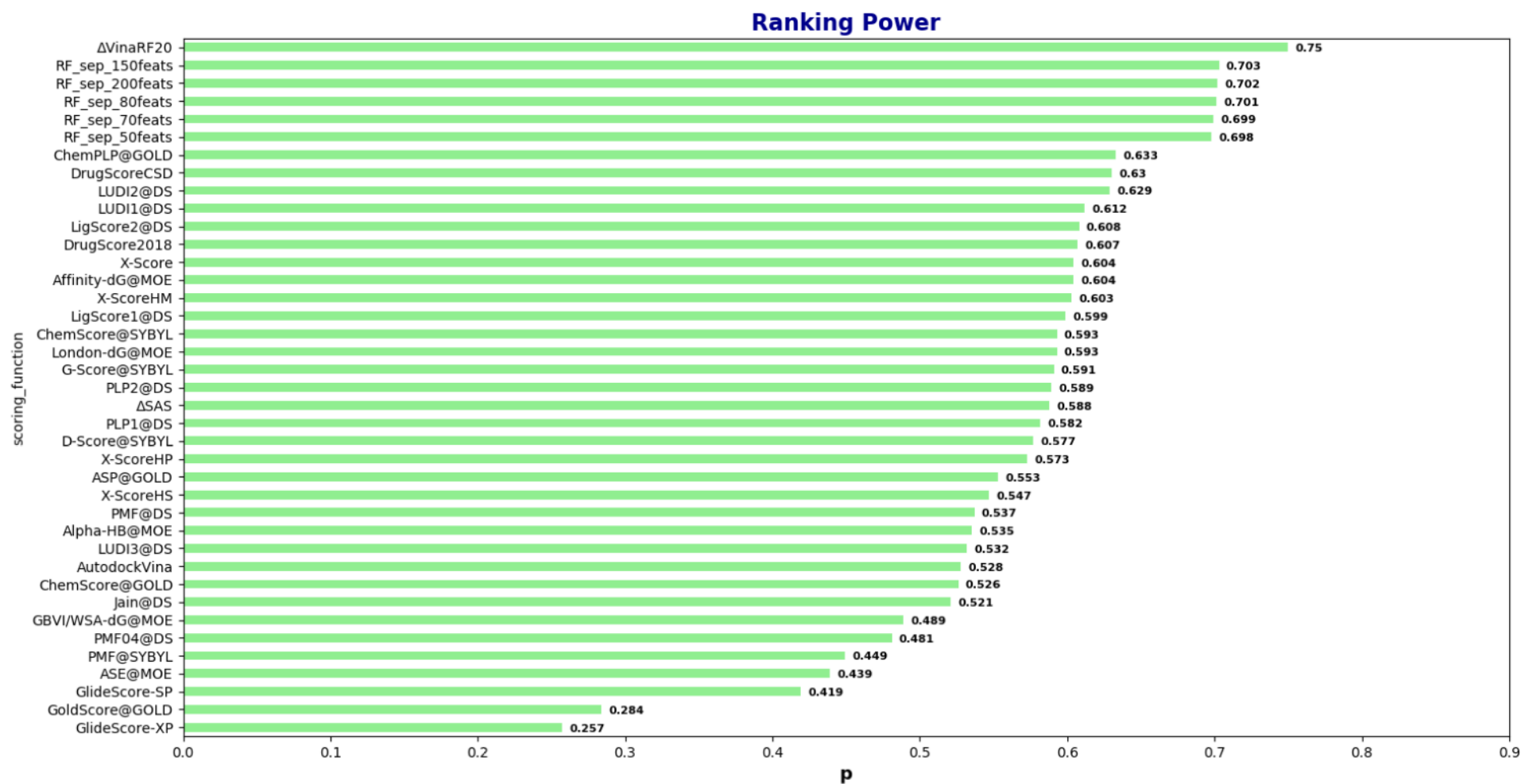


Figura 60: Comparação do Poder de Classificação para p_{k_d} com o CASF-2016

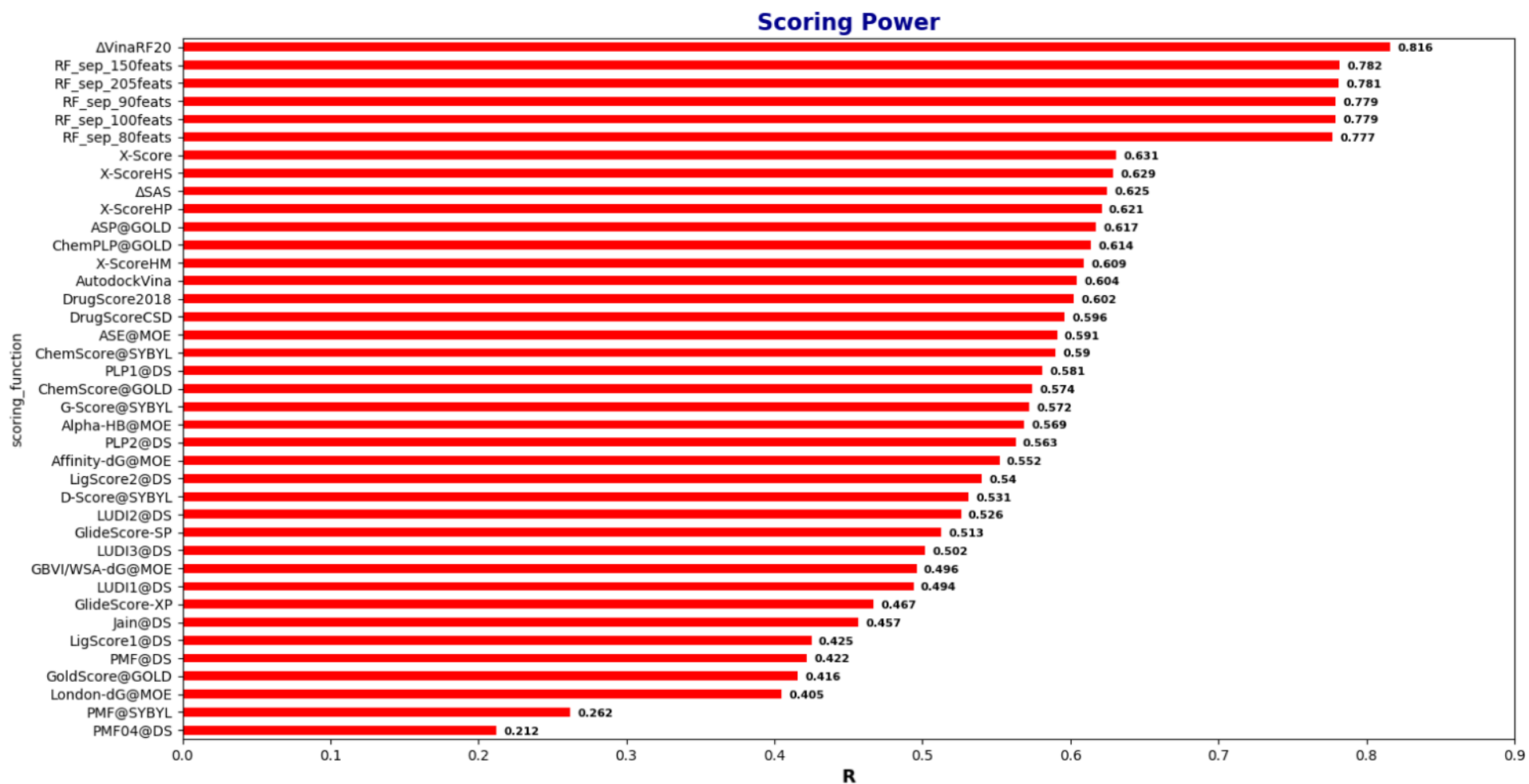


Figura 61: Comparação do Poder de Pontuação para p_{k_d} com o CASF-2016

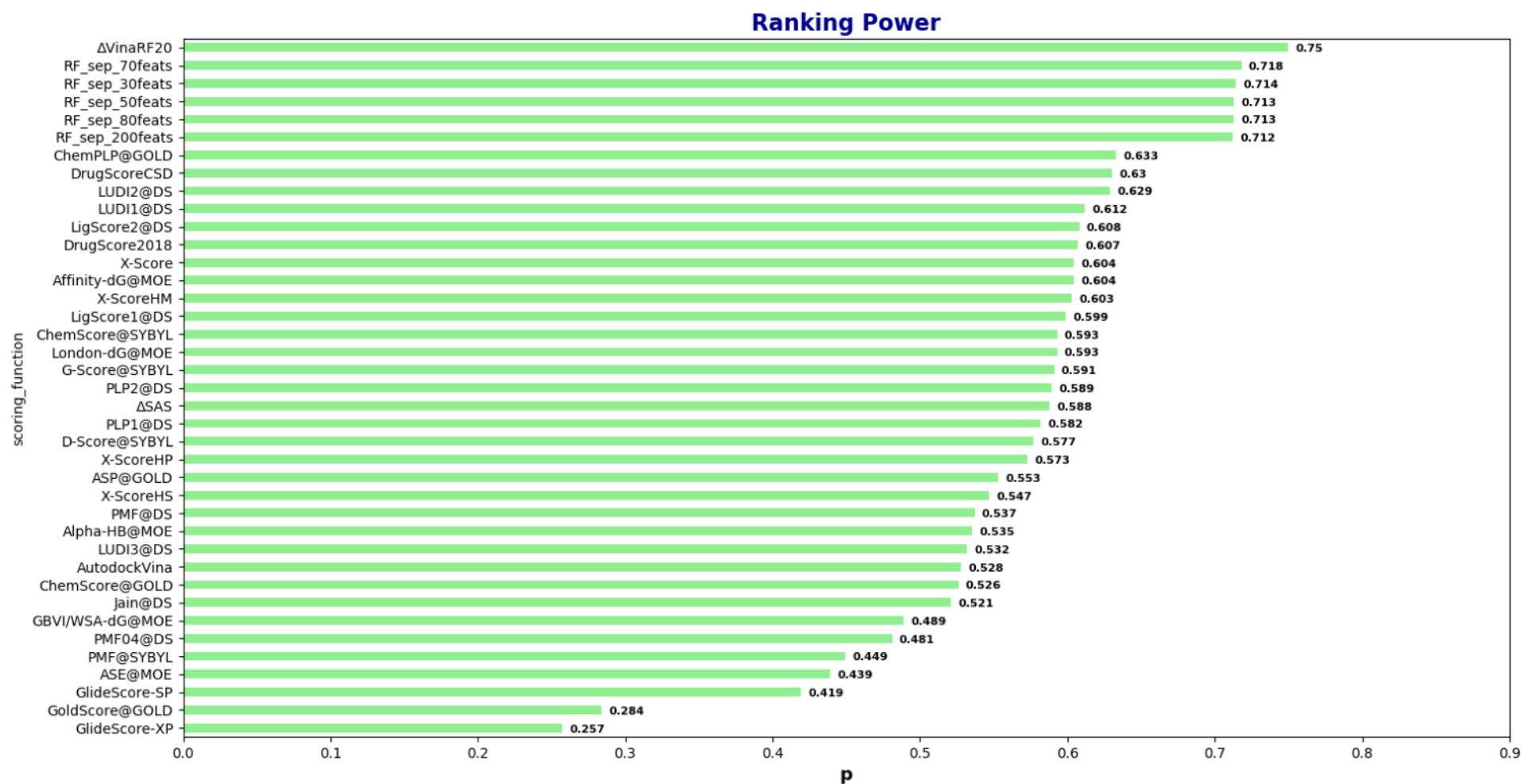


Figura 62: Comparação do Poder de Classificação para Δp_{k_d} com o CASF-2016

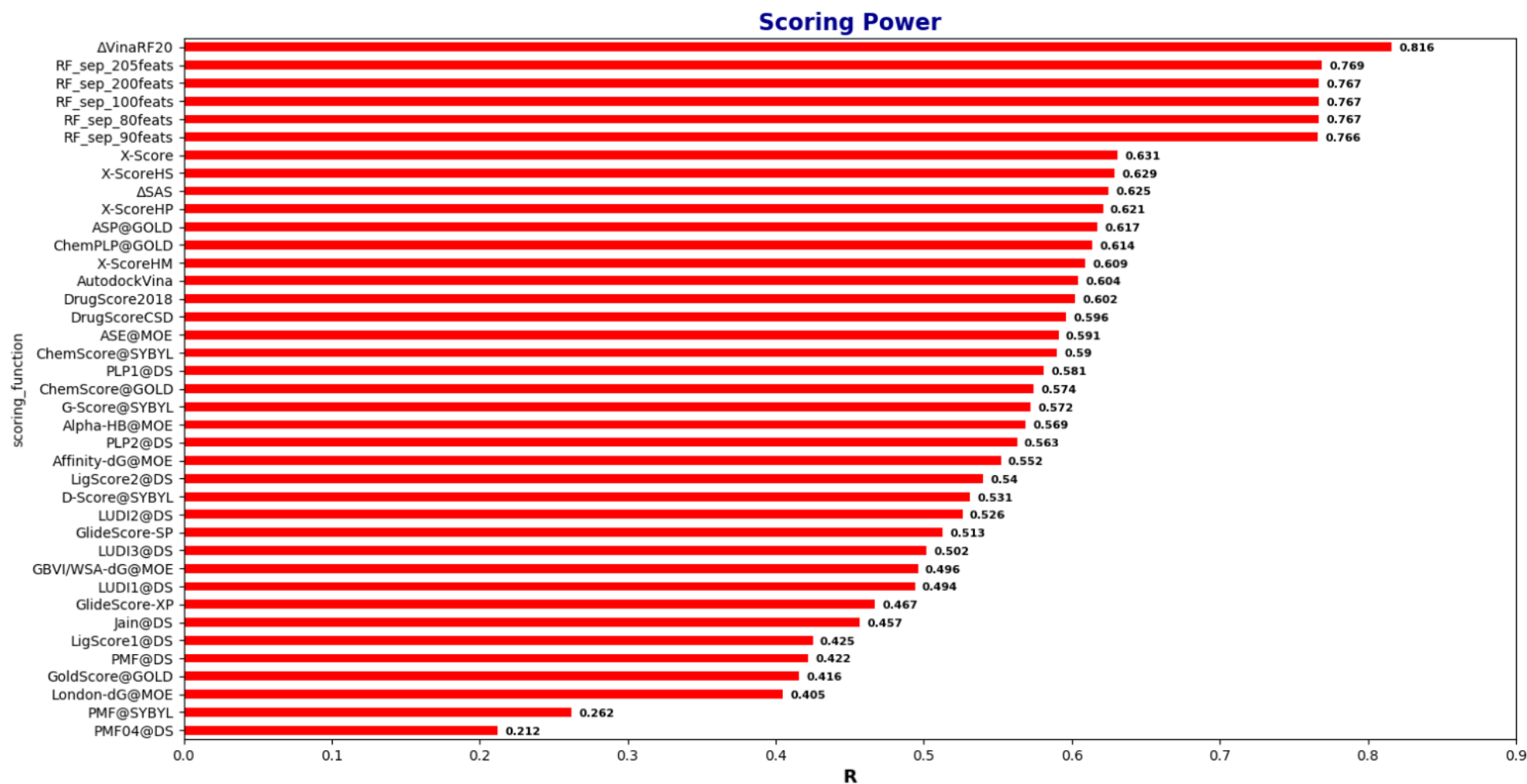


Figura 63: Comparação do Poder de Pontuação para Δp_{k_d} com o CASF-2016

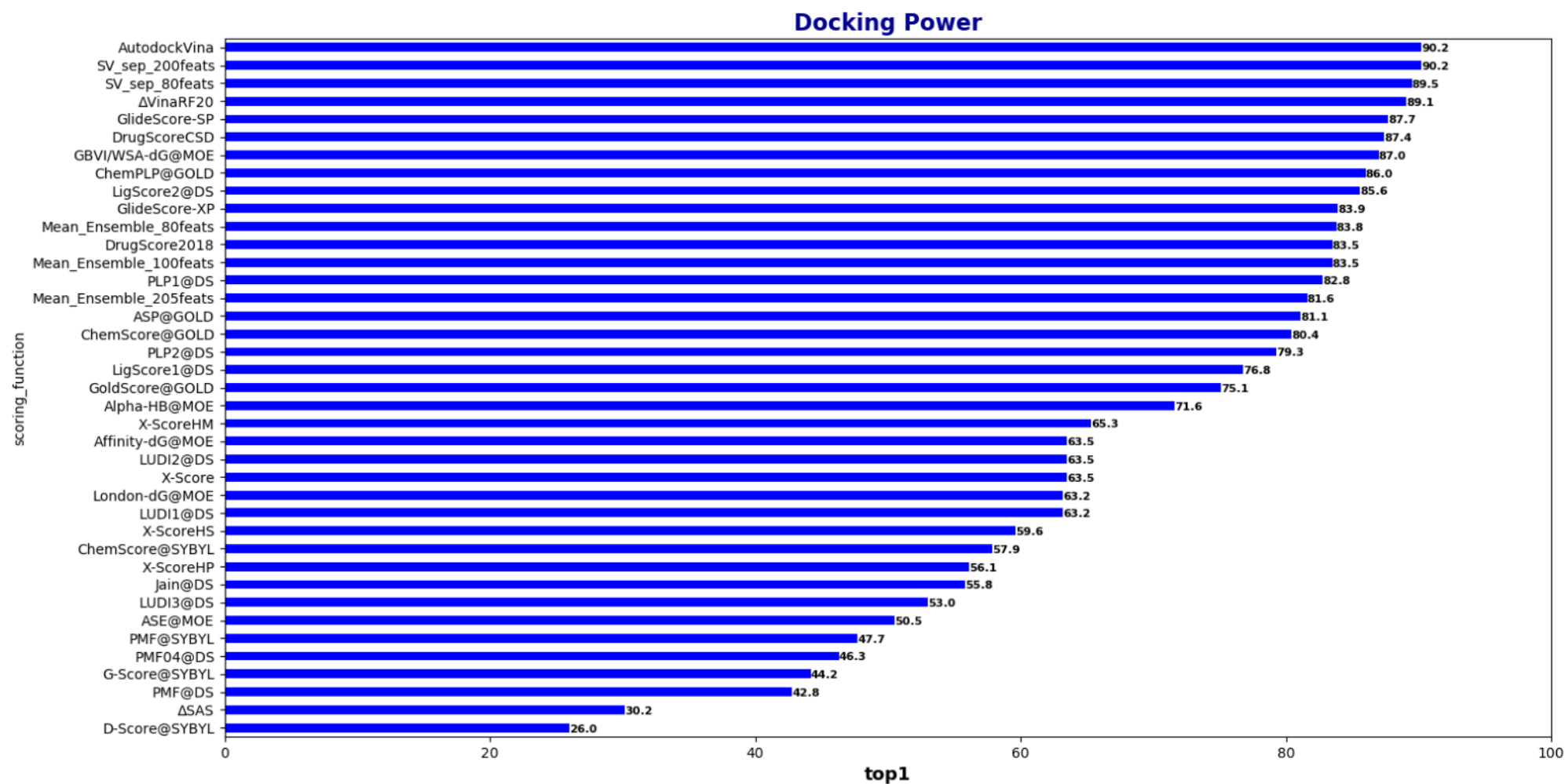


Figura 64: Comparação do Poder de Ancoragem para Δp_{k_d} com o CASF-2016

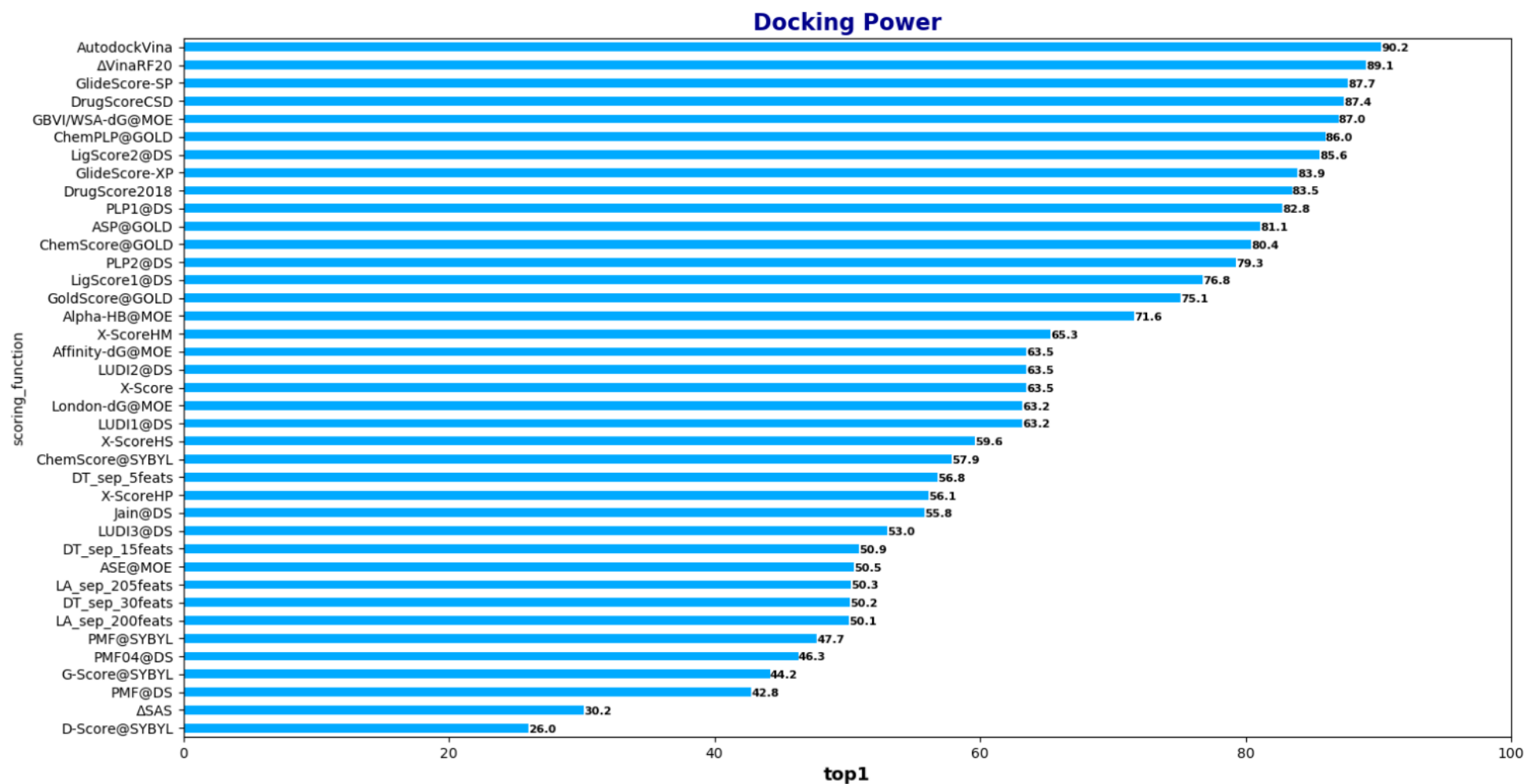


Figura 65: Comparação do Poder de Ancoragem para p_{k_d} com o CASF-2016

10 DISCUSSÕES

A seleção de atributos é realizada, dentre outros motivos, para combinar e identificar os atributos mais significativos em um conjunto de dados. Neste trabalho é apresentado um estudo comparativo entre diferentes tipos de medidas de avaliação de importância de atributos utilizando além da seleção de atributos, um sistema *Ensemble*.

Resultados experimentais demonstram que a seleção de atributos, por meio da redução da dimensionalidade, pode auxiliar na melhora da qualidade dos dados sob a perspectiva de desempenho preditivo, o que possivelmente contribui para a indução de modelos mais compreensíveis e com menor custo computacional.

Diferentes métricas, diferentes métodos e diferentes conjuntos de atributos podem ter um impacto diferente para estimar a energia de ligação de complexos proteína-ligante, utilizando funções de escore.

Muitas destas métricas, mencionadas ao longo deste trabalho são encontradas no artigo de Li et al. (2015). Analisando este estudo, observamos que a função DeltaVinaRF20, de acordo com os pesquisadores, apresenta o melhor poder de ancoragem. Isso quer dizer que essa função de pontuação consegue pontuar as conformações nativas de um ligante entre conformações “falsas” do ligante nativo (usando um corte de 2Å).

Fazendo um comparativo com os resultados que foram obtidos em nosso trabalho, em relação ao Poder de Classificação e do Poder de Pontuação e de acordo com a avaliação do CASF-2016, ocupamos a segunda posição deste *ranking*. Para essas métricas, não foi possível igualar o desempenho da nossa função de pontuação a do DeltaVinaRF20. Porém, observamos que a função de pontuação que foi proposta, conseguiu superar várias das funções de escore com licença comercial avaliadas no CASF-2016 em relação a estas duas métricas. Já em relação ao Poder de Ancoragem, conseguimos igualar nosso escore ao obtido pela função de pontuação DeltaVinaRF20, mas utilizando o método *Support Vector Machine* e aplicando várias técnicas já mencionadas anterior no decorrer deste estudo, como: Ensemble, técnicas de seleção de atributos e otimização de parâmetros.

11 CONCLUSÕES

O desenvolvimento de técnicas de triagem virtual tem sido bastante utilizada na identificação de novos compostos com atividade biológica, em especial de fármacos. A triagem virtual, através de inúmeras estratégias distintas, consegue realizar uma previsão da orientação e da conformação de uma molécula pequena (ligante da proteína) quando limitado a uma molécula receptora grande que é o alvo (proteína) (Lionta et al., 2014). Diversas abordagens com diferentes níveis de complexidade foram desenvolvidos para modelar as interações proteína-ligante e, assim, prever a afinidade de ligação. Porém, os mais precisos normalmente herdam altos custos computacionais, que os inviabilizam para aplicações em *virtual screening*. Como alternativa, os programas de *docking* preveem a afinidade ligante-receptor por meio de funções de ranqueamento (Guedes et al., 2014).

Normalmente, se analisa a saída dos programas de *docking* e se identifica manualmente as características que poderiam ser utilizadas para seleção de uma pose (“conformação de baixa energia”) ou para distinguir os ligantes ativos (Plewczynski et al., 2011). Devido ao grande número de conformações possíveis, a proteína é geralmente tratada de forma rígida (i.e., em uma única conformação observada por cristalografia) ou com flexibilidade limitada. O problema inerente a essa abordagem rígida é que ela leva a uma baixa previsibilidade para os complexos que não estão em conformidade com as premissas de modelagem (Ballester and Mitchell, 2010).

Para resolver esse problema, muitas técnicas tradicionais foram propostas, no entanto, o desempenho em relação a previsibilidade da afinidade de ligação entre complexos proteína-ligante utilizando funções de pontuação foi insatisfatório. Somente nos últimos anos a aplicação da tecnologia de aprendizado de máquina foi aplicada no projeto de funções de pontuação; e os resultados foram muito promissores (Schmidt et al., 2019).

Atualmente, pesquisas demonstraram uma série de benefícios com relação a utilização destas técnicas. A principal vantagem de utilizar aprendizado de máquina para predição da afinidade de ligação é que essa técnica não impõe nenhuma forma funcional específica para a função de pontuação (Ain et al., 2015). Assim, qualquer tipo possível de interação pode ser inferida diretamente a partir dos dados experimentais. Isso possibilita um maior desempenho para prever as afinidades de ligação de diversos complexos proteína-ligante.

De fato, o grau em que as funções de aprendizado de máquina superaram as funções clássicas na predição de afinidade foi destacado em vários estudos, como observado através das pesquisas bibliográficas que foram realizadas ao longo deste trabalho (Ain et al., 2015).

Contudo, estas funções envolvem vários parâmetros físico-químicos para identificação dos ligantes, que podem ser modificados para melhorar as previsões. Para estimar estes parâmetros, métodos de aprendizado de máquina vêm sendo utilizados para selecionar as características realmente relevantes para que ocorra uma previsão da função das proteínas com exatidão (Li et al., 2019).

Neste contexto, há uma clara necessidade de se analisar a produção científica sobre a utilização do aprendizado de máquina para seleção de *features* que visam melhorar o desempenho da predição da afinidade de ligação. Para alcançar este objetivo, este trabalho apresentou uma metodologia para o problema da predição da afinidade de ligação entre complexos proteína-ligante. Utilizando três métodos de aprendizado de máquina para seleção de atributos. O presente estudo teve como propósito, confirmar se os atributos utilizados por Wang and Zhang (2017) eram de fato os melhores atributos para predição da afinidade de ligação ao empregar a função de pontuação DeltaVinaRF20.

Foi realizado um experimento, utilizando linguagem de programação Python e o pacote da biblioteca *scikit-learn* para encontrar um subconjunto de atributos, envolvendo as características físico químicas que possuíam maior impacto para prever o fator de correção proposto por Wang and Zhang (2017). Com a metodologia proposta, foi possível encontrar o melhor conjunto de características que visa contribuir para uma melhor afinidade de ligação entre proteína-ligante.

Utilizamos essa classificação para fazer uma sub seleção do conjunto de atributos e melhorar a predição das características físico químicas mais importantes utilizando outros métodos de regressão. Por fim, combinamos os resultados para *ensemble score*, realizamos a otimização dos parâmetros do modelo e adicionamos novas características, com objetivo de melhorar o desempenho de ambos classificadores. Conseguimos obter um desempenho significativo em relação as métricas do Poder de Classificação e Poder de Pontuação. Ainda não conseguimos superar o DeltaVinaRF20, mas conseguimos obter o mesmo score para o Poder de Ancoragem e além disto, superar várias das funções de pontuações mais bem pontuadas de acordo com a avaliação do CASF-2016.

REFERÊNCIAS

- Ain, Q. U., Aleksandrova, A., Roessler, F. D., and Ballester, P. J. (2015). Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 5(6):405–424.
- Alberts, B., Johnson, A., Lewis, J., Morgan, D., Raff, M., Roberts, K., Walter, P., Wilson, J., and Hunt, T. (2010). *Biologia molecular da célula*. Artmed Editora.
- Almeida, A. B. d. (2016). Predição de estrutura terciária de proteínas com técnicas multiobjetivo no algoritmo de monte carlo. Master's thesis, Universidade Federal de Goiás.
- Ashtawy, H. M. and Mahapatra, N. R. (2012). A comparative assessment of ranking accuracies of conventional and machine-learning-based scoring functions for protein-ligand binding affinity prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(5):1301–1313.
- Ashtawy, H. M. and Mahapatra, N. R. (2013). Molecular docking for drug discovery: machine-learning approaches for native pose prediction of protein-ligand complexes. In *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 15–32. Springer.
- Ballester, P. J. and Mitchell, J. B. (2010). A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169–1175.
- Bock, J. R. and Gough, D. A. (2002). A new method to estimate ligand-receptor energetics. *Molecular & Cellular Proteomics*, 1(11):904–910.
- Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.
- Cai, J., Luo, J., Wang, S., and Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79.

- Carvalho, A., Faceli, K., LORENA, A., and Gama, J. (2011). Inteligência artificial—uma abordagem de aprendizado de máquina.
- Chaudhary, K. K. and Mishra, N. (2016). A review on molecular docking: novel tool for drug discovery. *databases*, 3(4).
- Cheng, T., Li, X., Li, Y., Liu, Z., and Wang, R. (2009). Comparative assessment of scoring functions on a diverse test set. *Journal of chemical information and modeling*, 49(4):1079–1093.
- Congreve, M., Murray, C. W., and Blundell, T. L. (2005). Keynote review: Structural biology and drug discovery. *Drug discovery today*, 10(13):895–907.
- de Oliveira Rodrigues, T. A., Leijôto, L. F., Brandão, P. C. O., and Nobre, C. N. (2015). Predição de função de proteínas através da extração de características físico-químicas. *Revista de Informática Teórica e Aplicada*, 22(1):29–51.
- Dei, M. (2019). Hyperparameter tuning explained — tuning phases, tuning methods, bayesian optimization, and sample code! [Online; accessed 30-Julho-2020].
- Dias, U. M. et al. (2007). Predição da função das proteínas sem alinhamentos usando máquinas de vetor de suporte. Master's thesis, Universidade Federal de Alagoas.
- Durrant, J. D., Friedman, A. J., Rogers, K. E., and McCammon, J. A. (2013). Comparing neural-network scoring functions and the state of the art: applications to common library screening. *Journal of chemical information and modeling*, 53(7):1726–1735.
- Escovedo, T. and Koshiyama, A. (2020). *Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise*. Casa do Código.
- Ewing, T. J., Makino, S., Skillman, A. G., and Kuntz, I. D. (2001). Dock 4.0: search strategies for automated molecular docking of flexible molecule databases. *Journal of computer-aided molecular design*, 15(5):411–428.
- Faceli, K., Lorena, A. C., Gama, J., Carvalho, A. C. P. d. L., et al. (2011). Inteligência artificial: Uma abordagem de aprendizado de máquina.
- Fogel, G. B., Tran, J., Johnson, S., and Hecht, D. (2010). Machine learning approaches for customized docking scores: Modeling of inhibition of mycobacterium tuberculosis enoyl acyl carrier protein reductase. In *2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–6. IEEE.
- Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer.

- Gabel, J., Desaphy, J., and Rognan, D. (2014). Beware of machine learning-based scoring functions on the danger of developing black boxes. *Journal of chemical information and modeling*, 54(10):2807–2815.
- Gaillard, T. (2018). Evaluation of autodock and autodock vina on the casf-2013 benchmark. *Journal of chemical information and modeling*, 58(8):1697–1706.
- Gao, H., Aderhold, A., Mangion, K., Luo, X., Husmeier, D., and Berry, C. (2017). Changes and classification in myocardial contractile function in the left ventricle following acute myocardial infarction. *Journal of the Royal Society Interface*, 14(132):20170203.
- Gao, Y. and Sun, S. (2010). An empirical evaluation of linear and nonlinear kernels for text classification using support vector machines. In *2010 seventh international conference on fuzzy systems and knowledge discovery*, volume 4, pages 1502–1505. IEEE.
- Guedes, I. A., de Magalhães, C. S., and Dardenne, L. E. (2014). Receptor–ligand molecular docking. *Biophysical reviews*, 6(1):75–87.
- Han, J., Kamber, M., and Pei, J. (2006). Data mining: concepts and techniques. 2001. San Francisco: Morgan Kauffman.
- Harrington, P. (2012). *Machine learning in action*. Manning Publications Co.
- Hecht, D. and Fogel, G. B. (2009). Computational intelligence methods for docking scores. *Current Computer-Aided Drug Design*, 5(1):56–68.
- Huang, S.-Y. and Zou, X. (2010). Mean-force scoring functions for protein–ligand binding. In *Annual Reports in Computational Chemistry*, volume 6, pages 280–296. Elsevier.
- Kamber, M. and Pei, J. (2006). *Data Mining*. Morgan kaufmann.
- Khamis, M. A. and Gomaa, W. (2015). Comparative assessment of machine-learning scoring functions on pdbbind 2013. *Engineering Applications of Artificial Intelligence*, 45:136–151.
- Koes, D. R., Baumgartner, M. P., and Camacho, C. J. (2013). Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, 53(8):1893–1904.
- Landrum, G. (2013). Rdkit documentation. *Release*, 1:1–79.
- Lengauer, T. and Rarey, M. (1996). Computational methods for biomolecular docking. *Current opinion in structural biology*, 6(3):402–406.
- Li, C. (2019). A gentle introduction to gradient boosting, 2019.

- Li, H., Leung, K.-S., Wong, M.-H., and Ballester, P. J. (2015). Improving autodock vina using random forest: the growing accuracy of binding affinity prediction by the effective exploitation of larger data sets. *Molecular informatics*, 34(2-3):115–126.
- Li, J., Fu, A., and Zhang, L. (2019). An overview of scoring functions used for protein–ligand interactions in molecular docking. *Interdisciplinary Sciences: Computational Life Sciences*, pages 1–9.
- Li, Y., Han, L., Liu, Z., and Wang, R. (2014). Comparative assessment of scoring functions on an updated benchmark: 2. evaluation methods and general results. *Journal of chemical information and modeling*, 54(6):1717–1736.
- Lionta, E., Spyrou, G., K Vassilatis, D., and Cournia, Z. (2014). Structure-based virtual screening for drug discovery: principles, applications and recent advances. *Current topics in medicinal chemistry*, 14(16):1923–1938.
- Lior, R. et al. (2014). *Data mining with decision trees: theory and applications*, volume 81. World scientific.
- Liu, J. and Wang, R. (2015). Classification of current scoring functions. *Journal of chemical information and modeling*, 55(3):475–482.
- Lopez-Rincon, A., Martinez-Archundia, M., Martinez-Ruiz, G. U., Schoenhuth, A., and Tonda, A. (2019). Automatic discovery of 100-mirna signature for cancer classification using ensemble feature selection. *BMC bioinformatics*, 20(1):480.
- Maheshwari, V. (2019). Implementation of random forest. [Online; accessed 15-Julho-2020].
- Merrill, J., Ward, G., Kamkar, S., Budzik, J., and Merrill, D. (2019). Generalized integrated gradients: A practical method for explaining diverse ensembles. *arXiv preprint arXiv:1909.01869*.
- Mitchell, R., Michalski, J., and Carbonell, T. (2013). *An artificial intelligence approach*. Springer.
- Monard, M. C. and Baranauskas, J. A. (2003). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32.
- Nelson, D. L. and Cox, M. M. (2018). *Princípios de Bioquímica de Lehninger-7*. Artmed Editora.
- Paananen, T., Piironen, J., Andersen, M. R., and Vehtari, A. (2019). Variable selection for gaussian processes via sensitivity analysis of the posterior predictive distribution. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1743–1752.

- Paris, B. M. et al. (2017). Learning to rank: combinação de algoritmos aplicando stacking e análise dos resultados. Master's thesis, Universidade Presbiteriana Mackenzie.
- Pason, L. P. and Sottriffer, C. A. (2016). Empirical scoring functions for affinity prediction of protein-ligand complexes. *Molecular informatics*, 35(11-12):541–548.
- Passari, L. M. Z. G., Soares, P. K., Bruns, R. E., Scarminio, I. S., et al. (2011). Statistics applied to chemistry: Ten common doubts [estatística aplicada à química: Dez dúvidas comuns]. *Química Nova*.
- Pathak, Y., Rana, P. S., Singh, P., and Saraswat, M. (2016). Protein structure prediction ($\text{rmsd} \leq 5 \text{ \AA}$) using machine learning models. *IJDMB*, 14(1):71–85.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.
- Plewczynski, D., Łażniewski, M., Grotthuss, M. V., Rychlewski, L., and Ginalski, K. (2011). Votedock: consensus docking method for prediction of protein–ligand interactions. *Journal of computational chemistry*, 32(4):568–581.
- Polishchuk, P. (2015). Application and efficacy of random forest method for qsar analysis. [Online; accessed 08-Julho-2020].
- PUC-Rio (2019). Seleção de características. [Online; accessed 02-Agosto-2020].
- Qi, Y., Minka, T. P., Picard, R. W., and Ghahramani, Z. (2004). Predictive automatic relevance determination by expectation propagation. In *Proceedings of the twenty-first international conference on Machine learning*, page 85.
- Quiroga, R. and Villarreal, M. A. (2016). Vinardo: A scoring function based on autodock vina improves scoring, docking, and virtual screening. *PloS one*, 11(5):e0155183.
- Rocha, K. B. and de Azevedo Júnior, W. F. (2009). Virtual screening para identificação de inibidores da corismato sintase de mycobacterium tuberculosis.
- Sammut, C. and Webb, G. I. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.
- Santos, B. C. d. (2017). *Algoritmo Genético Multiobjetivo para Seleção de Atributos Aplicando a Predição de Função de Proteína*. PhD thesis, Pontífica Universidade Católica de Minas Gerais.
- Sayad, S. (2019). Support vector machine - regression (svr). [Online; accessed 01-Agosto-2020].

- Schmidt, J., Marques, M. R., Botti, S., and Marques, M. A. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36.
- Schulz-Gasch, T. and Stahl, M. (2004). Scoring functions for protein–ligand interactions: a critical perspective. *Drug Discovery Today: Technologies*, 1(3):231–239.
- Serrano, G. L. (2019). *Grokking Machine Learning MEAP*. Manning Publications.
- Shafiei, E. and Jazayeri-Rad, H. (2012). Improving the identification performance of an industrial process using multiple neural networks. *American Journal of Intelligent Systems*, 2(4):40–44.
- Shamsara, J. (2018). Correlation between virtual screening performance and binding site descriptors of protein targets. *International journal of medicinal chemistry*, 2018.
- Shen, C., Ding, J., Wang, Z., Cao, D., Ding, X., and Hou, T. (2020). From machine learning to deep learning: Advances in scoring functions for protein–ligand docking. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 10(1):e1429.
- Singh, S. and Gupta, P. (2014). Comparative study id3, cart and c4. 5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, 27(27):97–103.
- Su, M., Yang, Q., Du, Y., Feng, G., Liu, Z., Li, Y., and Wang, R. (2018). Comparative assessment of scoring functions: the casf-2016 update. *Journal of chemical information and modeling*, 59(2):895–913.
- Távora, E. G. M., Vellasco, M. M. R., Horta, B. A., and Custodio, F. L. (2017). Feature importance calculation and protein quality assessment on the decoy discrimination problem. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 248–255. IEEE.
- Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.
- Trott, O. and Olson, A. J. (2010). Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461.
- Verli, H. (2014). *Bioinformática: da biologia à flexibilidade molecular*. Sociedade Brasileira de Bioquímica e Biologia Molecular.

Voet, D., Voet, J. G., and Pratt, C. W. (2014). *Fundamentos de Bioquímica-: A Vida em Nível Molecular*. Artmed Editora.

Wang, C. and Zhang, Y. (2017). Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest. *Journal of computational chemistry*, 38(3):169–177.

Wang, Y., Guo, Y., Kuang, Q., Pu, X., Ji, Y., Zhang, Z., and Li, M. (2015). A comparative study of family-specific protein–ligand complex affinity prediction based on random forest approach. *Journal of computer-aided molecular design*, 29(4):349–360.

Wang, Z., Sun, H., Yao, X., Li, D., Xu, L., Li, Y., Tian, S., and Hou, T. (2016). Comprehensive evaluation of ten docking programs on a diverse set of protein–ligand complexes: the prediction accuracy of sampling power and scoring power. *Physical Chemistry Chemical Physics*, 18(18):12964–12975.

Watson, J. D., Baker, T. A., Bell, S. P., Gann, A., Levine, M., and Losicke, R. (2015). *Biologia molecular do gene*. Artmed Editora.

Yiu, T. (2019). Understanding random forest how the algorithm works and why it is so effective. [Online; accessed 10-Julho-2020].

Yung-Chi, C. and Prusoff, W. H. (1973). Relationship between the inhibition constant (k_i) and the concentration of inhibitor which causes 50 per cent inhibition (i_{50}) of an enzymatic reaction. *Biochemical pharmacology*, 22(23):3099–3108.

Zhang, L., Zheng, C., Li, T., Xing, L., Zeng, H., Li, T., Yang, H., Cao, J., Chen, B., and Zhou, Z. (2017). Building up a robust risk mathematical platform to predict colorectal cancer. *Complexity*, 2017.

ANEXO A CONJUNTO DE DESCRITORES

A.1 Descritores do DeltaVina RF20

Tabela 7: Descritores gerados pelo DeltaVinaRF20. Fonte: Adaptado de (Wang and Zhang, 2017).

Identificador	Feature
<u>Autodock Vina Interaction</u>	
F1	non hydrohobic
F2	hydrogen bond
F3	solvation
F4-F5	electrostatic
<u>Autodock Vina Ligand Dependent Terms</u>	
F6	number of heavy atoms
F7	number of hydrophobic atoms
F8	number of torsion
F9	number of rotors
F10	ligand length
<u>bSASA Features</u>	
F11	positive
F12	negative
F13	donor-acceptor
F14	donor
F15	acceptor
F16	aromatic
F17	hydrophobic
F18	polar
F19	halogen
F20	total bSASA

A.2 Descritores do AutoDock Vina

Tabela 8: Descritores gerados pelo AutoDock Vina Fonte: Adaptado de (Wang and Zhang, 2017).

Tipo	Termos	Parâmetros	Número de termos
Interação Intermolecular	ad4_desolvation	q	2
	electrostatic	x	2
	gauss	o, w	31
	repulsion	o	8
	hydrophobic		4
	non_hydrophobic		1
	vdW		1
	hydrogen bond	b	3
Conformação Independente	num_tors		1
	num_rotors		1
	num_heavy_atoms		1
	num_hydrophobic_atoms		1
	ligand_max_num_h_bonds		1
	ligand_lengths_sum		1

A.3 Descritores gerados com RDKit

Tabela 9: Características bidimensionais e tridimensionais dos ligantes calculadas com o RDKit. Fonte: (Landrum, 2013).

Tipo	Descritores
2D descriptors	BalabanJ, BertzCT, Ipc, HallKierAlpha, Kappa1, Kappa2, Kappa3, Chi0, Chi1, Chi0n, Chi1n, Chi2n, Chi3n, Chi4n, Chi0v, Chi1v, Chi2v, Chi3v, Chi4v, MolLogP, MolMR, MolWt, ExactMolWt, HeavyAtomCount, HeavyAtomMolWt, NHOH-Count, NOCount, NumHAcceptors, NumHDonors, NumHeteroatoms, NumRotatableBonds, NumValenceElectrons, NumAmideBonds, NumAromaticRings, NumSaturatedRings, NumAliphaticRings, NumAromaticHeterocycles, NumSaturatedHeterocycles, NumAliphaticHeterocycles, NumAromaticCarbocycles, NumSaturatedCarbocycles, NumAliphaticCarbocycles, RingCount, FractionCSP3, NumSpiroAtoms, NumBridgeheadAtoms, TPSA, LabuteASA, PEOE_VSA1, PEOE_VSA2, PEOE_VSA3, PEOE_VSA4, PEOE_VSA5, PEOE_VSA6, PEOE_VSA7, PEOE_VSA8, PEOE_VSA9, PEOE_VSA10, PEOE_VSA11, PEOE_VSA12, PEOE_VSA13, PEOE_VSA14, SMR_VSA1, SMR_VSA2, SMR_VSA3, SMR_VSA4, SMR_VSA5, SMR_VSA6, SMR_VSA7, SMR_VSA8, SMR_VSA9, SMR_VSA10, SlogP_VSA1, SlogP_VSA2, SlogP_VSA3, SlogP_VSA4, SlogP_VSA5, SlogP_VSA6, SlogP_VSA7, SlogP_VSA8, SlogP_VSA9, SlogP_VSA10, SlogP_VSA11, SlogP_VSA12, EState_VSA1, EState_VSA2, EState_VSA3, EState_VSA4, EState_VSA5, EState_VSA6, EState_VSA7, EState_VSA8, EState_VSA9, EState_VSA10, EState_VSA11, VSA_EState1, VSA_EState2, VSA_EState3, VSA_EState4, VSA_EState5, VSA_EState6, VSA_EState7, VSA_EState8, VSA_EState9, VSA_EState10, MQNs_atom_counts_c, MQNs_atom_counts_f, MQNs_atom_counts_cl, MQNs_atom_counts_br, MQNs_atom_counts_i, MQNs_atom_counts_s, MQNs_atom_counts_p, MQNs_atom_counts_an, MQNs_atom_counts_cn, MQNs_atom_counts_ao, MQNs_atom_counts_co, MQNs_atom_counts_hac, MQNs_bond_counts_asb, MQNs_bond_counts_adb, MQNs_bond_counts_atb, MQNs_bond_counts_csb, MQNs_bond_counts_cdb, MQNs_bond_counts_ctb, MQNs_bond_counts_rbc, MQNs_polarity_counts_hbam, MQNs_polarity_counts_hba, MQNs_polarity_counts_hbdm, MQNs_polarity_counts_hbd, MQNs_polarity_counts_negc, MQNs_polarity_counts_posc, MQNs_topology_counts_asv, MQNs_topology_counts_adv, MQNs_topology_counts_atv, MQNs_topology_counts_aqv, MQNs_topology_counts_cdv, MQNs_topology_counts_ctv, MQNs_topology_counts_cqv, MQNs_topology_counts_r3, MQNs_topology_counts_r4, MQNs_topology_counts_r5, MQNs_topology_counts_r6, MQNs_topology_counts_r7, MQNs_topology_counts_r8, MQNs_topology_counts_rg, MQNs_topology_counts_rgIO, MQNs_topology_counts_afrc, MQNs_topology_counts_bfrc
3D descriptors	PBF, PMI1, PMI2, PMI3, NPR1, NPR2, RadiusOfGyration, InertialShapeFactor, Eccentricity, Asphericity, SphericityIndex

A.4 Gráficos para *Feature Importance*

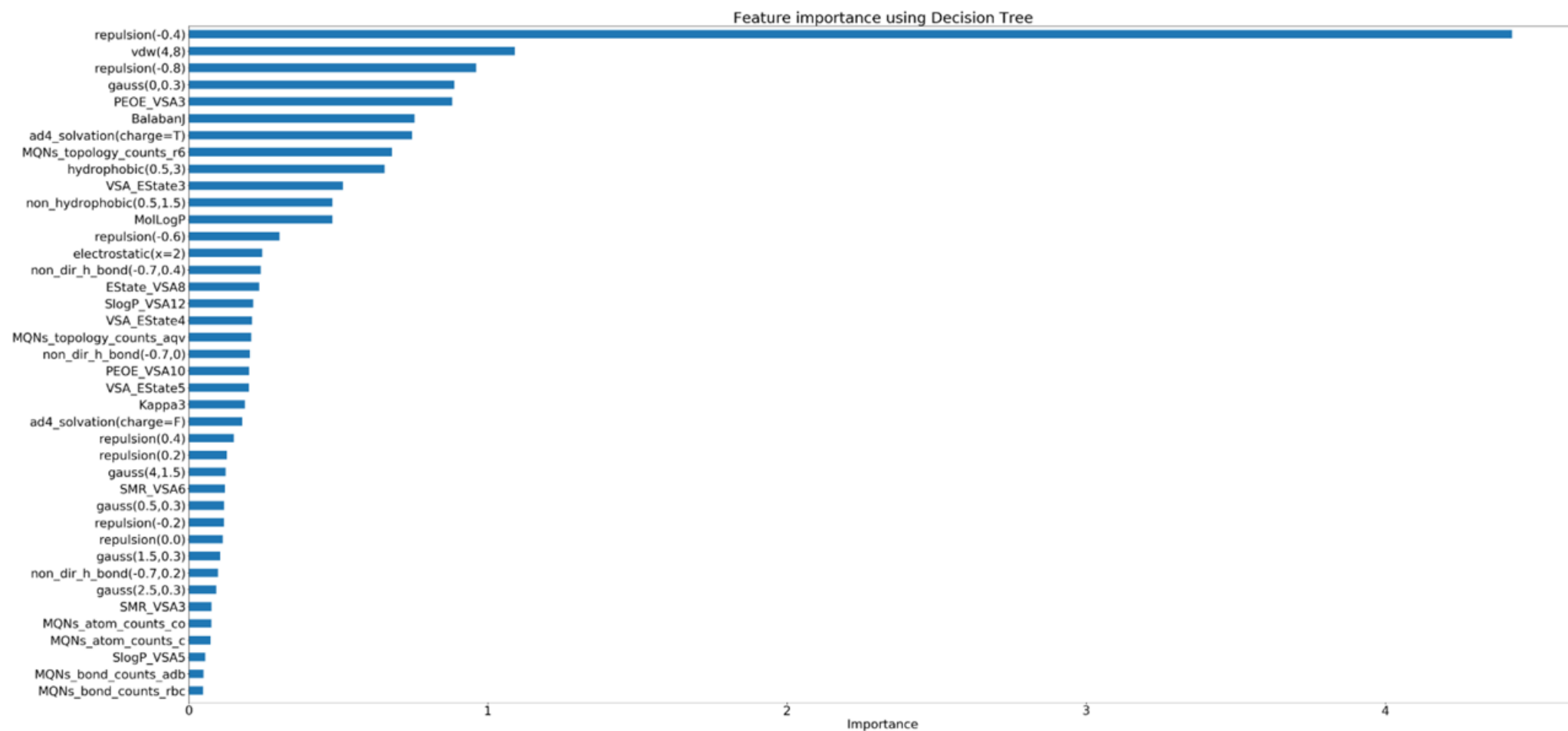


Figura 66: Medida de importância calculada pelo algoritmo *Decision Tree* para Δp_{k_d}

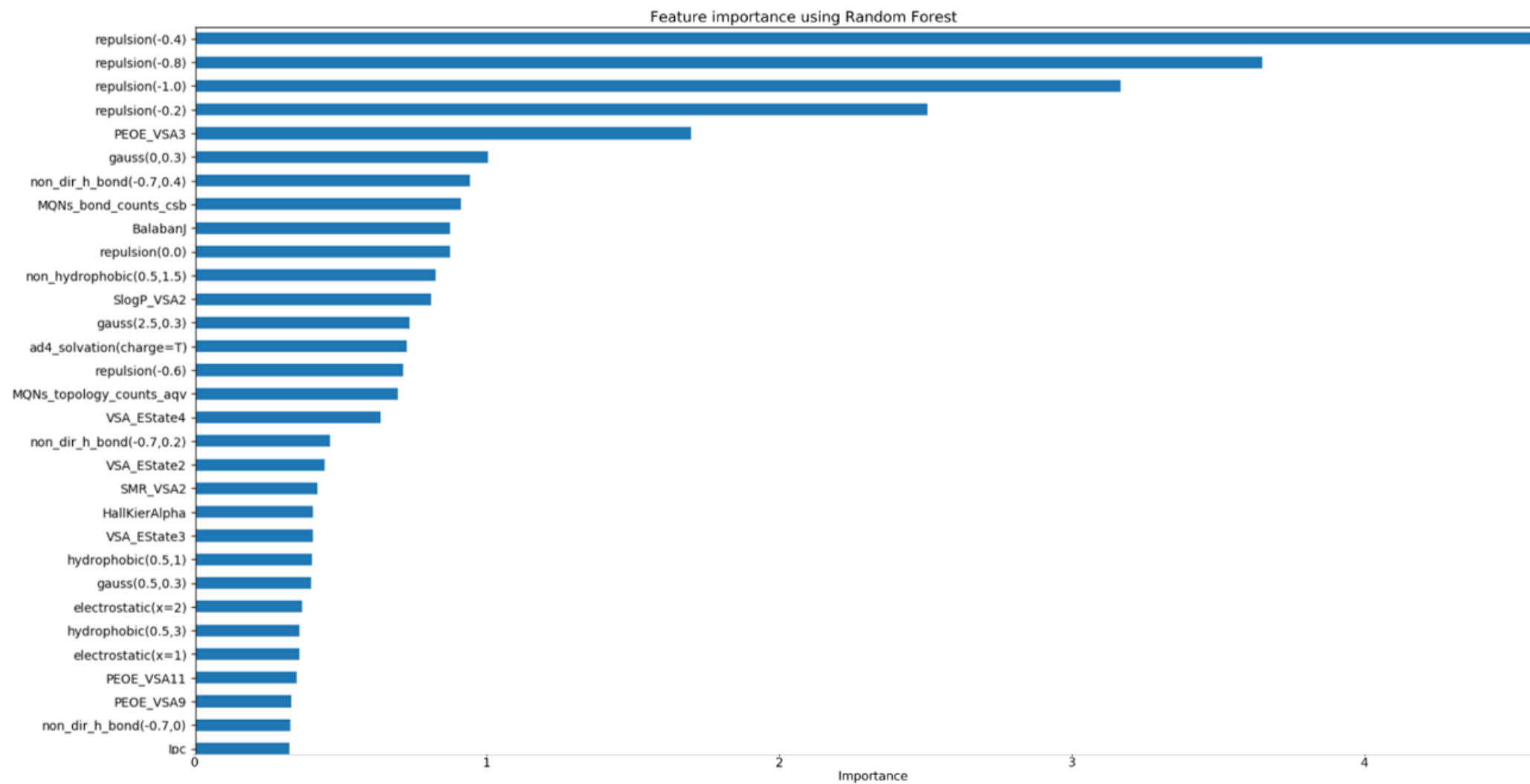


Figura 67: Medida de importância calculada pelo algoritmo *Random Forest* para Δp_{k_d}

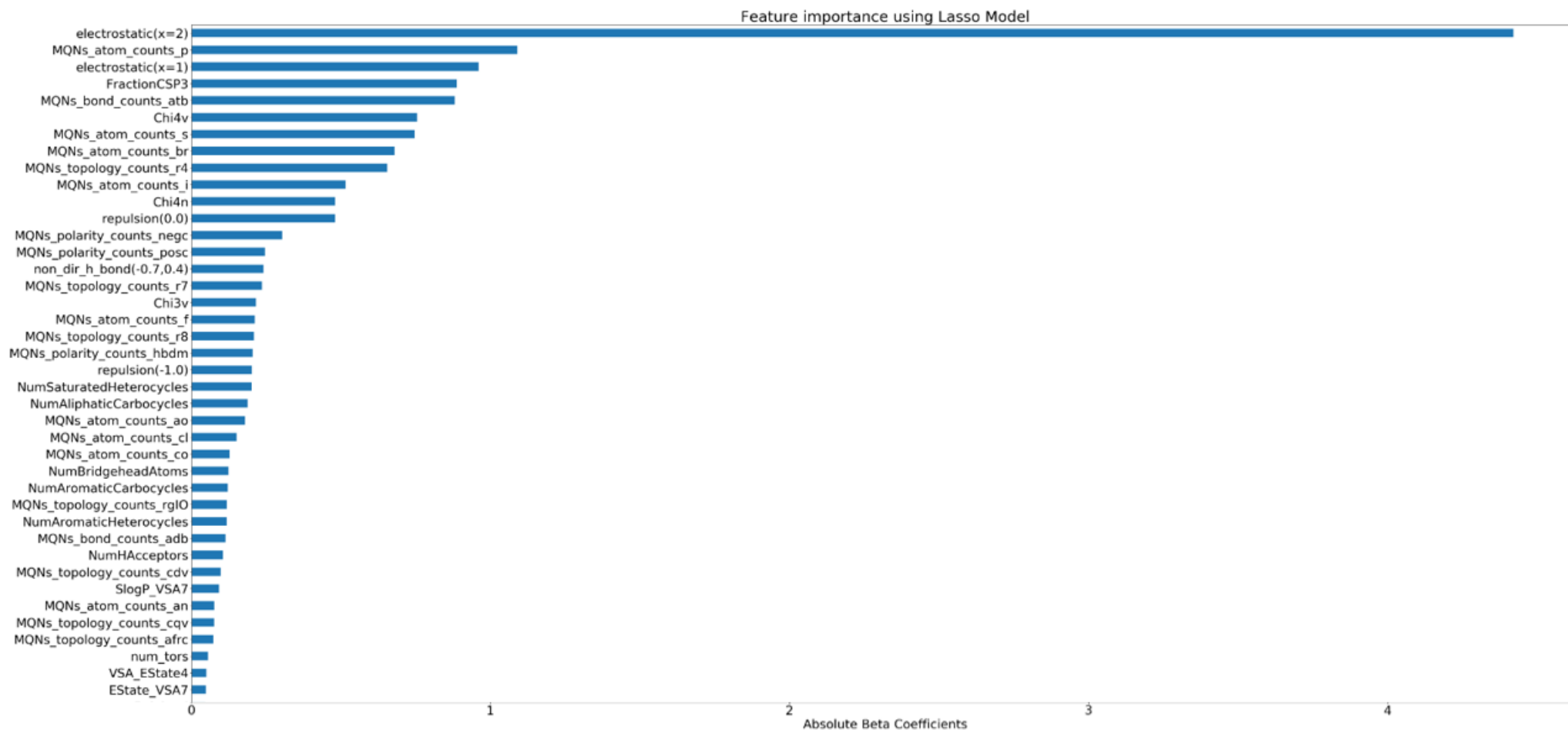


Figura 68: Medida de importância calculada pelo algoritmo *Lasso* para Δp_{k_d}

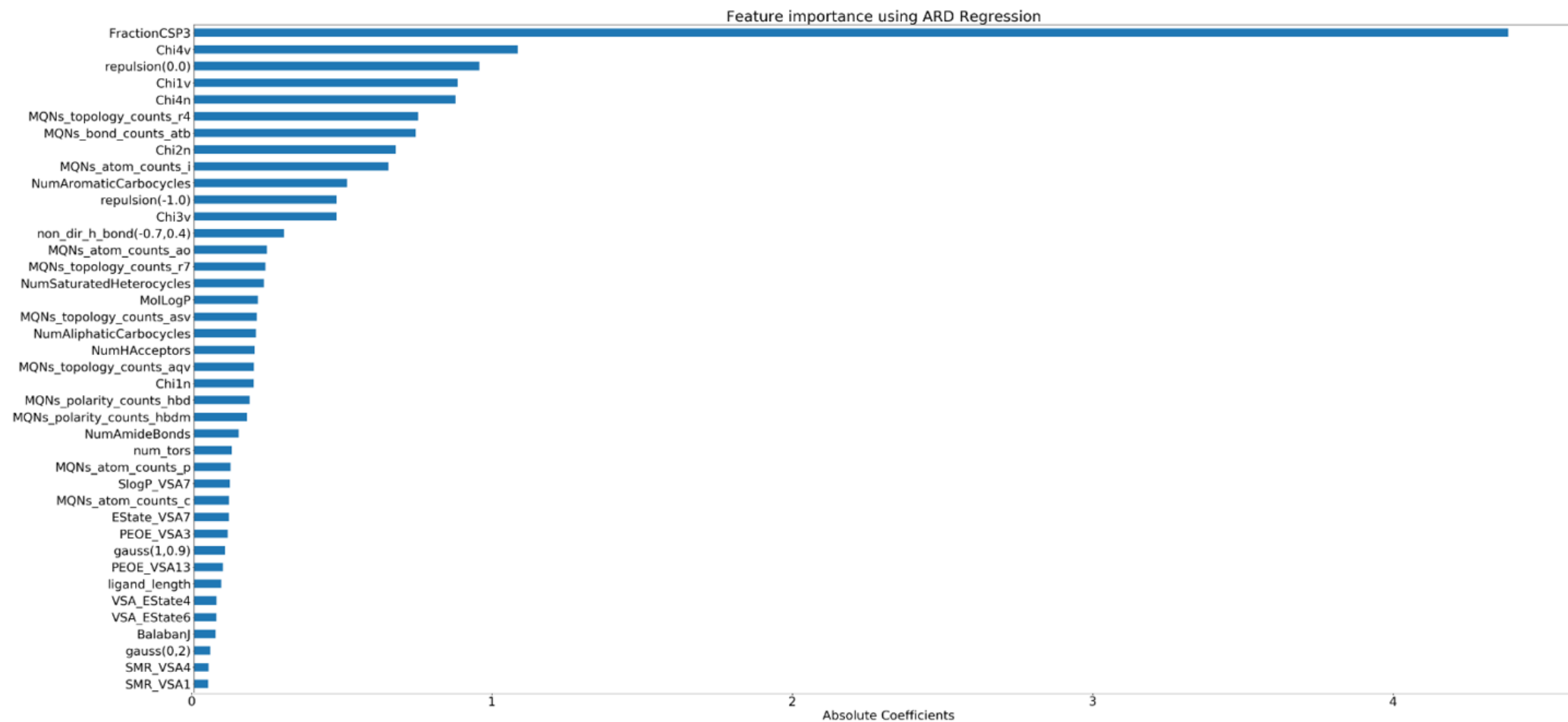


Figura 69: Medida de importância calculada pelo algoritmo ARD para Δp_{k_d}

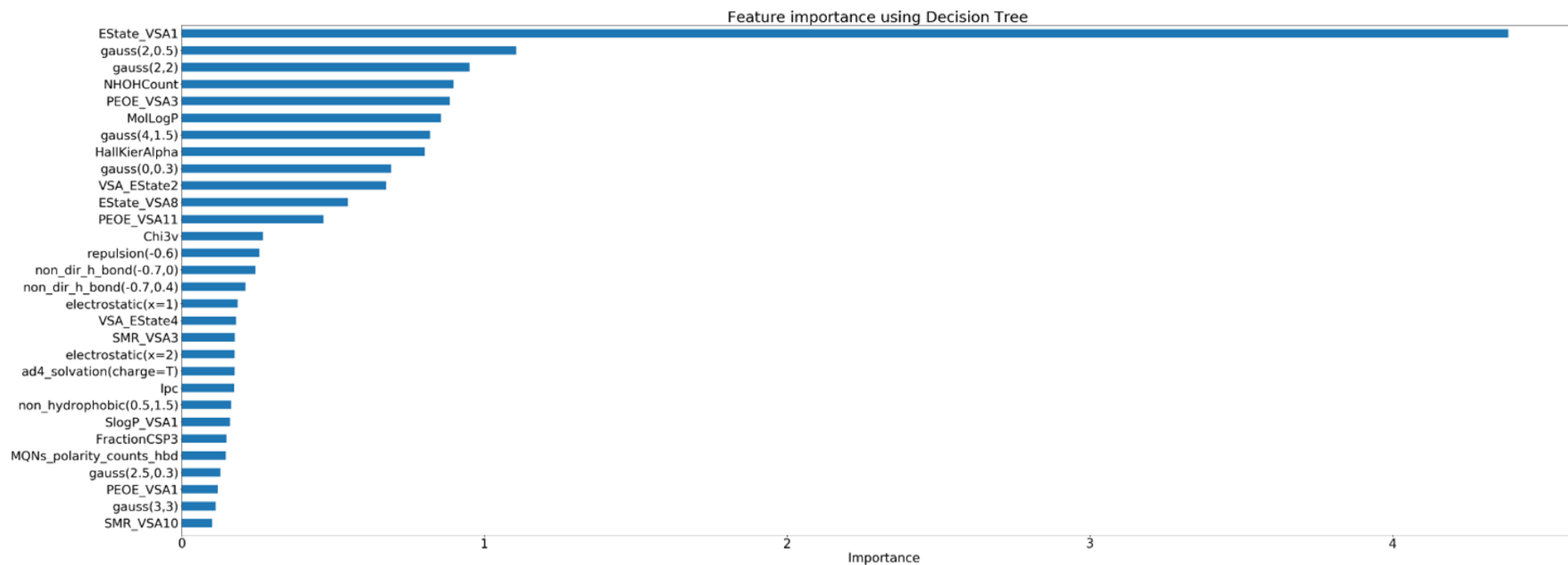


Figura 70: Medida de importância calculada pelo *Decision Tree* para p_{k_d}

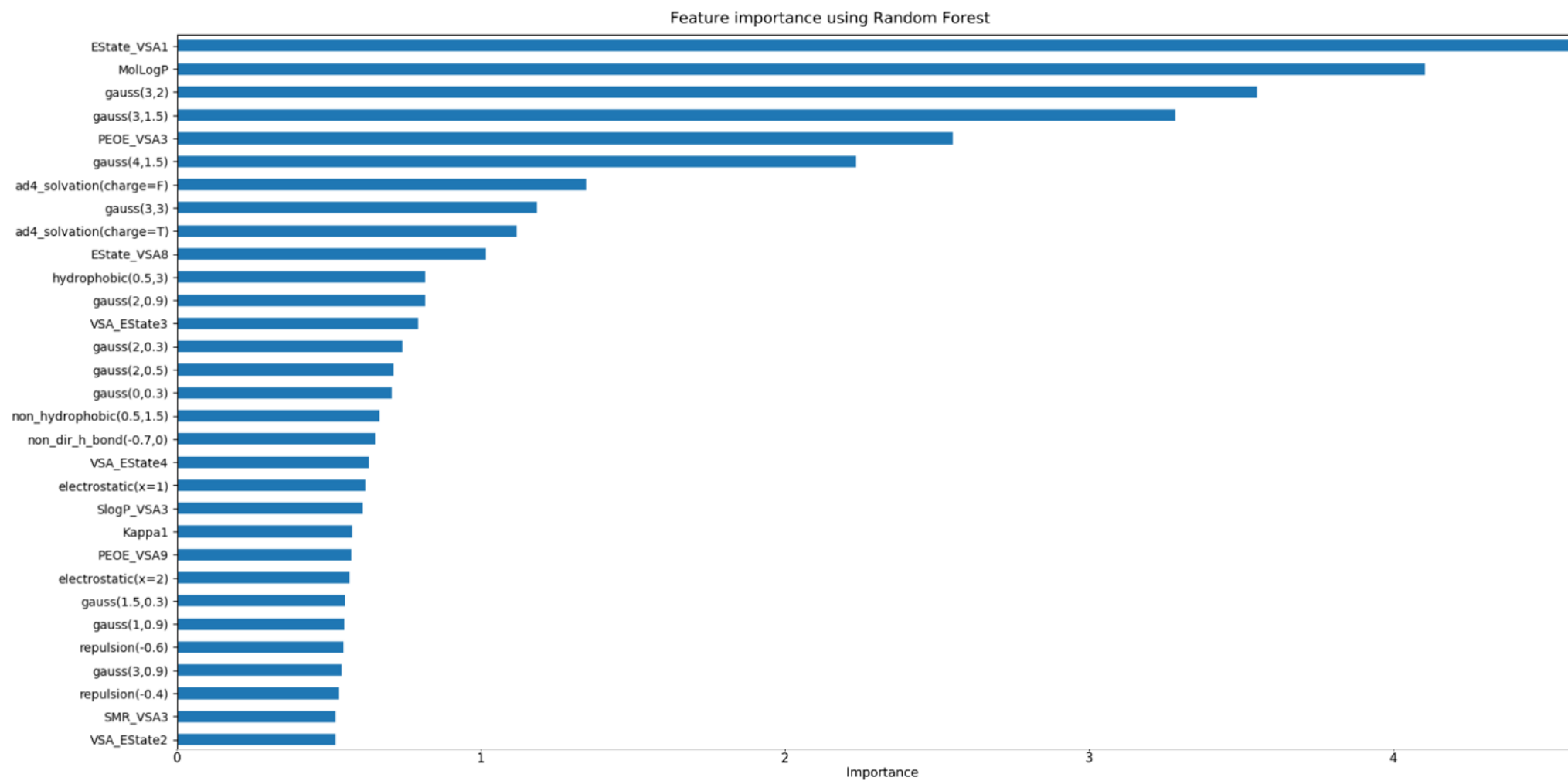


Figura 71: Medida de importância calculada pelo *Random Forest* para p_{k_d}

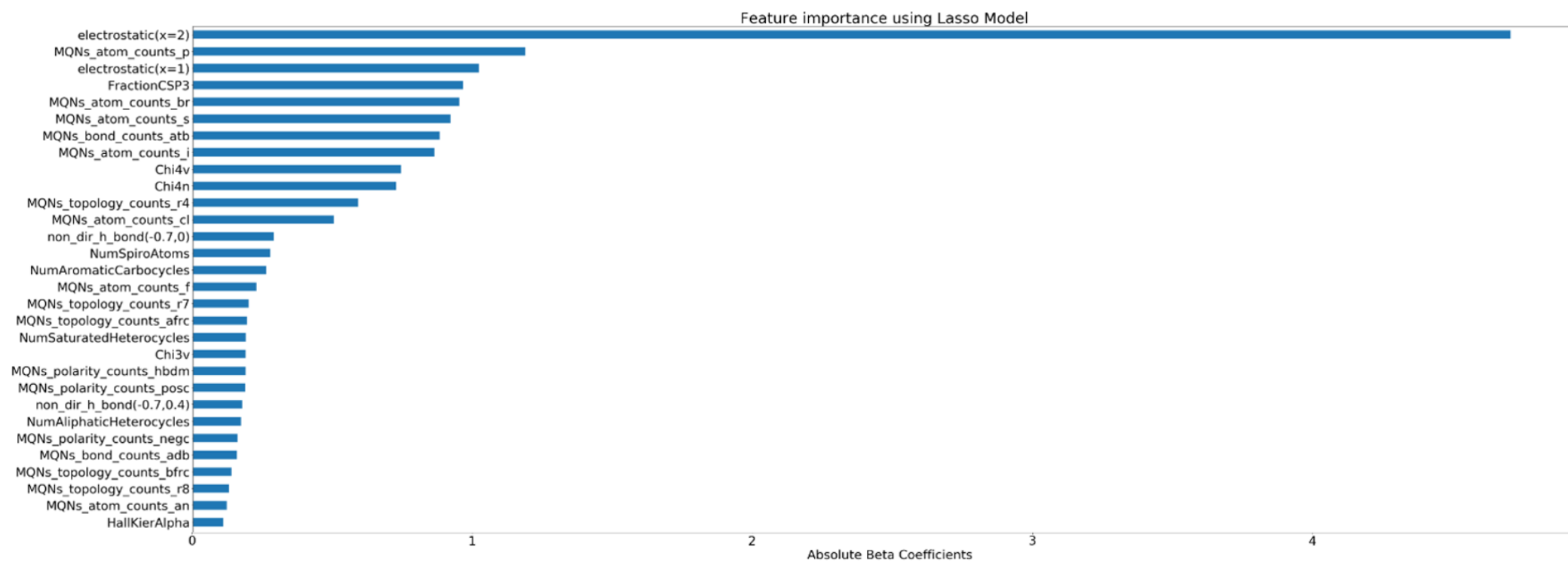


Figura 72: Medida de importância calculada pelo algoritmo *Lasso* para p_{k_d}

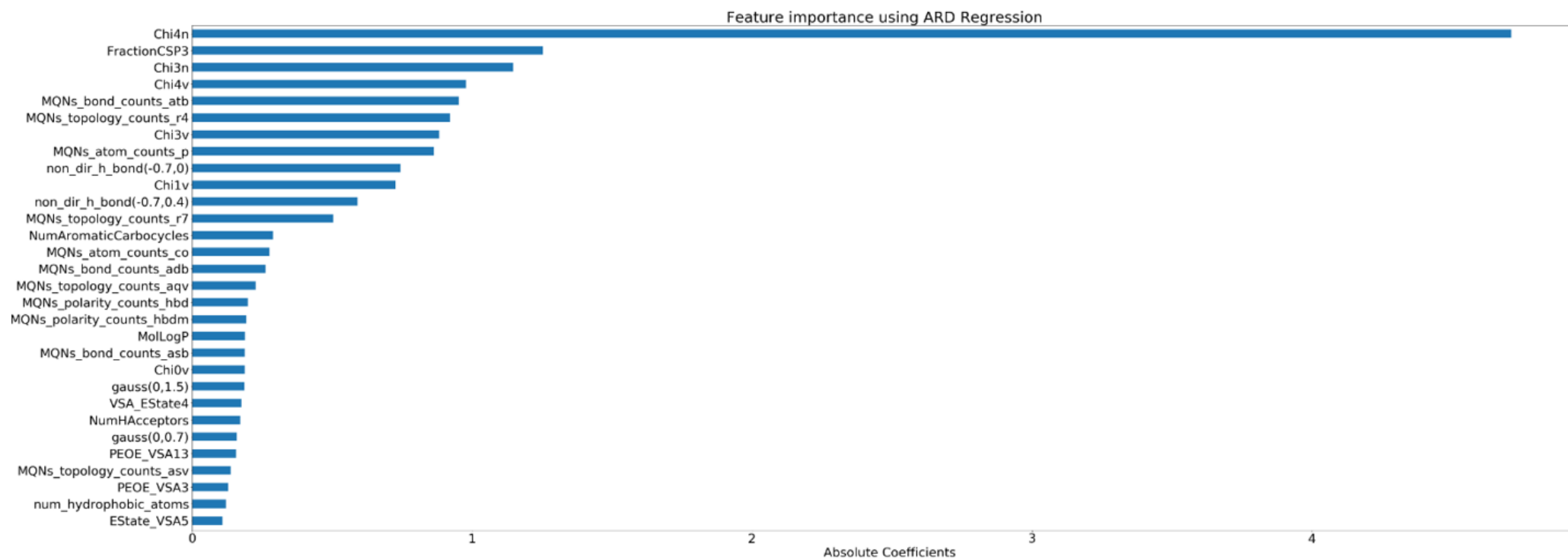


Figura 73: Medida de importância calculada pelo algoritmo ARD para p_{k_d}

A.5 Gráficos com a comparação das predições para cada método individualmente combinado a cada grupo de atributos

Pearson correlation vs. Spearman rank correlation - 5 Features

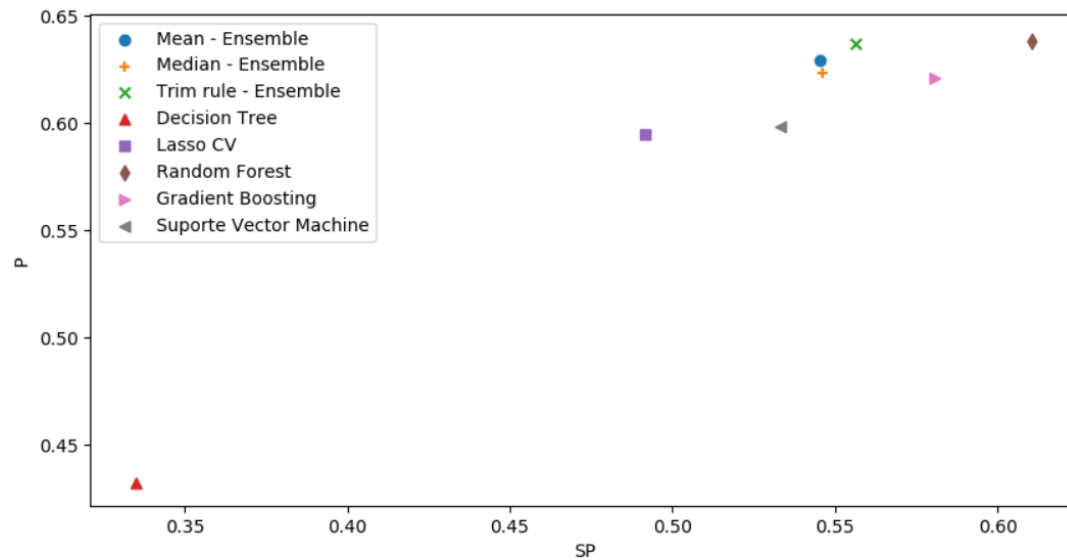


Figura 74: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{kd} - 5 melhores *features*

Pearson correlation vs. Spearman rank correlation - 10 Features

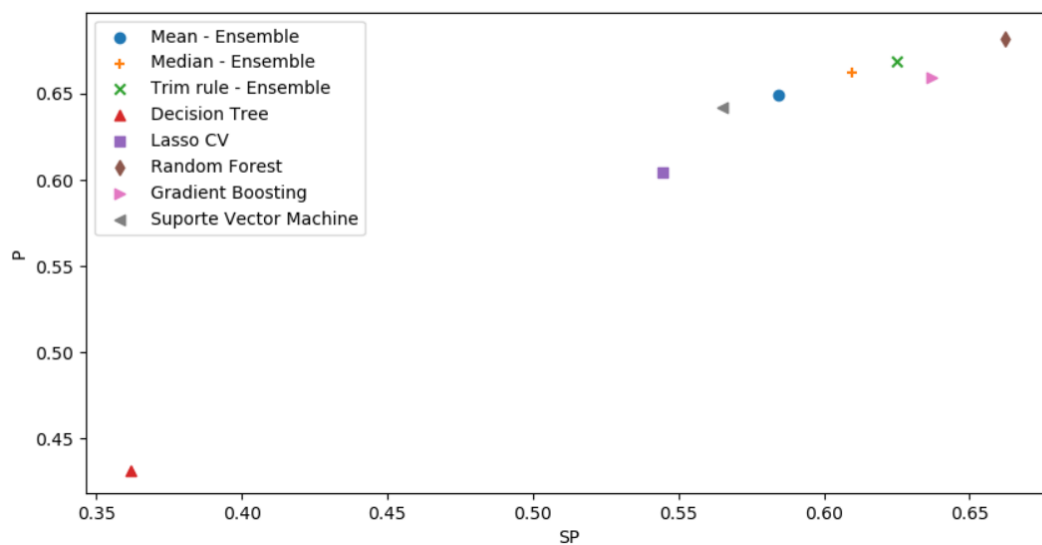


Figura 75: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{kd} - 10 melhores *features*

Pearson correlation vs. Spearman rank correlation - 15 Features

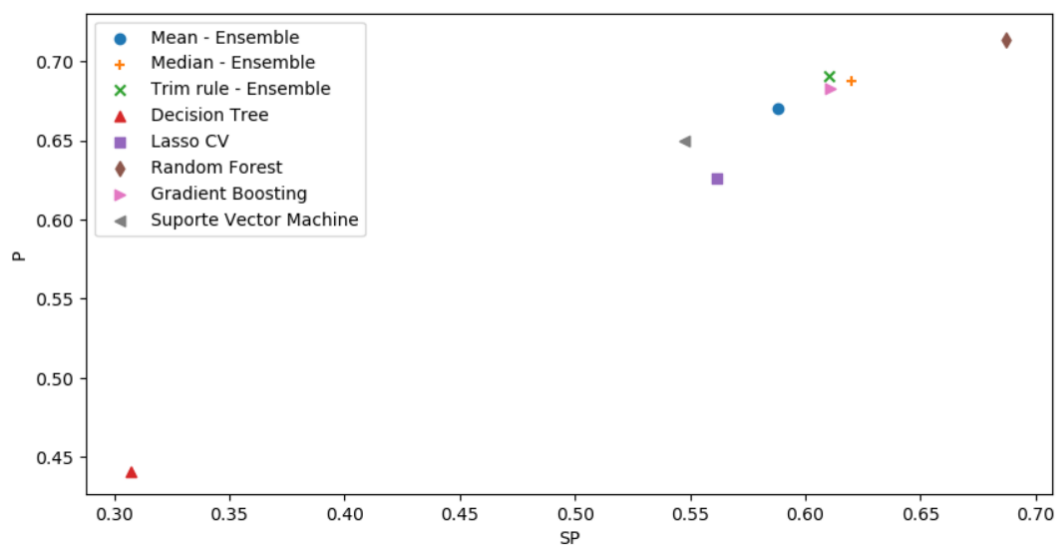


Figura 76: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 15 melhores *features*

Pearson correlation vs. Spearman rank correlation - 20 Features

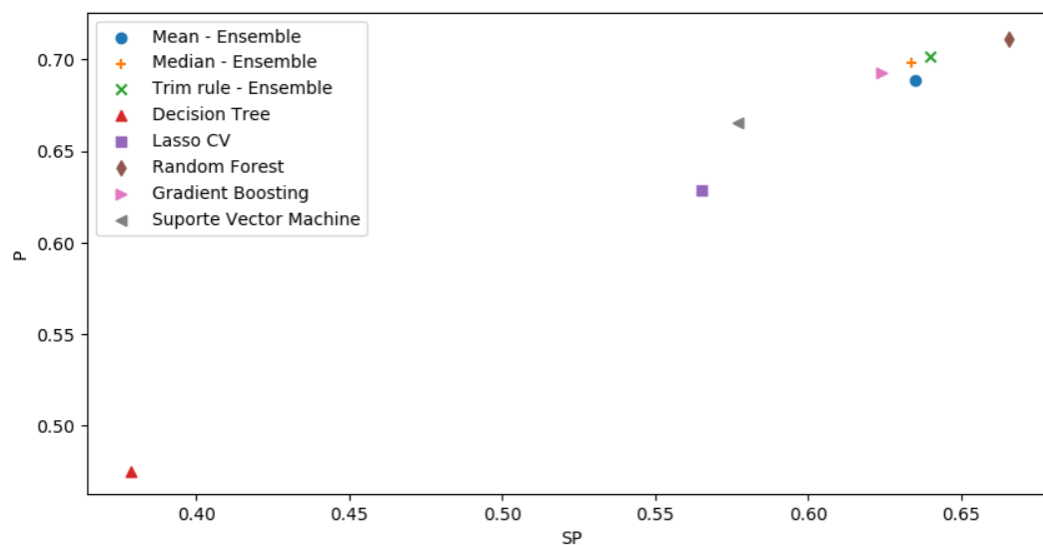


Figura 77: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 20 melhores *features*

Pearson correlation vs. Spearman rank correlation - 30 Features

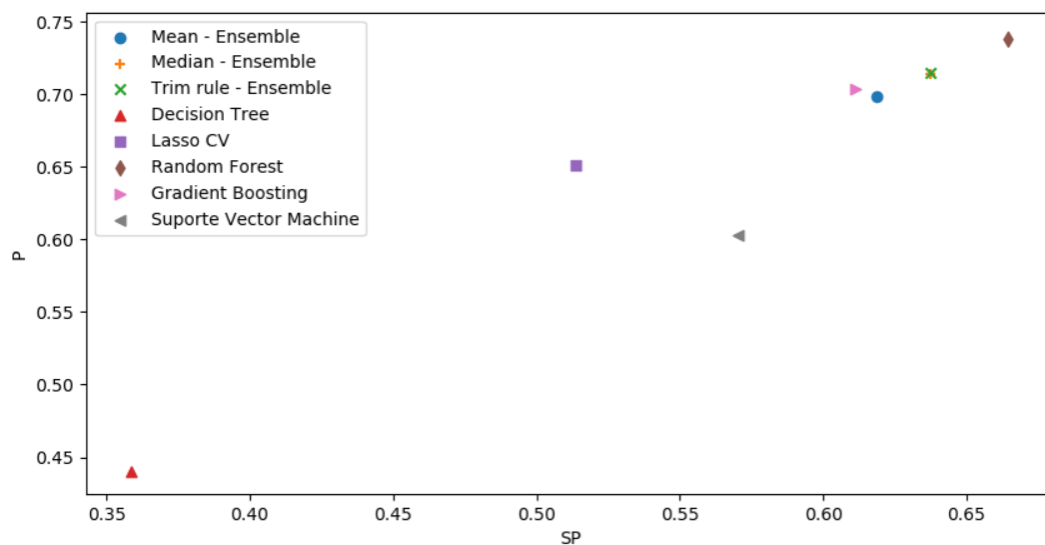


Figura 78: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 30 melhores *features*

Pearson correlation vs. Spearman rank correlation - 40 Features

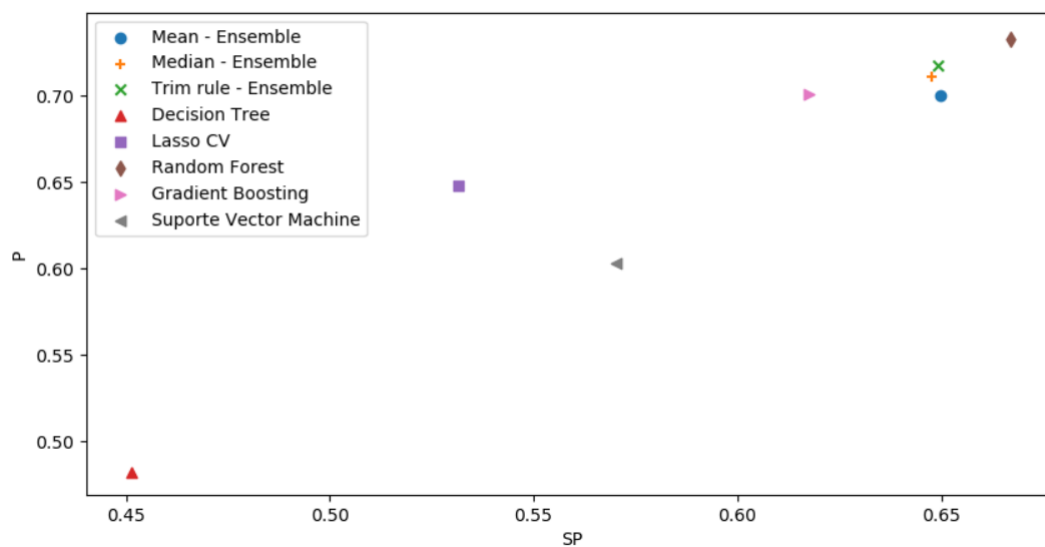


Figura 79: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 40 melhores *features*

Pearson correlation vs. Spearman rank correlation - 50 Features

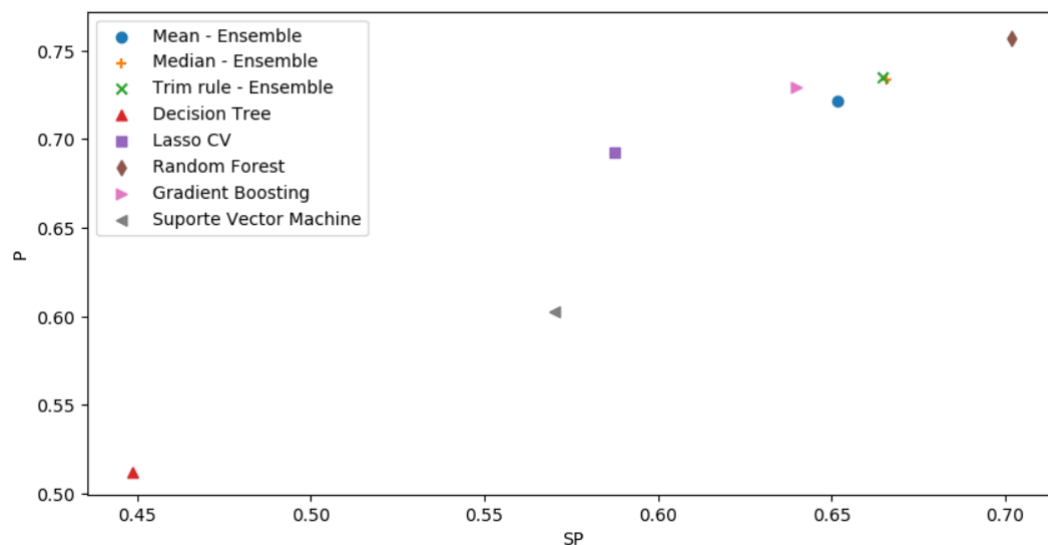


Figura 80: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 50 melhores *features*

Pearson correlation vs. Spearman rank correlation - 70 Features

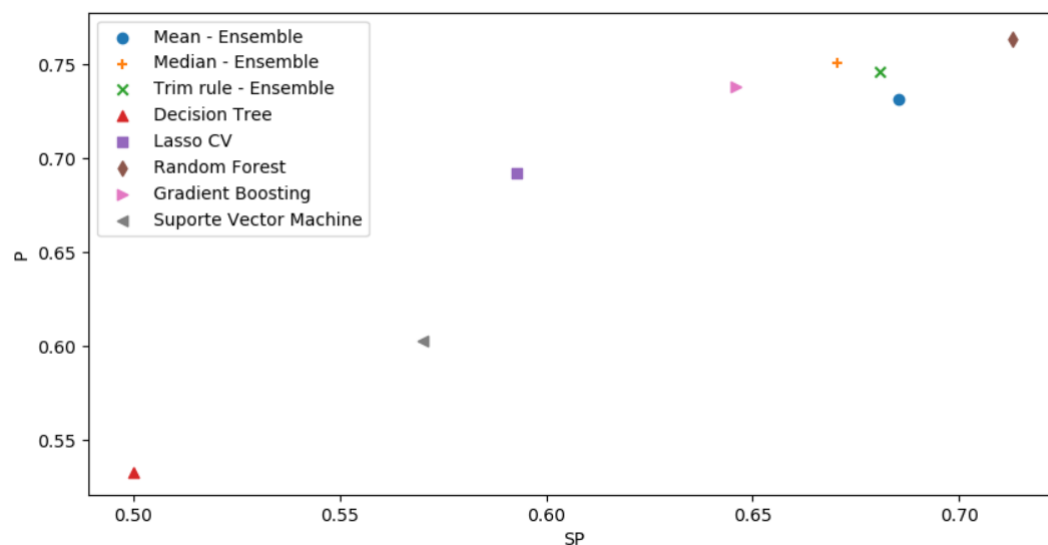


Figura 81: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 70 melhores *features*

Pearson correlation vs. Spearman rank correlation - 80 Features

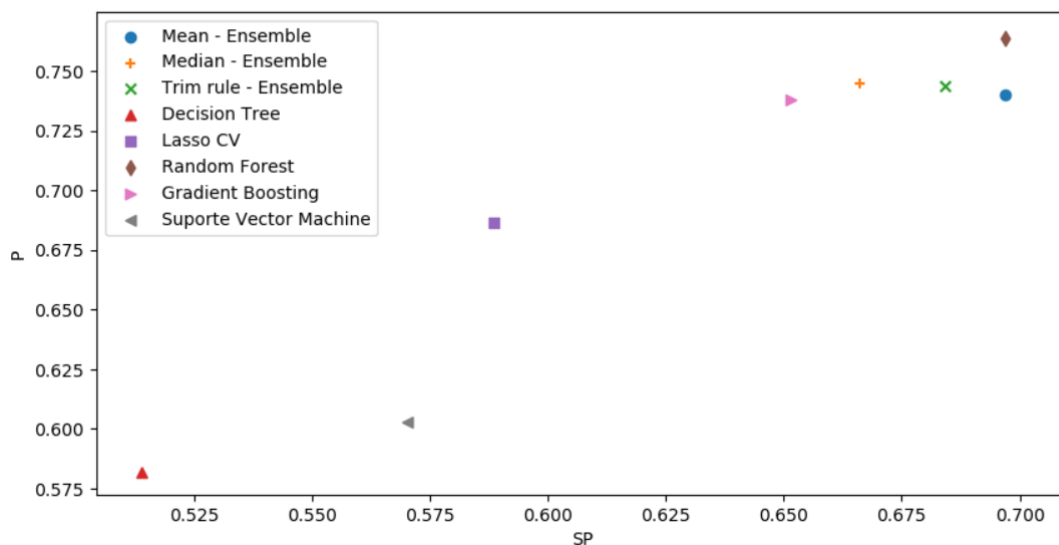


Figura 82: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 80 melhores *features*

Pearson correlation vs. Spearman rank correlation - 90 Features

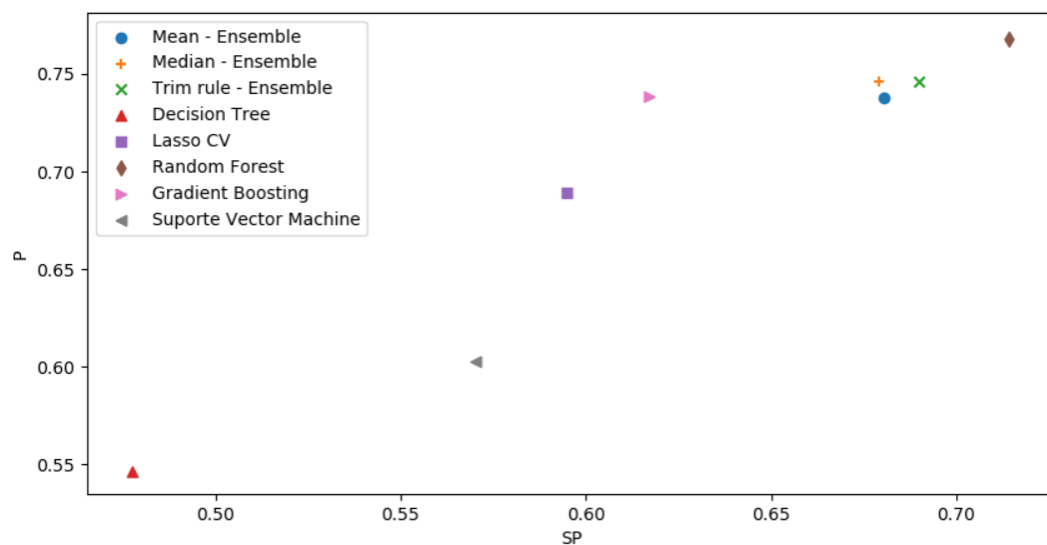


Figura 83: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 90 melhores *features*

Pearson correlation vs. Spearman rank correlation - 100 Features

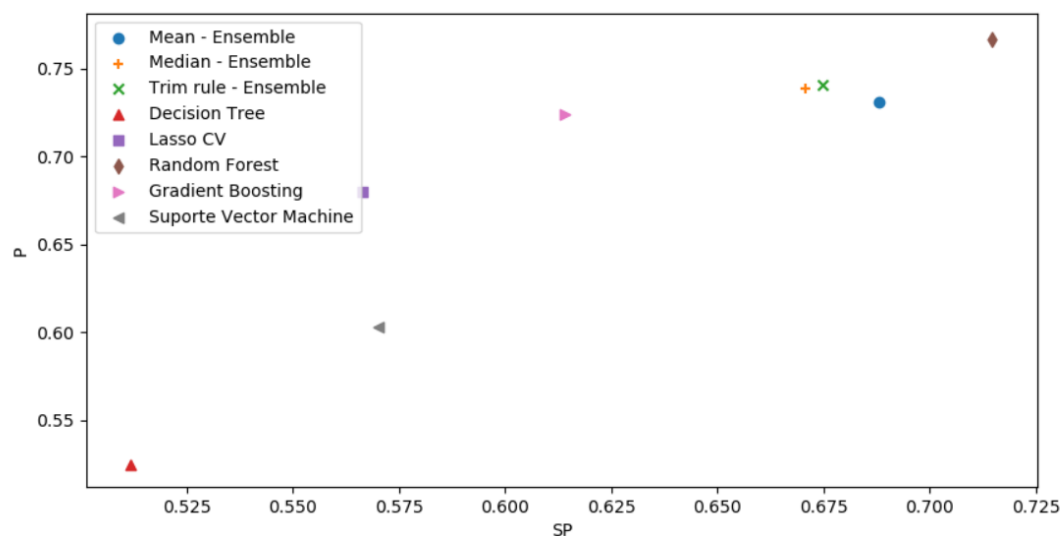


Figura 84: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 100 melhores *features*

Pearson correlation vs. Spearman rank correlation - 150 Features

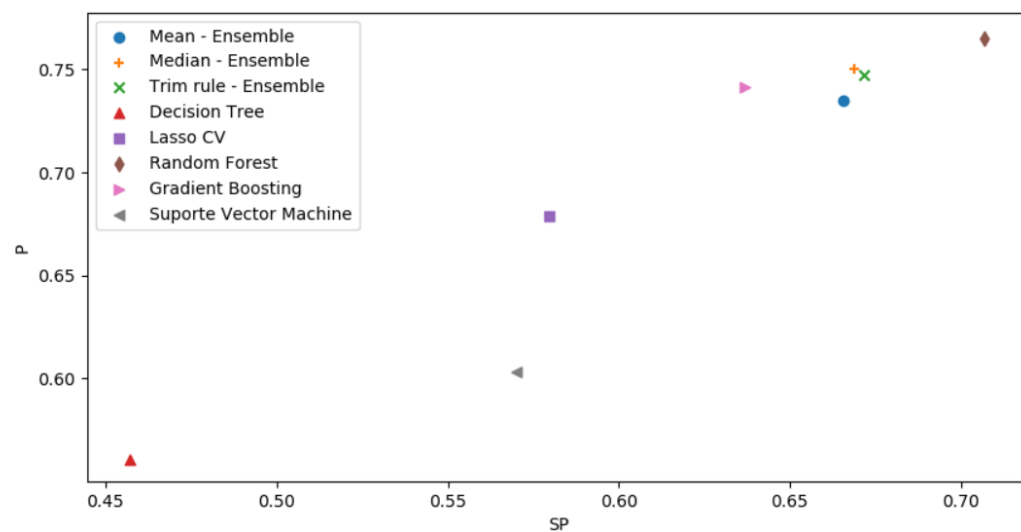


Figura 85: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 150 melhores *features*

Pearson correlation vs. Spearman rank correlation - 200 Features

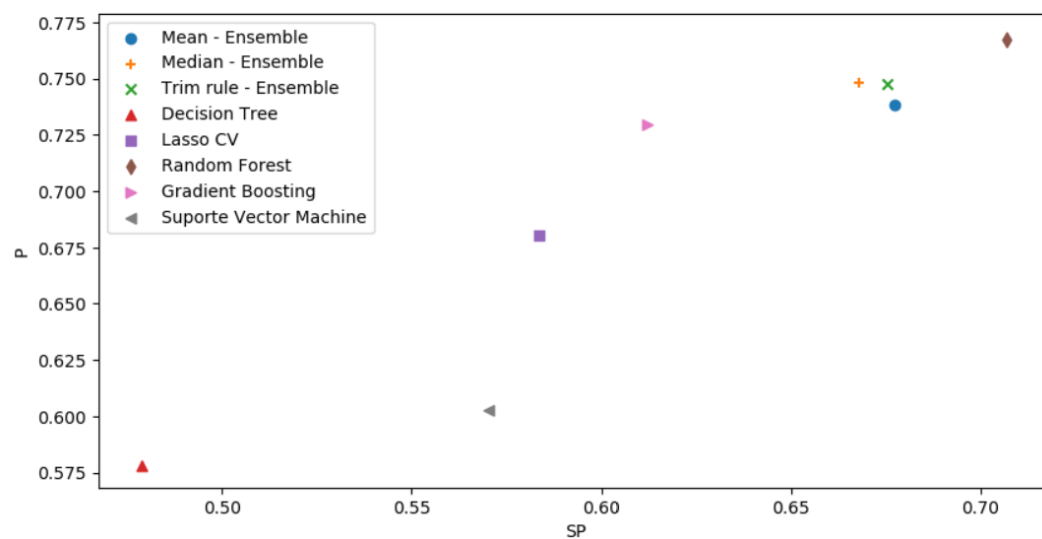


Figura 86: Comparação do *Ensemble* com os métodos individualmente para a predição de Δp_{k_d} - 200 melhores *features*

Pearson correlation vs. Spearman rank correlation - 5 Features

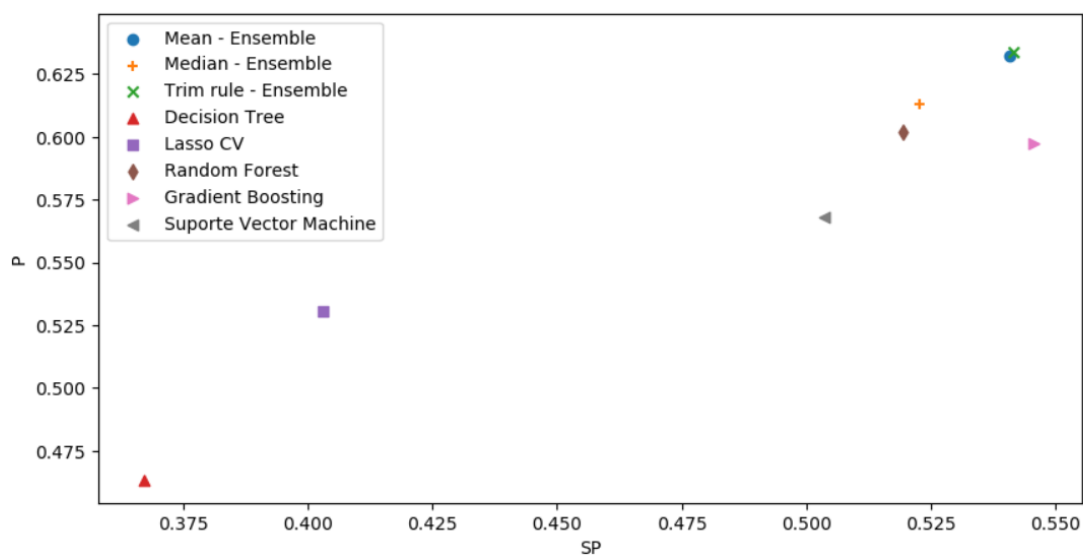


Figura 87: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 5 melhores *features*

Pearson correlation vs. Spearman rank correlation - 10 Features

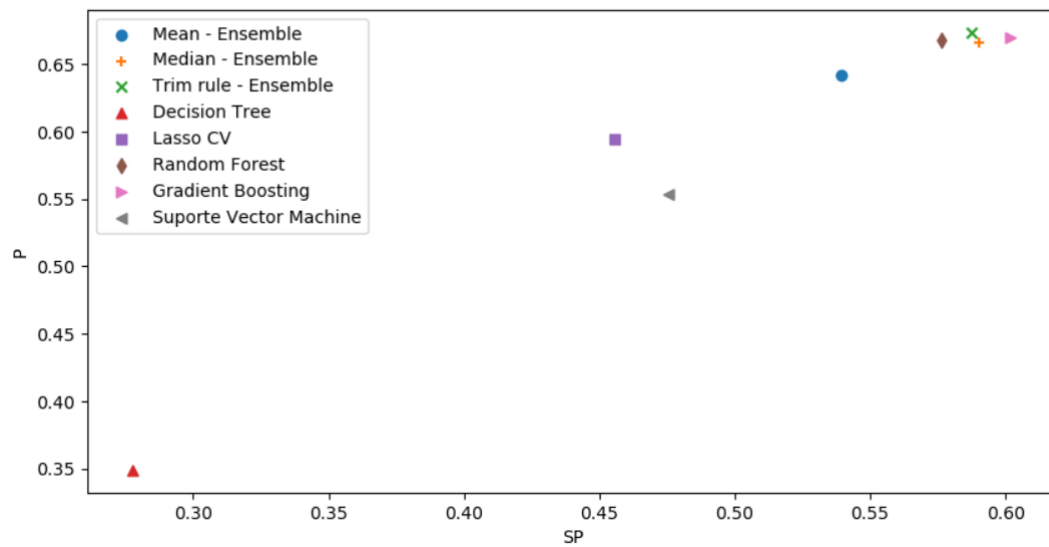


Figura 88: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 10 melhores *features*

Pearson correlation vs. Spearman rank correlation - 15 Features

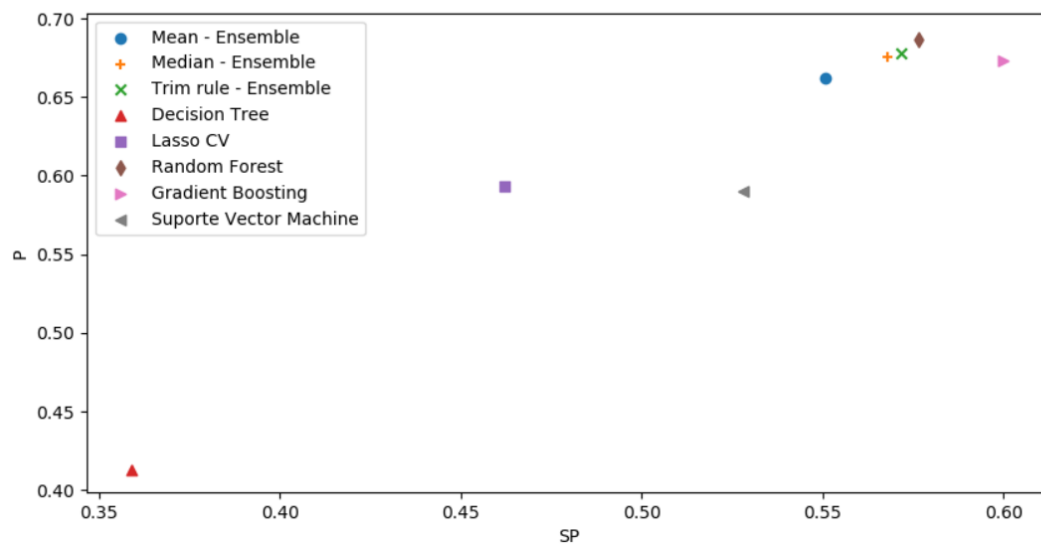


Figura 89: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 15 melhores *features*

Pearson correlation vs. Spearman rank correlation - 20 Features

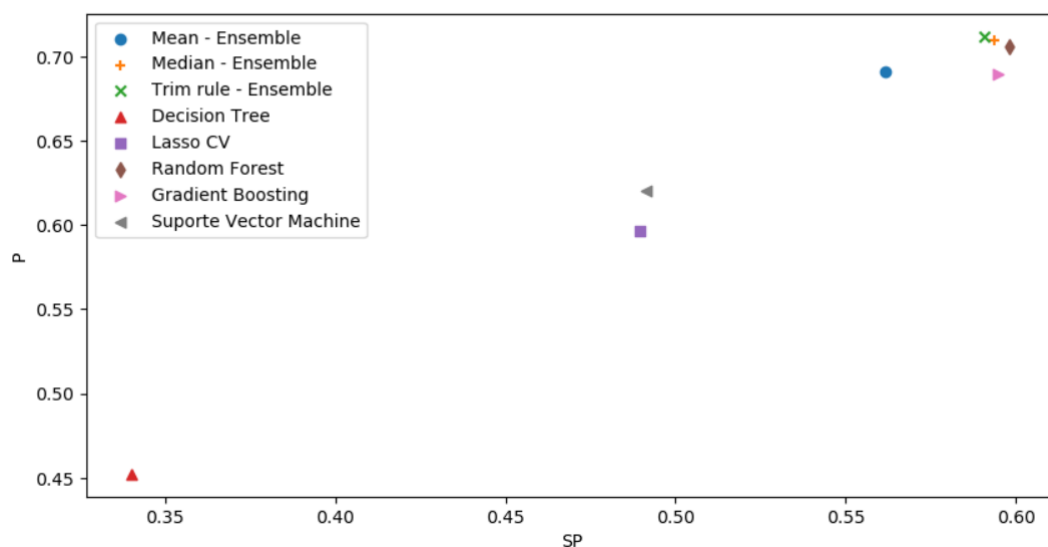


Figura 90: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 20 melhores *features*

Pearson correlation vs. Spearman rank correlation - 30 Features

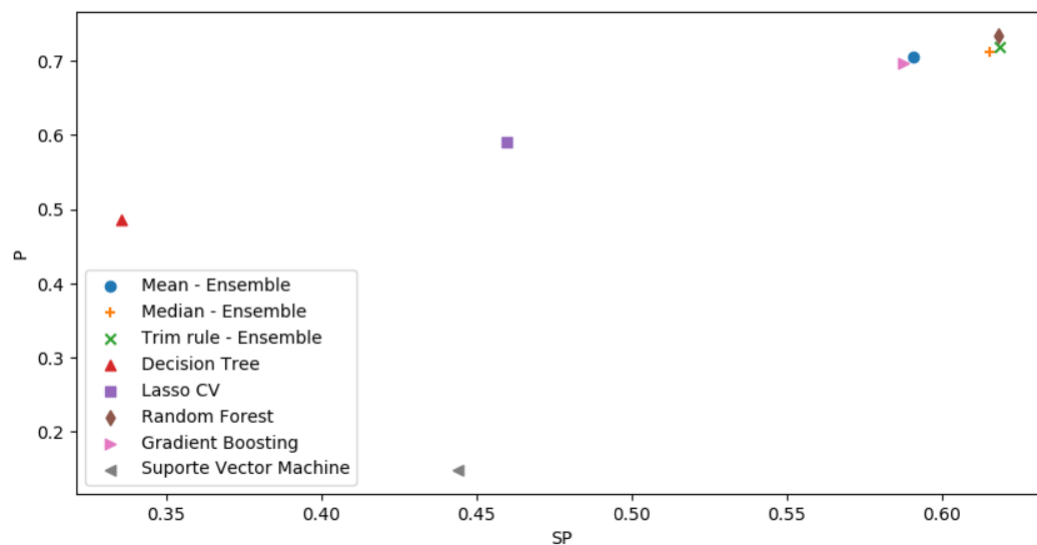


Figura 91: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 30 melhores *features*

Pearson correlation vs. Spearman rank correlation - 40 Features

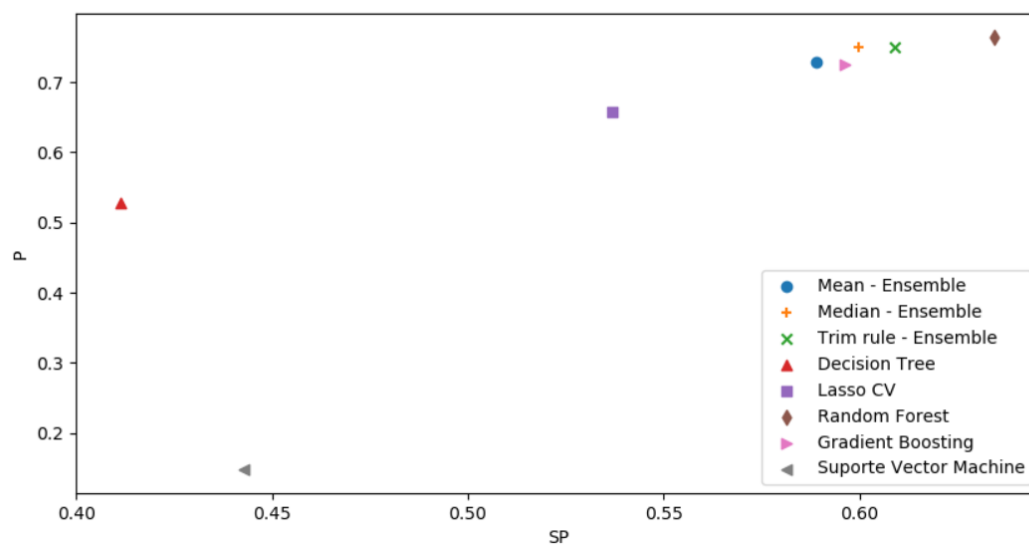


Figura 92: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 40 melhores *features*

Pearson correlation vs. Spearman rank correlation - 50 Features

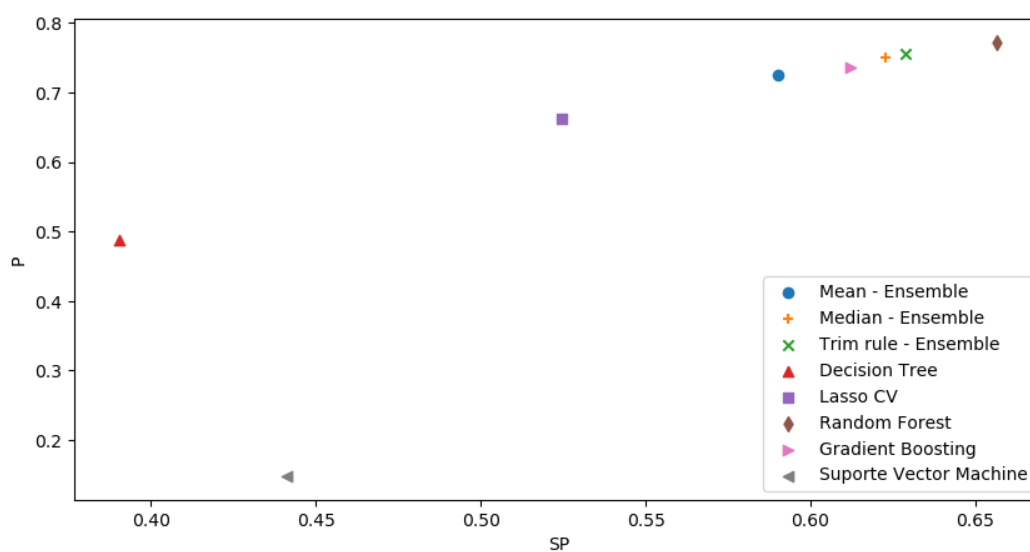


Figura 93: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 50 melhores *features*

Pearson correlation vs. Spearman rank correlation - 70 Features

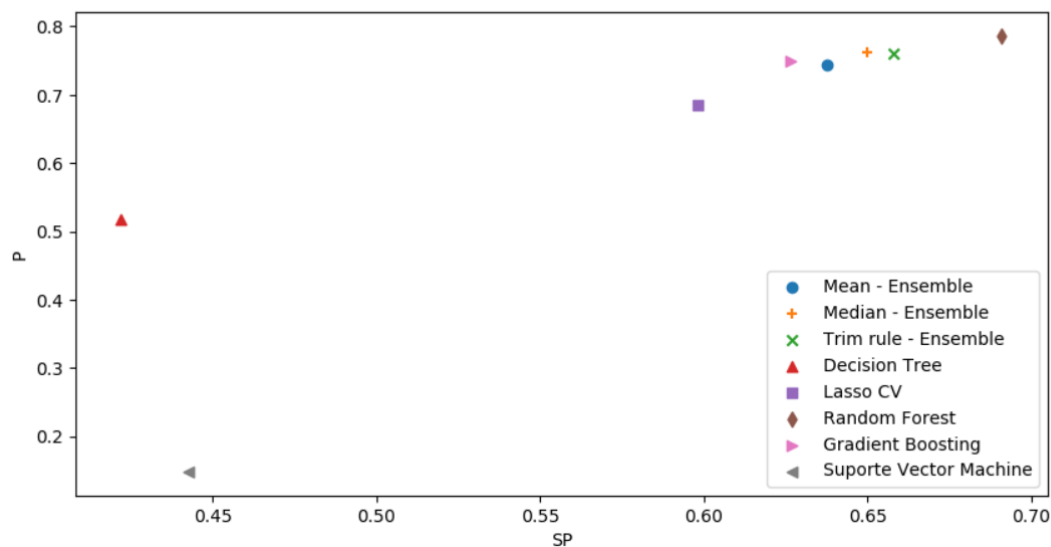


Figura 94: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 70 melhores *features*

Pearson correlation vs. Spearman rank correlation - 80 Features

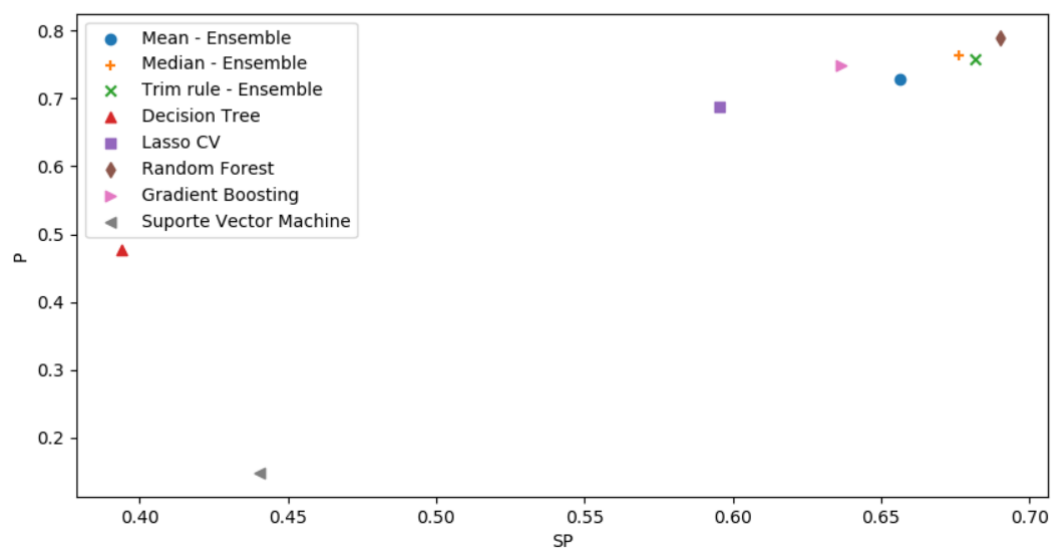


Figura 95: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 80 melhores *features*

Pearson correlation vs. Spearman rank correlation - 90 Features

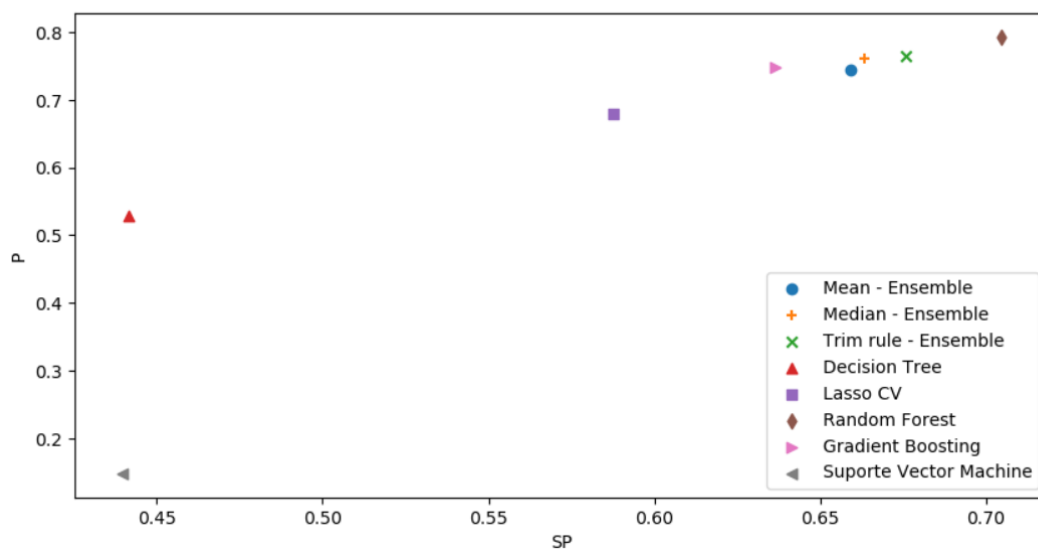


Figura 96: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 90 melhores *features*

Pearson correlation vs. Spearman rank correlation - 100 Features

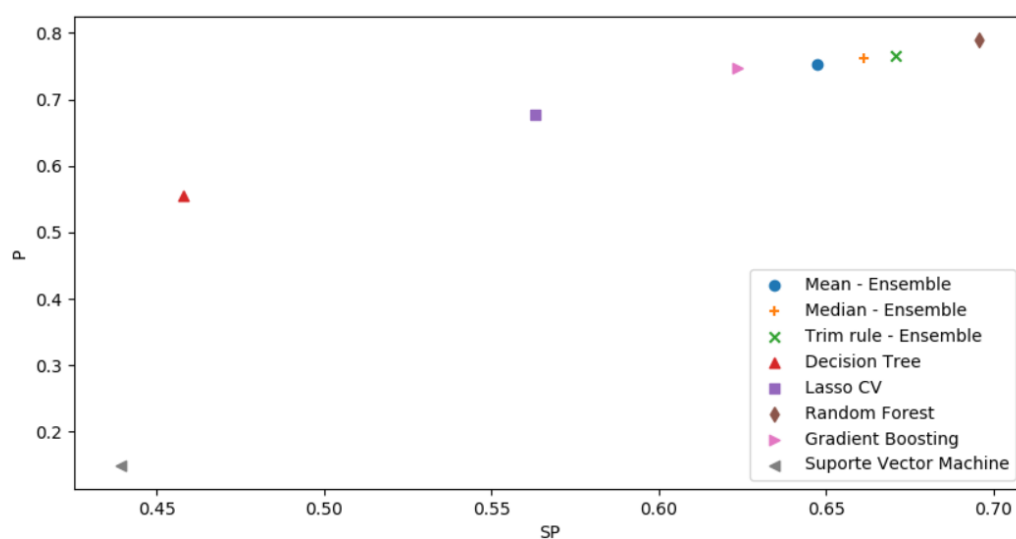


Figura 97: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 100 melhores *features*

Pearson correlation vs. Spearman rank correlation - 150 Features

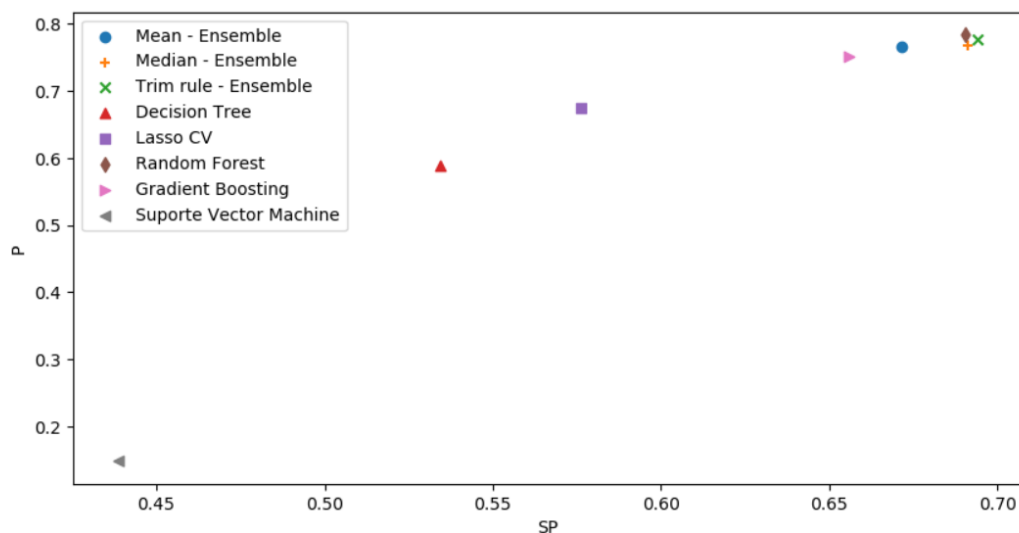


Figura 98: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 150 melhores *features*

Pearson correlation vs. Spearman rank correlation - 200 Features

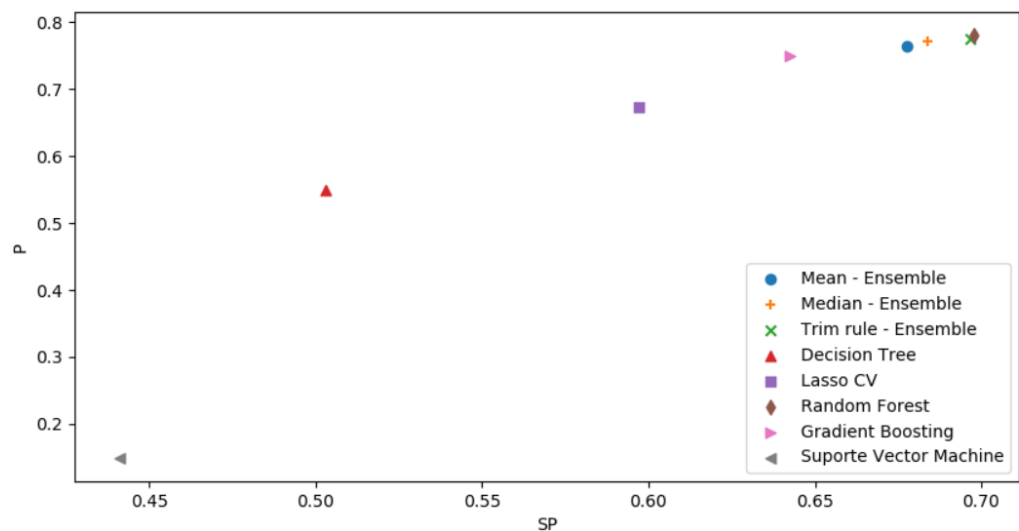


Figura 99: Comparação do *Ensemble* com os métodos individualmente para a predição de p_{k_d} - 200 melhores *features*

A.6 Gráficos para *Feature Importance* com otimização de parâmetros

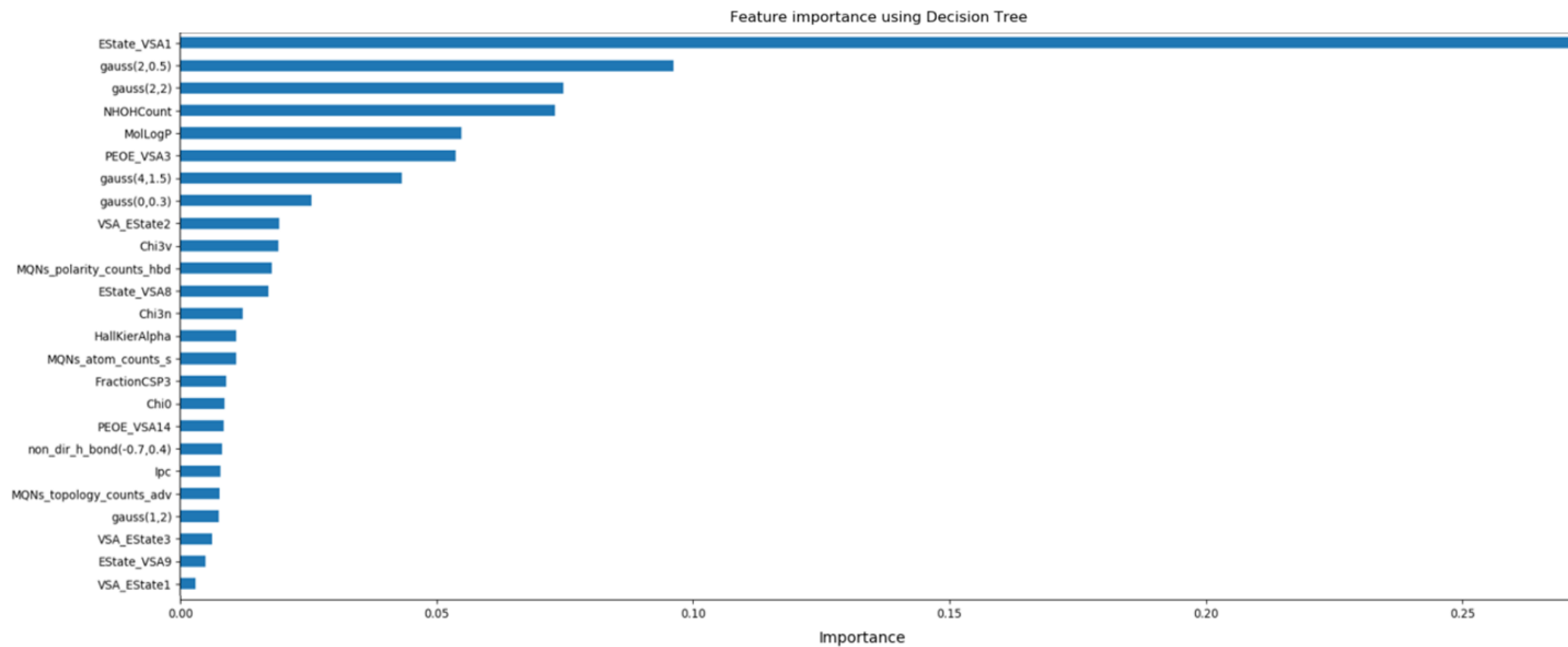


Figura 100: Medida de importância calculada pelo algoritmo *Decision Tree* para p_{k_d}

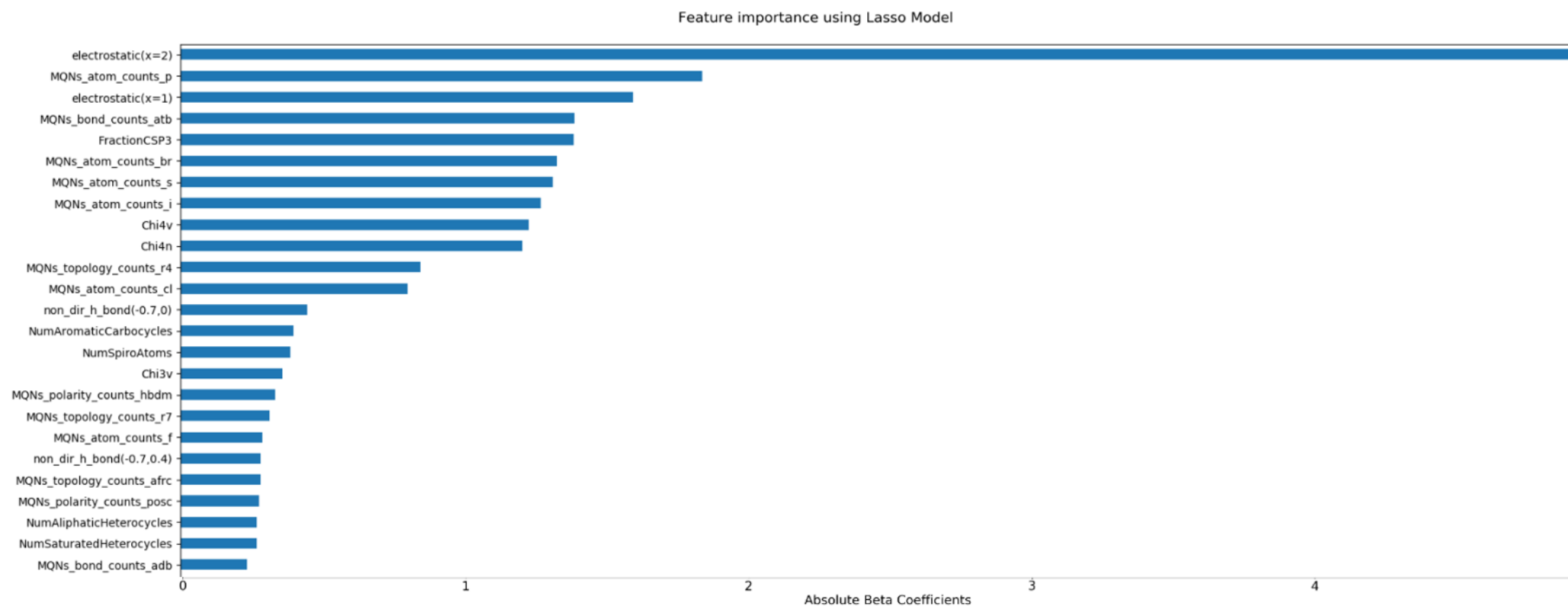


Figura 101: Medida de importância calculada pelo algoritmo *Lasso CV* para p_{k_d}

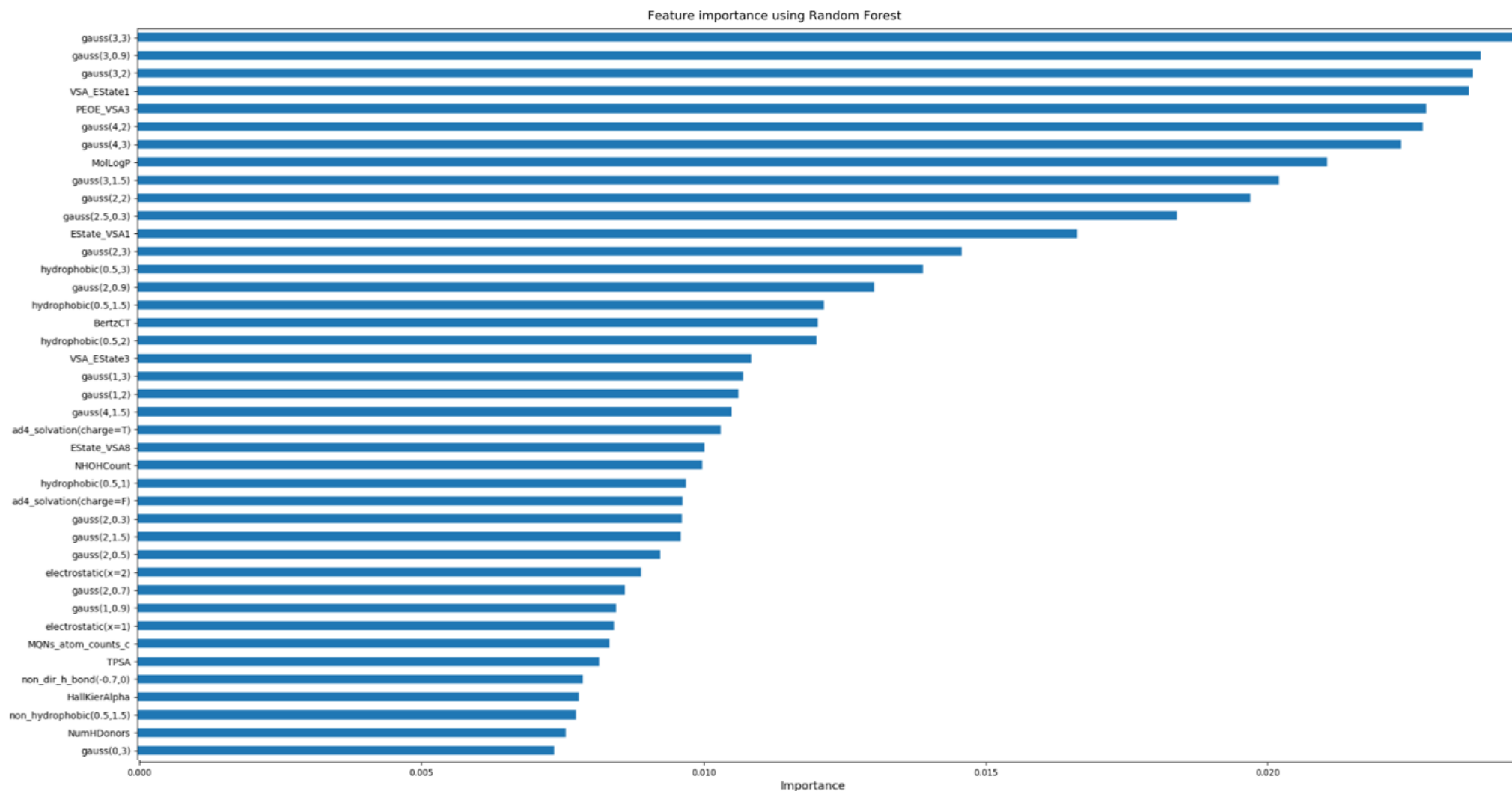


Figura 102: Medida de importância calculada pelo algoritmo *Random Forest* para p_{k_d} (1 a 41)

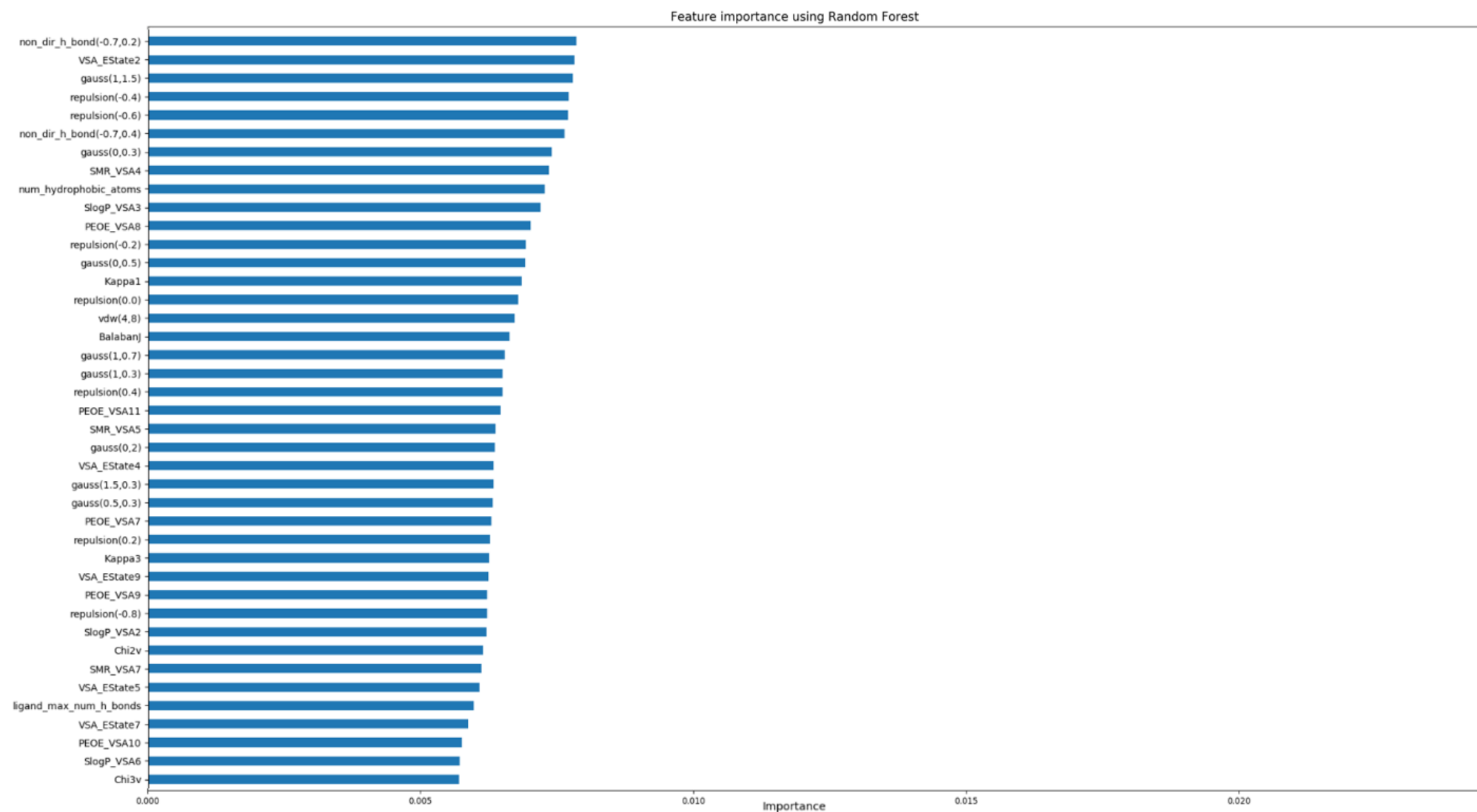


Figura 103: Medida de importância calculada pelo algoritmo *Random Forest* para p_{k_d} (41 a 82)

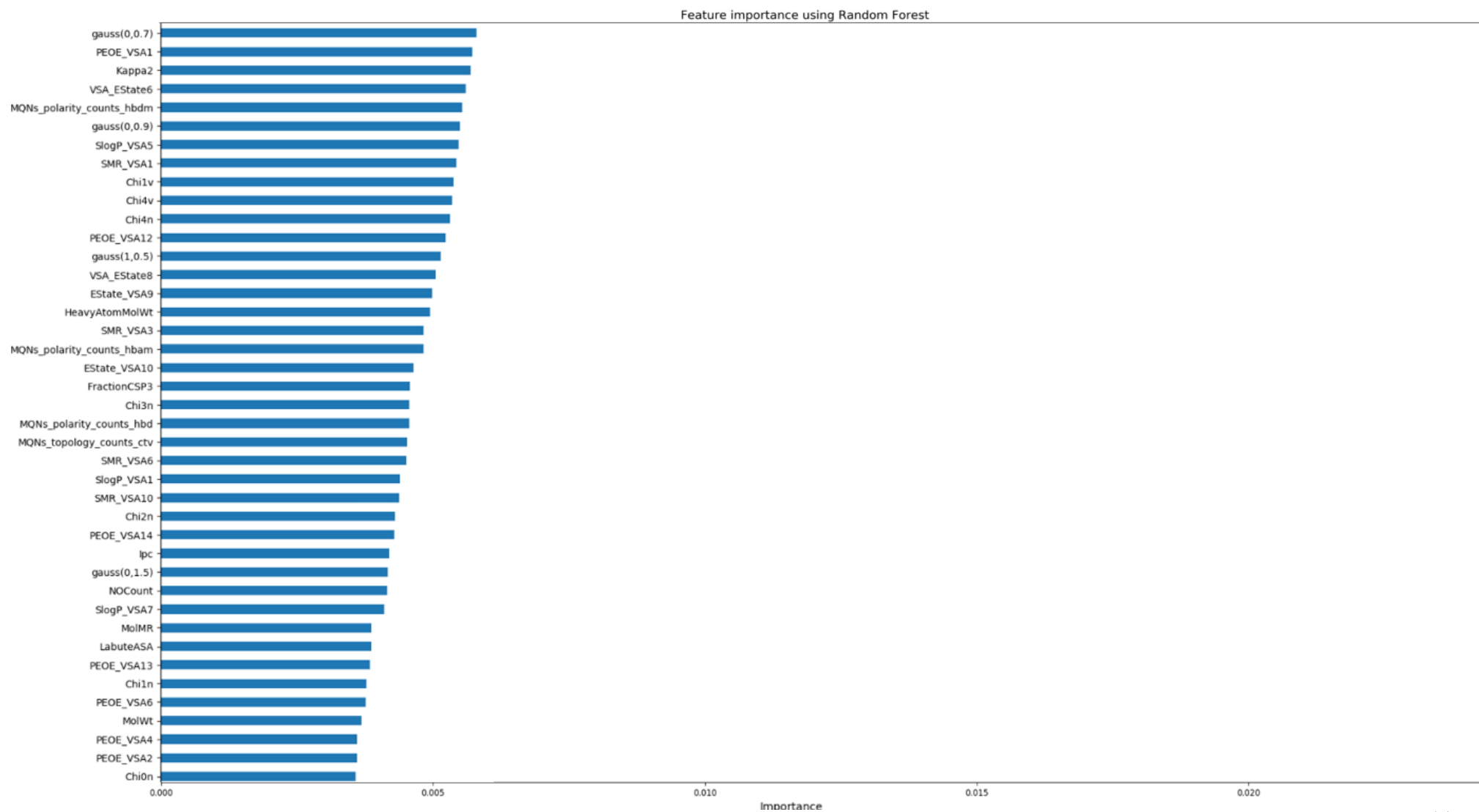


Figura 104: Medida de importância calculada pelo algoritmo *Random Forest* para p_{k_d} (83 a 123)

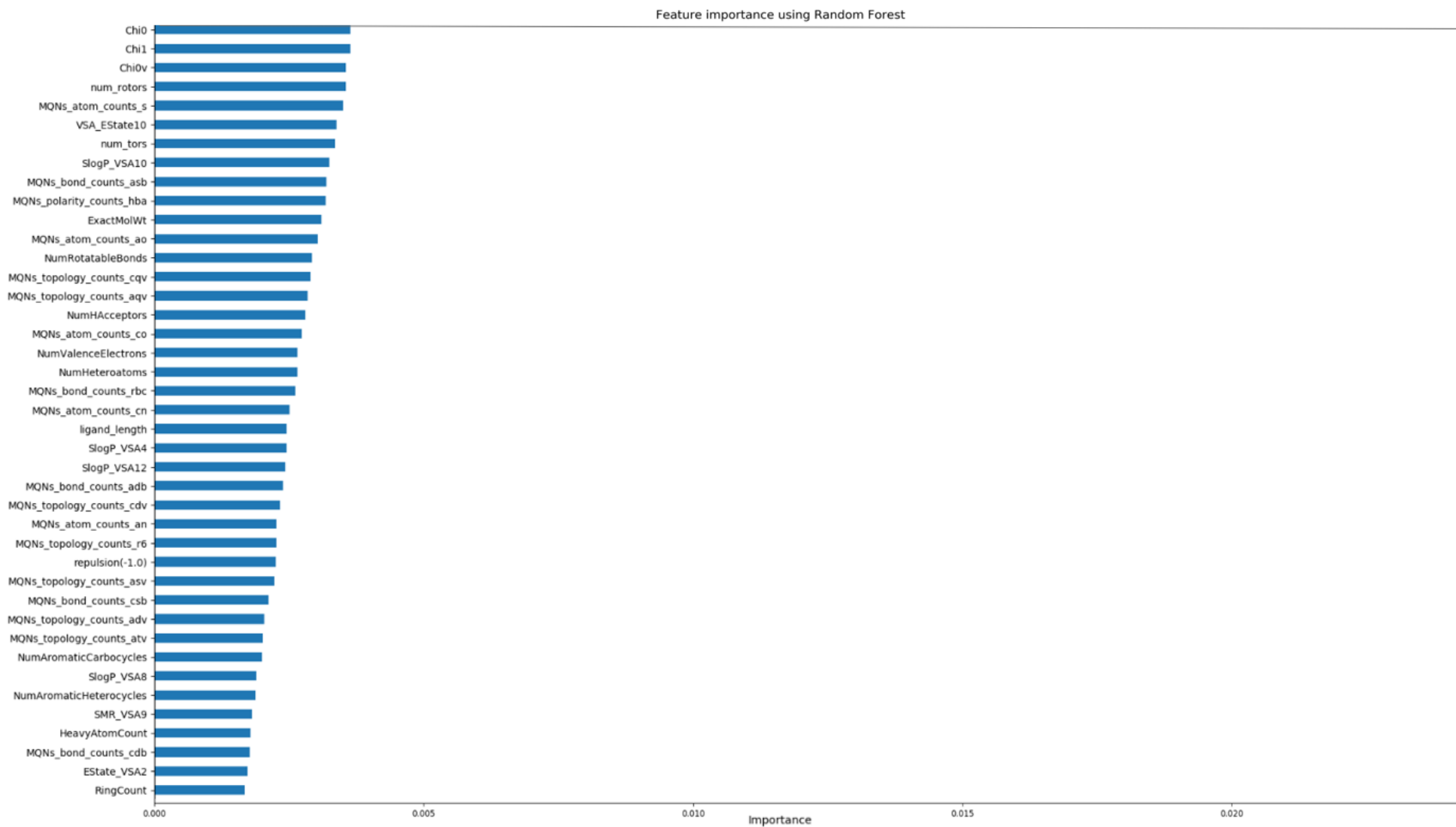


Figura 105: Medida de importância calculada pelo algoritmo *Random Forest* para p_{k_d} (125 a 164)

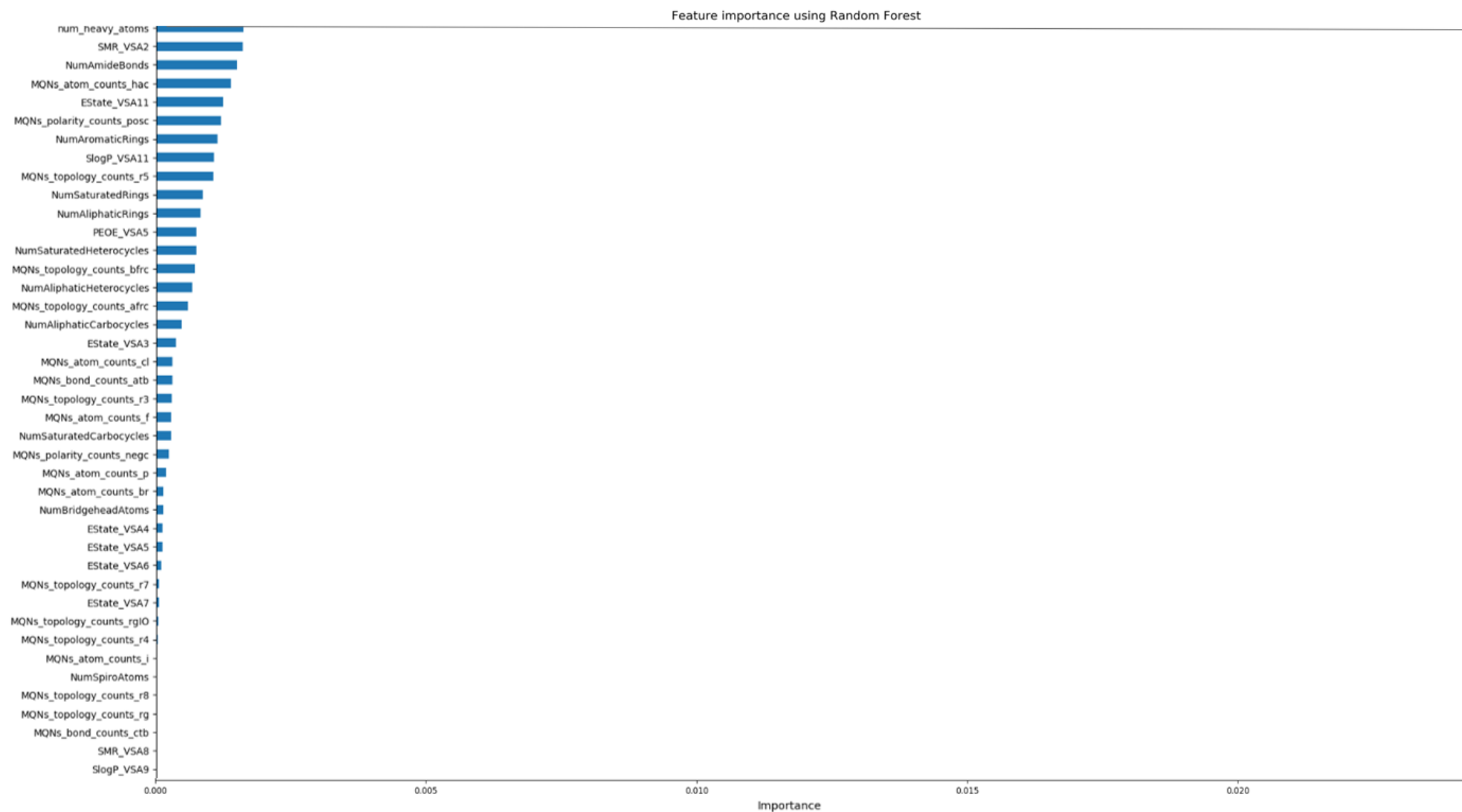


Figura 106: Medida de importância calculada pelo algoritmo *Random Forest* para p_{k_d} (165 a 205)

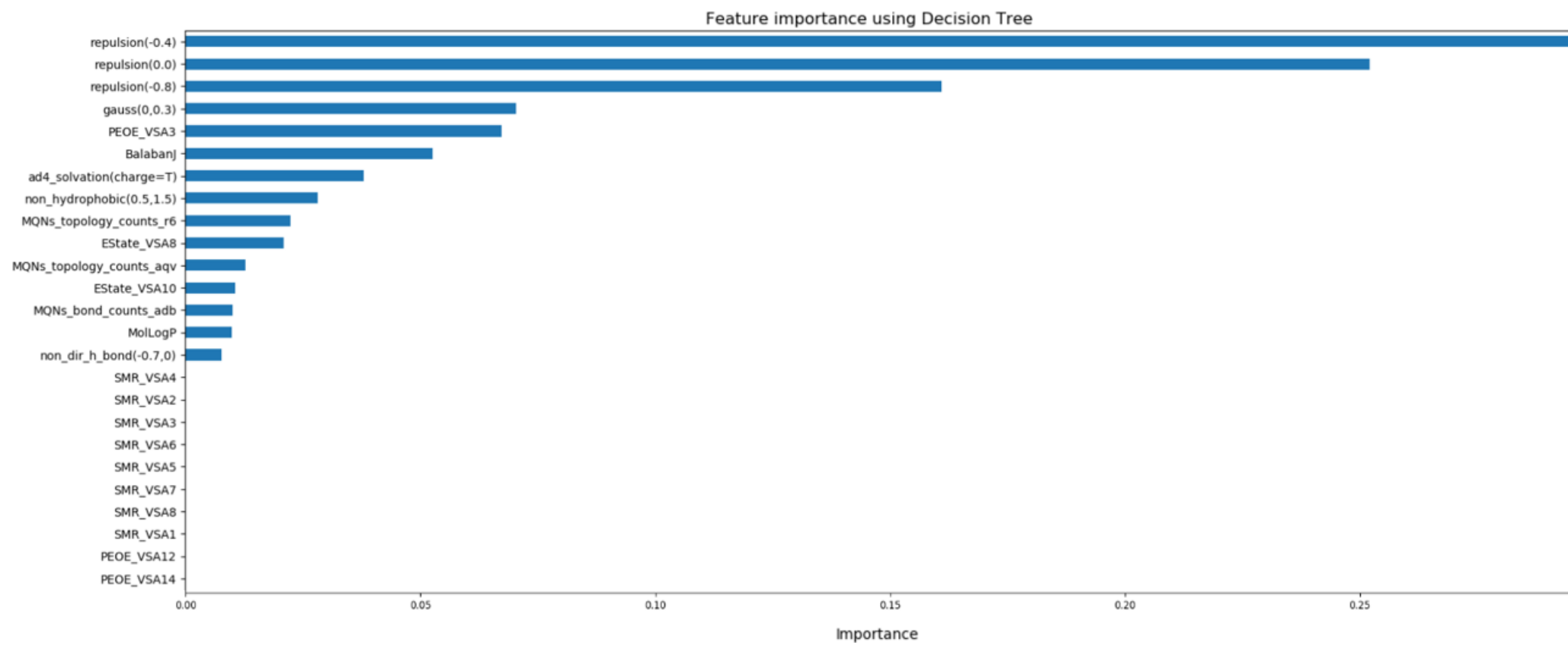


Figura 107: Medida de importância calculada pelo algoritmo *Decision Tree* para Δp_{k_d}

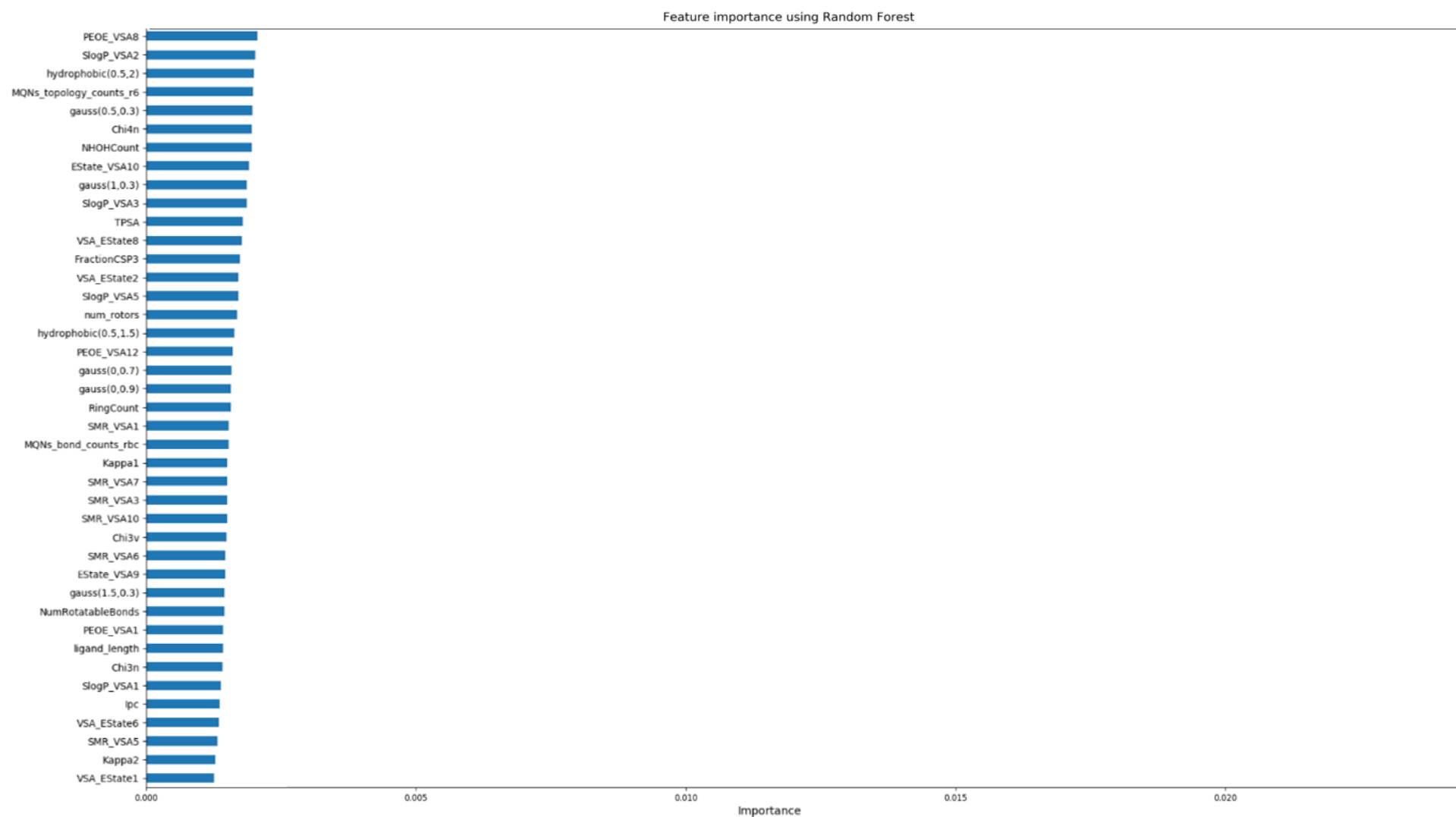


Figura 108: Medida de importância calculada pelo algoritmo *Random Forest* para Δp_{k_d} (1 a 41)

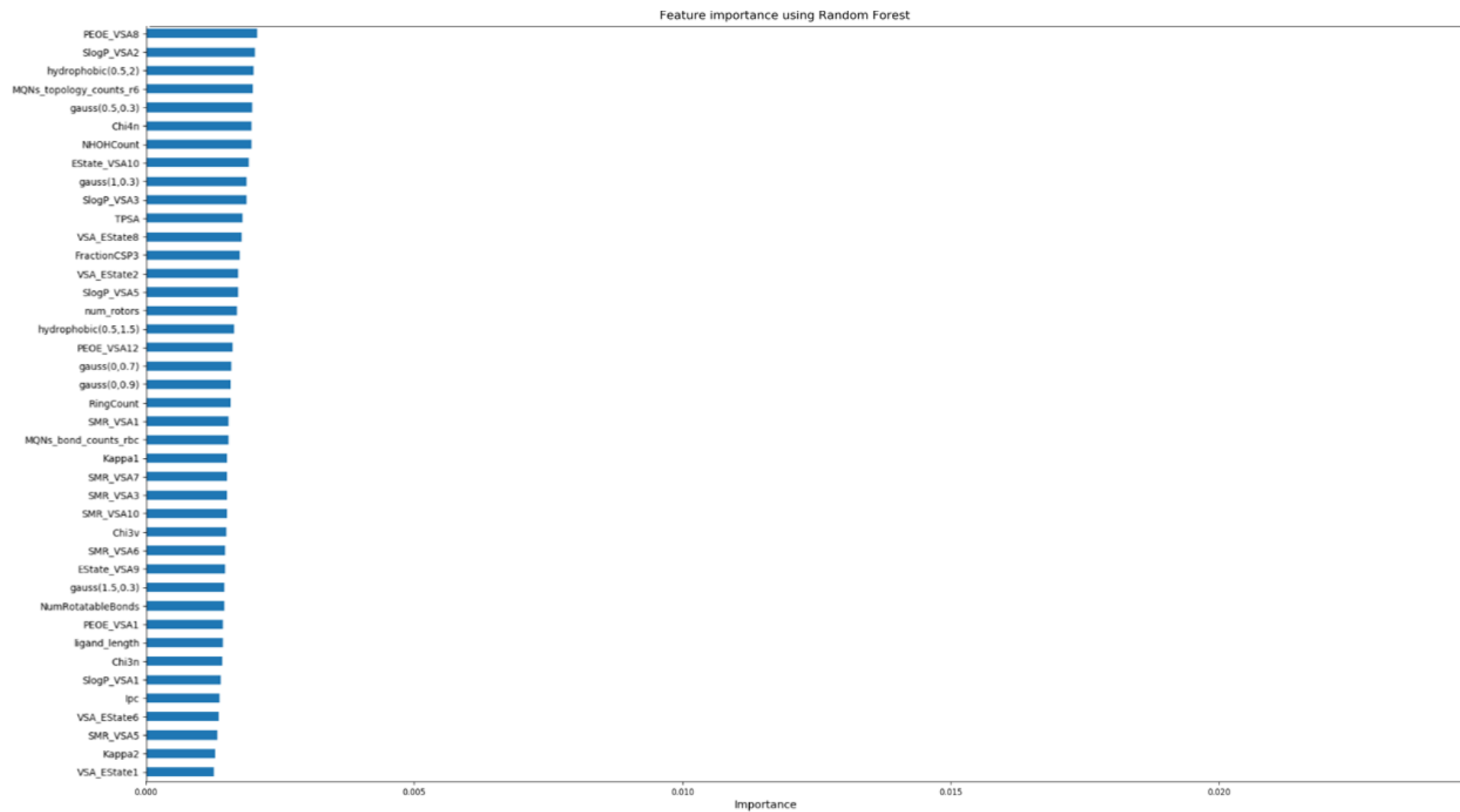


Figura 109: Medida de importância calculada pelo algoritmo *Random Forest* para Δp_{k_d} (42 a 81)

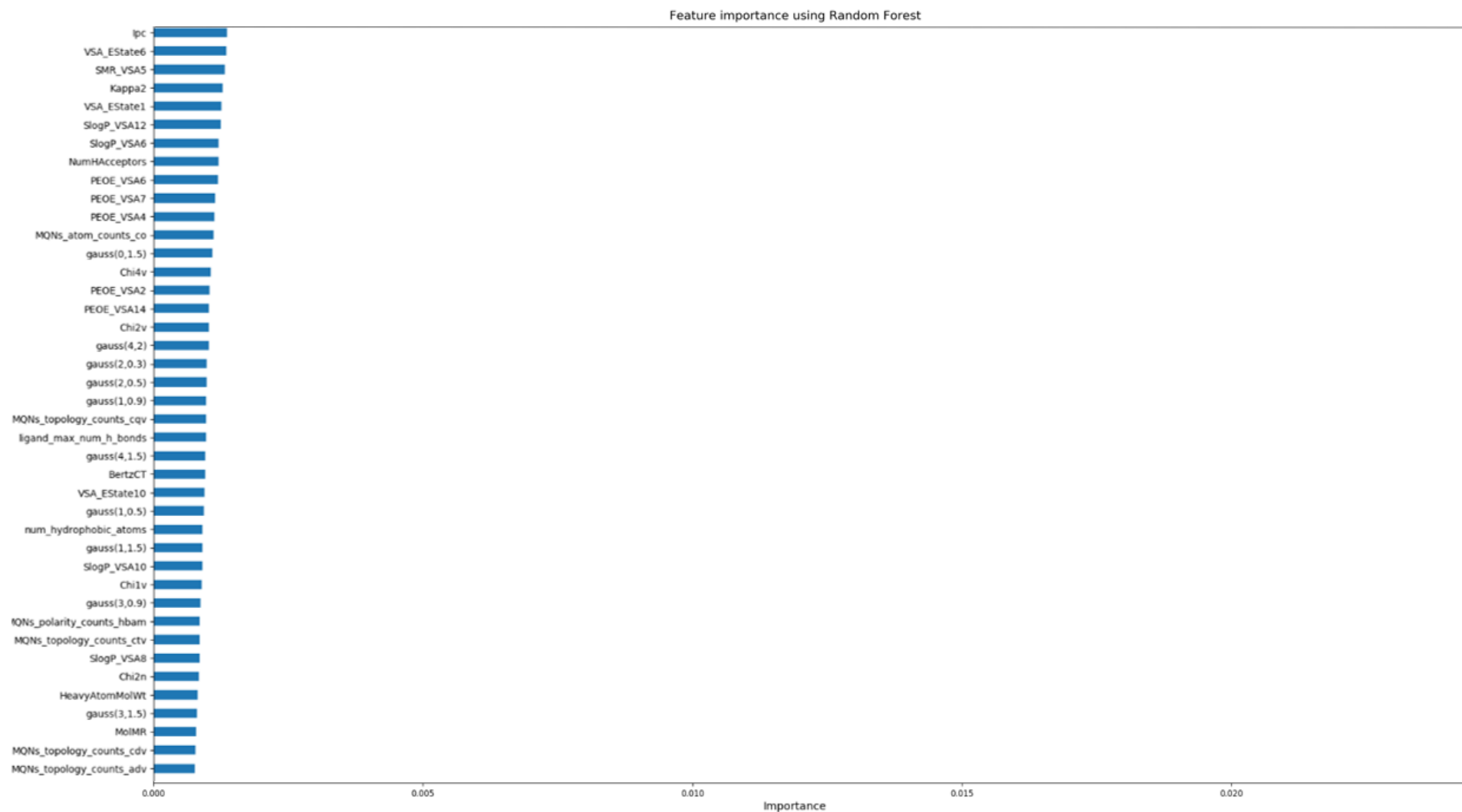


Figura 110: Medida de importância calculada pelo algoritmo *Random Forest* para Δp_{k_d} (82 a 123)

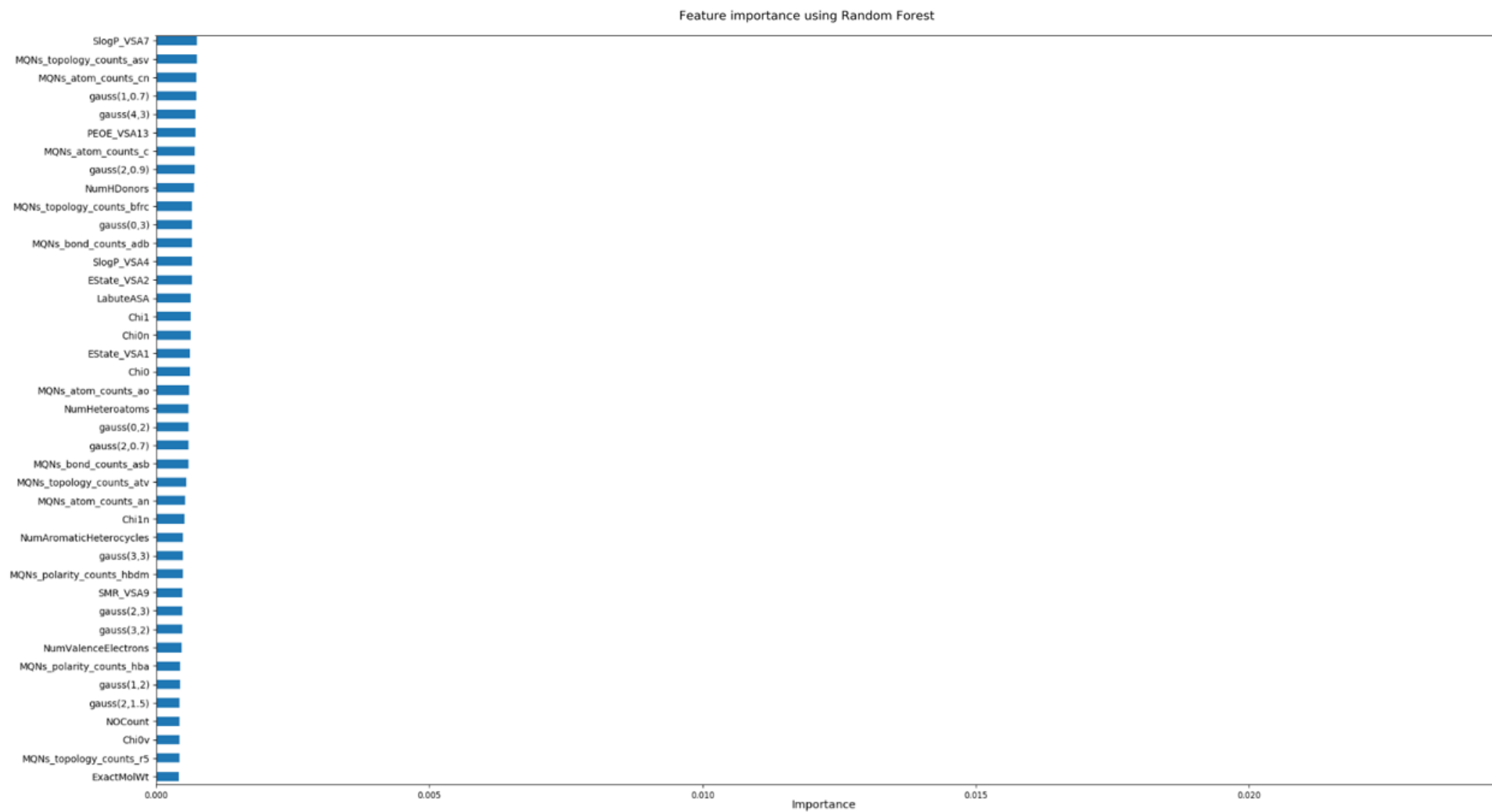


Figura 111: Medida de importância calculada pelo algoritmo *Random Forest* para Δp_{k_d} (124 a 163)

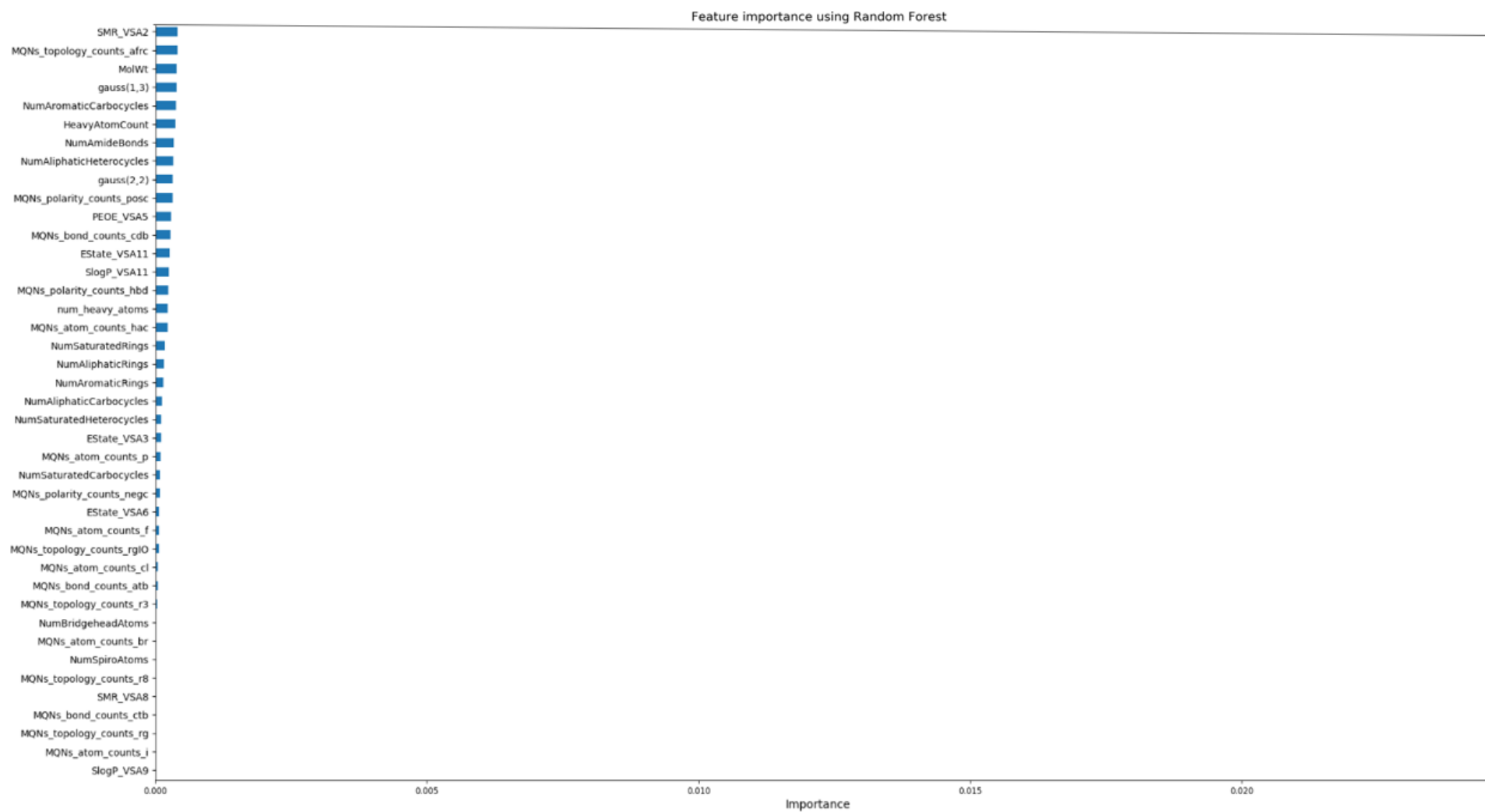


Figura 112: Medida de importância calculada pelo algoritmo *Random Forest* para Δp_{k_d} (164 a 205)

A.7 Configuração da escolha dos parâmetros para p_{kd}

Grupo de Atributos com maior importância	Parâmetros obtidos para o RF	Parâmetros obtidos para o DT	Parâmetros obtidos para o GBRT	Parâmetros obtidos para o SVM
5	max_features: 0.2, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.5, min_samples_leaf: 5, min_samples_split: 4	C: 1.0, gamma: 0.2
10	max_features: 0.2, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.7, min_samples_leaf: 10, min_samples_split: 6	C: 10.0, gamma: 0.0001
15	max_features: 0.2, min_samples_leaf: 1, n_estimators: 75	max_depth: 5, max_leaf_nodes: None, min_samples_split: 15	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 6	C: 10.0, gamma: 0.0001
20	max_features: 0.5, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 2	max_features: 0.5, min_samples_leaf: 5, min_samples_split: 8	C: 10.0, gamma: 0.0001
30	max_features: 0.2, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.5, min_samples_leaf: 2, min_samples_split: 4	C: 1.0, gamma: 0.0001
40	max_features: 0.5, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.5, min_samples_leaf: 5, min_samples_split: 6	C: 1.0, gamma: 0.0001
50	max_features: 0.5, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.5, min_samples_leaf: 10, min_samples_split: 6	C: 1.0, gamma: 0.0001
70	max_features: 0.2, min_samples_leaf: 2, n_estimators: 75	max_depth: 5, max_leaf_nodes: None, min_samples_split: 15	max_features: 0.5, min_samples_leaf: 1, min_samples_split: 4	C: 1.0, gamma: 0.0001
80	max_features: 0.7, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.5, min_samples_leaf: 1, min_samples_split: 4	C: 1.0, gamma: 0.0001
90	max_features: 0.5, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 6	C: 1.0, gamma: 0.0001
100	max_features: 0.2, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 2	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 6	C: 1.0, gamma: 0.0001
150	max_features: 0.7, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.5, min_samples_leaf: 10, min_samples_split: 6	C: 1.0, gamma: 0.0001
200	max_features: 0.2, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.5, min_samples_leaf: 5, min_samples_split: 8	C: 1.0, gamma: 0.0001
205	max_features: 0.7, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.5, min_samples_leaf: 5, min_samples_split: 4	C: 1.0, gamma: 0.0001

A.8 Configuração da escolha dos parâmetros para Δp_{kd}

Grupo de Atributos com maior importância	Parâmetros obtidos para o RF	Parâmetros obtidos para o DT	Parâmetros obtidos para o GBRT	Parâmetros obtidos para o SVM
5	max_features: 1.0, min_samples_leaf: 1, n_estimators: 75	max_depth: 10, max_leaf_nodes: None, min_samples_split: 15	max_features: 1.0, min_samples_leaf: 1, min_samples_split: 6	C: 100.0, gamma: 0.0001
10	max_features: 0.7, min_samples_leaf: 2, n_estimators: 100	max_depth: 10, max_leaf_nodes: None, min_samples_split: 10	max_features: 1.0, min_samples_leaf: 2, min_samples_split: 4	C: 100.0, gamma: 0.0001
15	max_features: 0.5, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 15	max_features: 1.0, min_samples_leaf: 2, min_samples_split: 4	C: 100.0, gamma: 0.0001
20	max_features: 0.5, min_samples_leaf: 2, n_estimators: 75	max_depth: 5, max_leaf_nodes: None, min_samples_split: 15	max_features: 0.7, min_samples_leaf: 2, min_samples_split: 8	C: 100.0, gamma: 0.0001
30	max_features: 0.5, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 4	C: 100.0, gamma: 0.0001
40	max_features: 0.5, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 8	C: 100.0, gamma: 0.0001
50	max_features: 0.5, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 2	C: 100.0, gamma: 0.0001
70	max_features: 0.2, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.7, min_samples_leaf: 2, min_samples_split: 8	C: 10.0, gamma: 0.0001
80	max_features: 0.5, min_samples_leaf: 2, n_estimators: 75	max_depth: 5, max_leaf_nodes: None, min_samples_split: 5	max_features: 0.5, min_samples_leaf: 1, min_samples_split: 6	C: 1.0, gamma: 0.0001
90	max_features: 1.0, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 2	max_features: 0.5, min_samples_leaf: 1, min_samples_split: 8	C: 1.0, gamma: 0.0001
100	max_features: 0.2, min_samples_leaf: 1, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.7, min_samples_leaf: 1, min_samples_split: 4	C: 1.0, gamma: 0.0001
150	max_features: 0.2, min_samples_leaf: 2, n_estimators: 100	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.7, min_samples_leaf: 2, min_samples_split: 6	C: 1.0, gamma: 0.0001
200	max_features: 0.5, min_samples_leaf: 1, n_estimators: 75	max_depth: 5, max_leaf_nodes: None, min_samples_split: 15	max_features: 0.5, min_samples_leaf: 2, min_samples_split: 8	C: 1.0, gamma: 0.0001
205	max_features: 0.2, min_samples_leaf: 1, n_estimators: 75	max_depth: 5, max_leaf_nodes: None, min_samples_split: 10	max_features: 0.7, min_samples_leaf: 2, min_samples_split: 6	C: 1.0, gamma: 0.0001