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

Disciplina: Princípios e Aplicações de Mineração de Dados

Tarefa Final

Maurício Balboni

Contents

1 Dataset - Pesquisa

Objetivo dele, tentar estimar a variável vina e pKd.

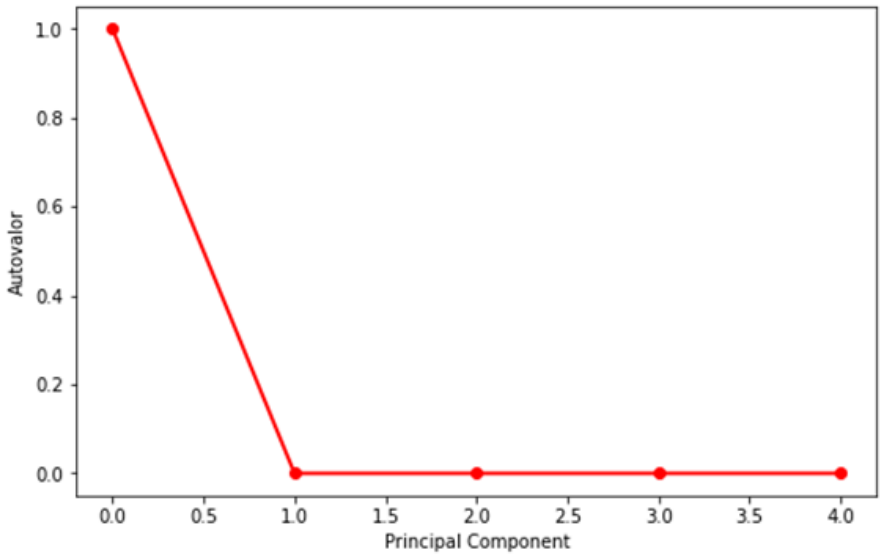
Como ele é um pdb com muitos atributos, o primeiro que eu tenho que fazer é a redução de atributos.

1.1 Estimando atributo "Vina"

Primeiramente eu tentei fazer com os valores padrão, exclui a primeira coluna que não é relevante para o calculo.

Depois Gerei o gráfico pra ver quantos PCA seriam necessários, pelo que eu entendi, pelo gráfico seria só 1, mas quando eu fiz pra 1 só, o resultado deu muito ruim, então, resolvi fazer com 5.

Figure 1: Gráfico de cotovelo



Fiz os 5 pca, e botei cada um numa variável, e depois observei de cada pca quais eram as variáveis que tinham alguma ligação com esses pca.

Ai descobri que as variáveis eram "$BalabanJ + D4 + NN115 + NN119 + NN124 + NN138 + NN161 + NN162 + NN172 + PBF + PMI1 + D34 + NN132 + S8 + F18 + F21 + F22 + F23 + F25 + F26 + F27 + F28 + F29 + F30 + F31 + F32 + F33 + R + D5 + NN192$"

No total foram 30 variáveis, então eu reduzi de 750 pra 30, reduzindo 96% do meu data set original.

1.1.1 Resultados

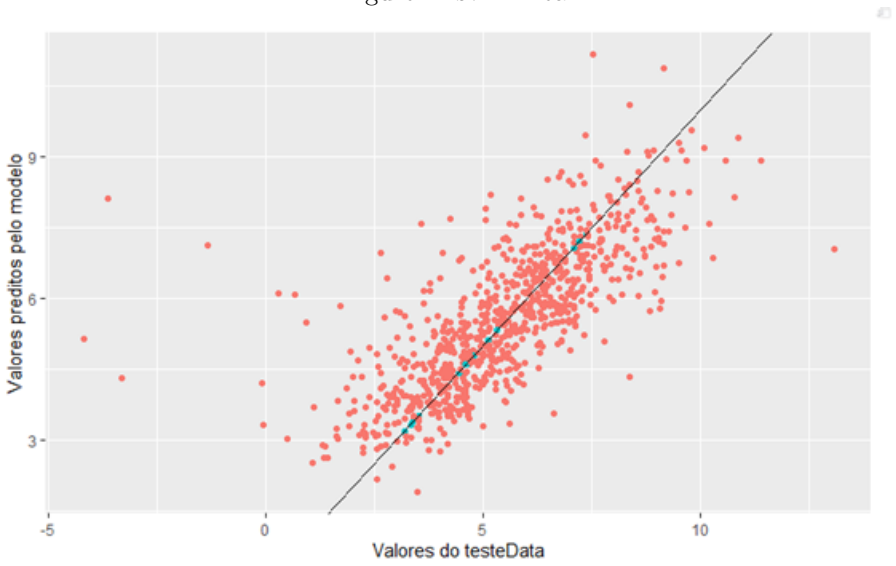
E treinei utilizando regressão com o método svmLinear, o meu resultado foi:

RMSE = 1.642832

RRSE = 2.115839

E meu gráfico ficou:

Figure 2: svmLinear



Não foi um resultado ruim, no meu entendimento, tal que eu reduzi 96% dos dados. Mas mesmo assim eu achei estranho, pois o PCA estava priorizando variáveis que tinham um valor alto.

1.1.2 Normalização

Então o que eu pensei em fazer, foi tentar normalizar esses dados. Busquei na literatura, e encontrei um formula:

$$\frac{valor \times Media\_Coluna}{Desvio\_Padrão\_da\_Coluna} \tag{1}$$

E também vi uma médida de avaliação se meus dados estavam normalizados que é o teste de fligner, inclusive já tem implementado no R esse teste, apliquei ele pra verificar se meus dados estavam normalizados.

1.2 Redução de Atributos com PCA

Fiz os PCA de novo pra verificar as novas variáveis, com um threshold de 0.1, e ele me retornou 77 novas variáveis, sendo elas: "$F58+NN3+NN4+NN29+NN51+NN62+NN219+NN233+NN234+NN285+NN313+Chi4v+NumBridgeheadAtoms+PEOE_VSA6+SMR_VSA6+SlogP_VSA5+S1+S5+S10+F2+F3+F43+F44+F45+F46+F47+F49+F50+F51+F55+F56+NN28+NN50+NN61+NN161+NN232+NN284+nBondsM+n4Ring+nF8Ring+nT4Ring+nT8Ring+nTG12Ring+n4HeteroRing+nF8HeteroRing+nT4HeteroRing+nT8HeteroRing+nTG12HeteroRing+SlogP_VSA7+EState_VSA1+nG12Ring+nF10Ring+nF11Ring+nF12Ring+nFG12Ring+nT10Ring+nT11Ring+nT12Ring+nF10HeteroRing+nF11HeteroRing+nF12HeteroRing+nT10HeteroRing+nT11HeteroRing+nT12HeteroRing+NumSaturatedRings+NumSpiroAtoms+LabuteASA+PEOE_VSA9+SMR_VSA1+VSA_EState6+MQNs_topology_counts_rgIO+NN9+NN45+NN47+NN52+NN60+NN100+NN106+NN229+NN231+PEOE_VSA10+SMR_VSA2+SlogP_VSA6$".

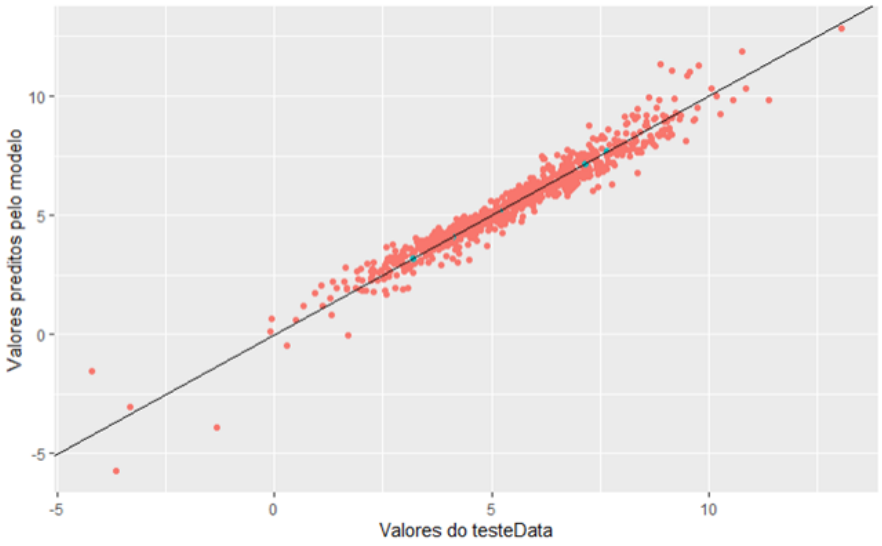
1.2.1 Regressão Linear com "svmLinear"

Fiz o treinamento com regressão linear com essas 77 variáveis, utilizando o método svmLinear e meu resultado foi:

1.2.1.1 Resultados

RMSE = 0.4713043.
RRSE = 0.2368207.
E meu gráfico ficou:

Figure 3: svmLinear



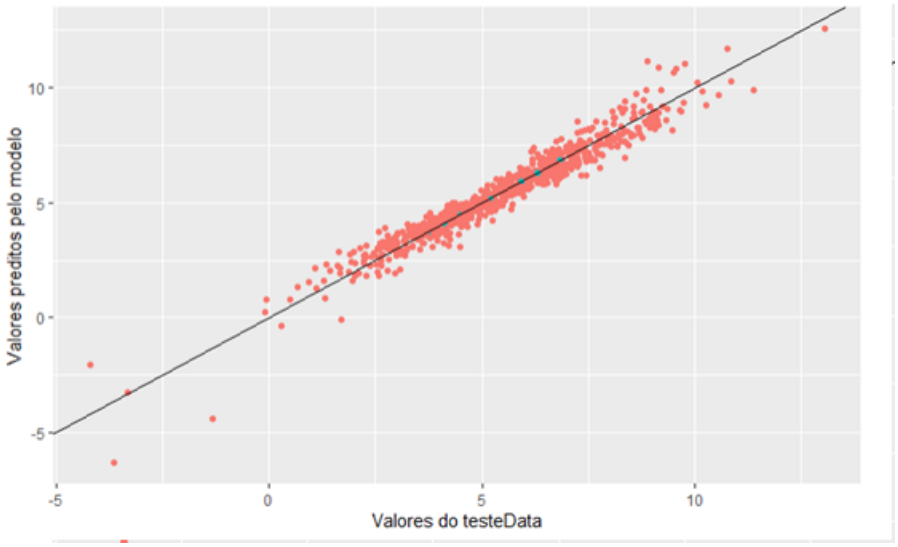
1.2.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.2.2.1 Resultados

RMSE = 0.4767907.
RRSE = 0.2395774.
E meu gráfico ficou:

Figure 4: enet



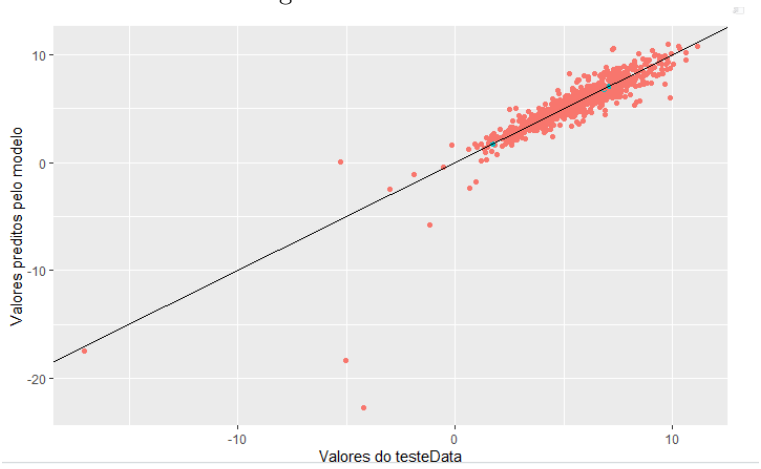
1.2.3 Rede Neural

E rodei também com uma Rede neural resultado com 2 camadas ocultas de 50 neurônios cada, procurei na literatura algo sobre quantas camadas ocultas usar, e achei pouca coisa, achei na verdade um artigo falando que o ganho a partir de "n" camada é bem baixo, então muitas vezes não é viável computacionalmente utilizar muitas camadas, usei threshold de 0.1.

1.2.3.1 Resultados

RMSE = 0.4767907.
RRSE = 0.2395774.
E meu gráfico ficou:

Figure 5: Rede Neural



1.3 Redução de Atributos com ANOVA F-Value

Então resolvi fazer a seleção de atributos com outros métodos do estilo "feature importance", procurei na literatura e encontrei um método utilizado em variáveis experimentais, achei que se encaixava com o dataset, o nome do método é "ANOVA F-value".

Apliquei ele, e normalmente esses outros métodos ao contrario do pca, eles precisam de um "k" variáveis importantes, resolvi utilizar k = 77, pois foi isso que o PCA me retornou e é melhor pra comparar os resultados.

As variáveis que ele me retornou foram:

"$NN231 + NN243 + NN248 + NN254 + NN255 + NN262 + NN265 + NN277 + NN278 + NN279 + NN284 + NN285 + NN287 + NN288 + NN292 + NN293 + NN295 + NN296 + NN297 + NN300 + nS + nCl + nBr + nI + n7Ring + n8Ring + n12Ring + nG12Ring + nF4Ring + nF5Ring + nF6Ring + nF7Ring + nF8Ring + nF11Ring + nF12Ring + nFG12Ring + nT7Ring + nT8Ring + nT11Ring + nT12Ring + nTG12Ring + n3HeteroRing + n7HeteroRing + n12HeteroRing + nG12HeteroRing + nF4HeteroRing + nF5HeteroRing + nF6HeteroRing + nF7HeteroRing + nF8HeteroRing + nF11HeteroRing + nF12HeteroRing + nFG12HeteroRing + nT7HeteroRing + nT8HeteroRing + nT11HeteroRing + nT12HeteroRing + nTG12HeteroRing + NumHAcceptors + NumSaturatedHeterocycles + NumAliphaticHeterocycles + NumSpiroAtoms + NumBridgeheadAtoms + EState_VSA4 + EState_VSA5 + MQNs_atom_counts_cl + MQNs_atom_counts_br + MQNs_atom_counts_i + MQNs_atom_counts_an + MQNs_atom_counts_co + MQNs_topology_counts_atv + MQNs_topology_counts_dv + MQNs_topology_counts_qv + MQNs_topology_counts_r7 + MQNs_topology_counts_r8 + MQNs_topology_counts_rgIO + MQNs_topology_counts_bfrc$".

Um pouco diferente das do PCA.

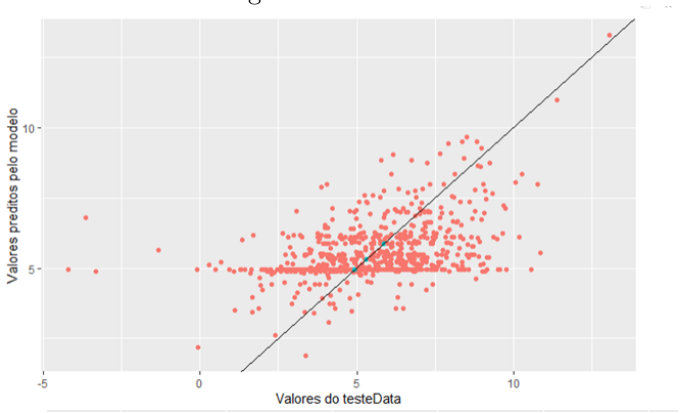
1.3.1 Regressão Linear com "svmLinear"

Fiz o treinamento com regressão linear com essas 77 variáveis, utilizando o método svmLinear e meu resultado foi:

1.3.1.1 Resultados

RMSE = 1.783138.
RRSE = 0.89599.
E meu gráfico ficou:

Figure 6: svmLinear



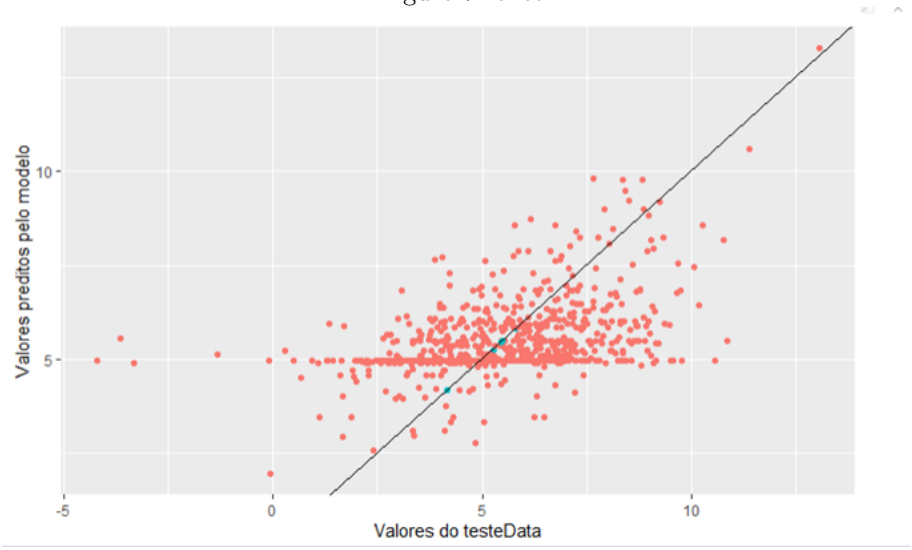
1.3.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.3.2.1 Resultados

RMSE = 1.76689.
RRSE = 0.8878259.
E meu gráfico ficou:

Figure 7: enet



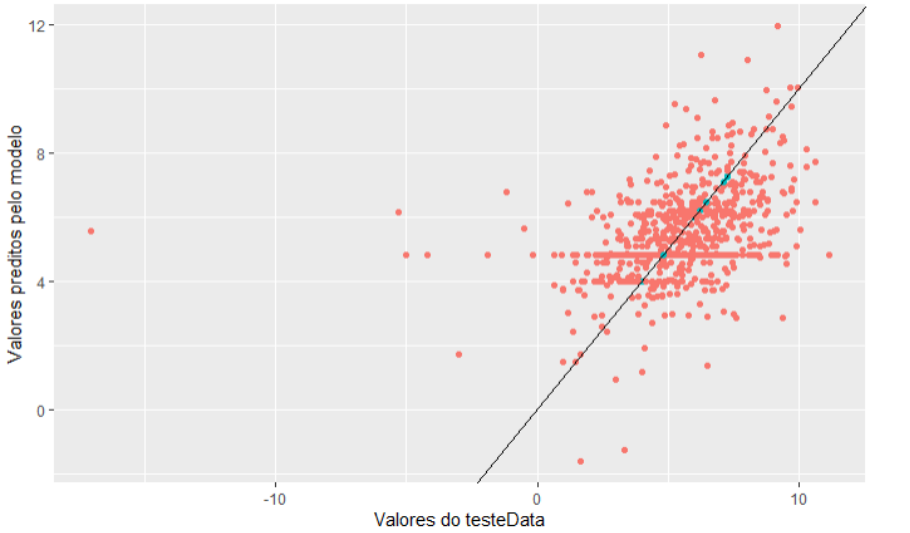
1.3.3 Rede Neural

E rodei também com uma Rede neural resultado com 2 camadas ocultas de 50 neurônios cada e threshold de 0.1.

1.3.3.1 Resultados

RMSE = 2.053643.
RRSE = 0.2395774.
E meu gráfico ficou:

Figure 8: Rede Neural



1.4 Redução de Atributos com *Random Forest Regression*

Como eu variáveis continuas, com o *Random Forest* normal não deu pra fazer.
Rodei com ele e ele me retornou as 77 variáveis, em ordem de importância.
As variáveis que ele me retornou foram:
"$F49 + NN3 + F40 + F41 + F42 + nRing + F39 + F53 + F54 + Kappa3 + F20 + F44 + NN350 + nTRing + F5 + F58 + F47 + F50 + MQNs_bond_counts_csb + S5 + NN4 + F30 + F25 + F46 + HallKierAlpha + F38 + Asphericity + NN2 + F24 + F10 + BalabanJ + F6 + F51 + EState_VSA2 + MQNs_atom_counts_s + F21 + F11 + NN5 + NN1 + F31 + F8 + F13 + F2 + MQNs_topology_counts_aqv + S6 + F15 + F9 + F52 + EState_VSA10 + F45 + NN344 + SlogP_VSA3 + F22 + PEOE_VSA1 + MQNs_topology_counts_asv + F28 + pKd + F26 + M + S2 + F19 + I + F37 + D6 + F17 + NN22 + F36 + F7 + F16 + Kappa1 + H + S8 + SMR_VSA10 + S3 + F32 + EState_VSA9 + PMI2$".

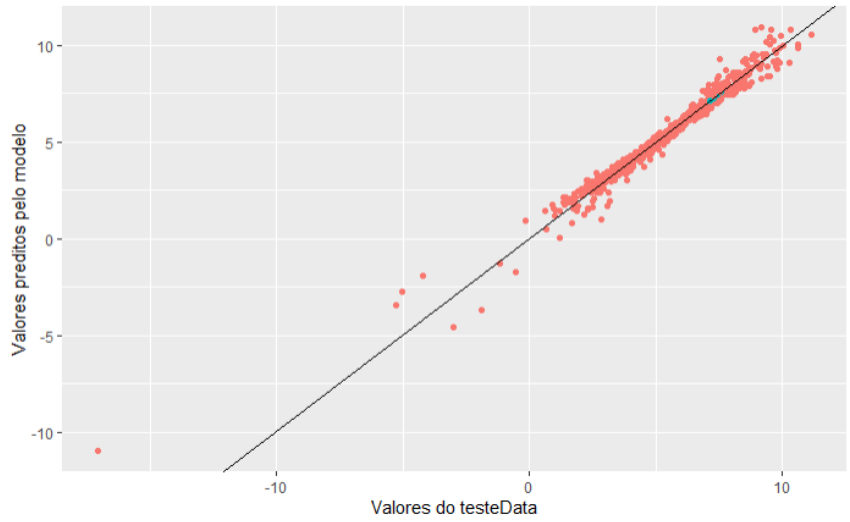
1.4.1 Regressão Linear com "svmLinear"

Fiz o treinamento com regressão linear com essas 77 variáveis, utilizando o método "svmLinear" e meu resultado foi:

1.4.1.1 Resultados

RMSE = 0.4198481.
RRSE = 0.189789.
E meu gráfico ficou:

Figure 9: svmLinear



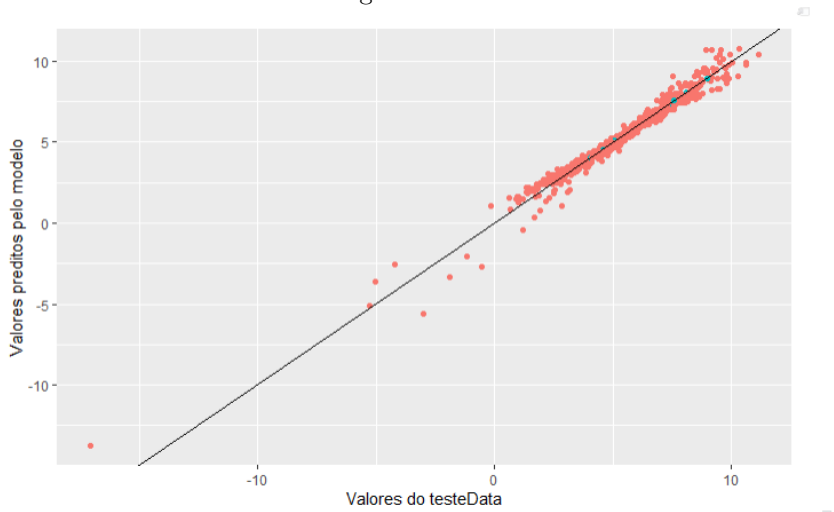
1.4.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.4.2.1 Resultados

RMSE = 0.3848866.
RRSE = 0.1739849.
E meu gráfico ficou:

Figure 10: enet



1.4.3 Rede Neural

E rodei também com uma Rede neural com 2 camadas ocultas de 50 neurônios cada e threshold de 0.1.

1.4.3.1 Resultados

RMSE = 1.204053.
RRSE = 0.5442826.
E meu gráfico ficou:

Figure 11: Rede Neural

1.5 Tentativa de obter melhores modelos

Como o *Random Forest Regression* me retornou o melhor resultado, eu decide partir dele pra tentar criar modelos mais precisos.

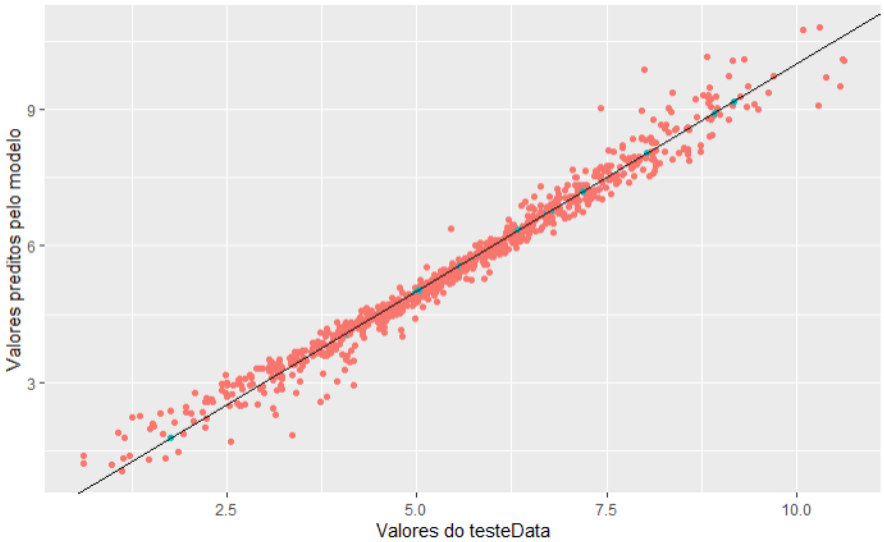
Primeira coisa que eu fiz foi remover os dados discrepantes da minha variável resposta ("vina"), então, apos remover e colocar pra treinar o modelo, o resultado foi:

1.5.1 Regressão Linear com svmLinear

1.5.1.1 Resultados

RMSE = 0.2998834.
RRSE = 0.1614564.
E meu gráfico ficou:

Figure 12: svmLinear



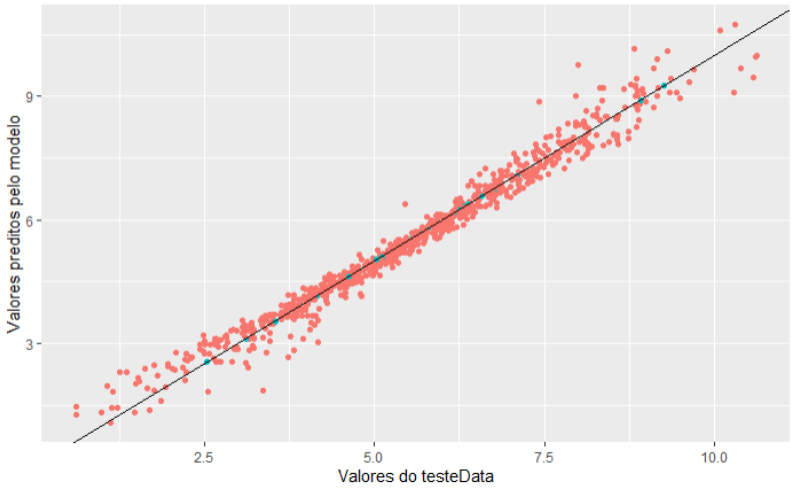
1.5.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.5.2.1 Resultados

RMSE = 0.3015436.
RRSE = 0.1623502.
E meu gráfico ficou:

Figure 13: enet



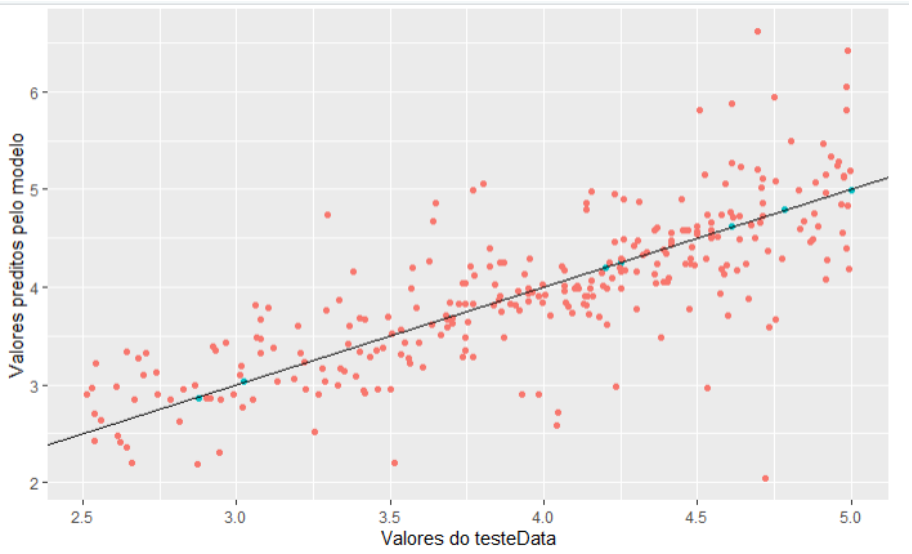
1.5.3 Rede Neural

E rodei também com uma Rede neural com 2 camadas ocultas de 50 neurônios cada e threshold de 0.1.

1.5.3.1 Resultados

RMSE = 0.499828.
RRSE = 0.7450212.
E meu gráfico ficou:

Figure 14: Rede Neural



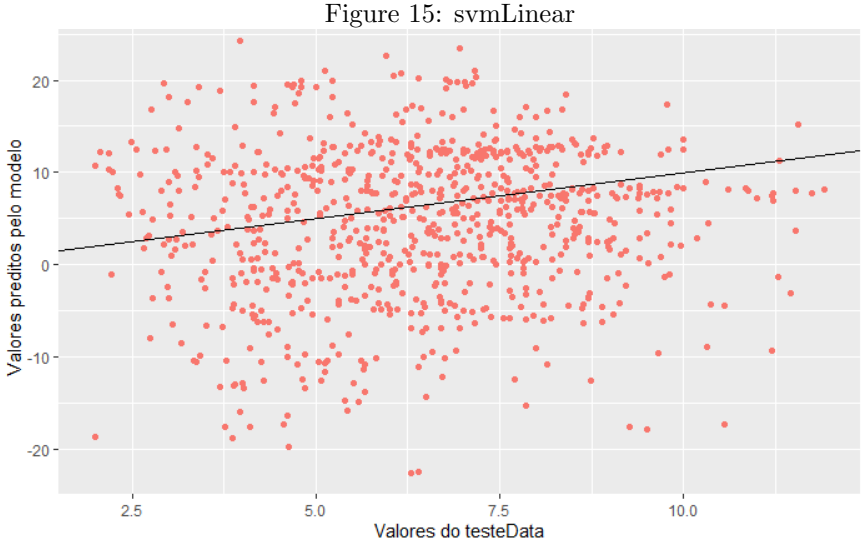
1.6 Estimando atributo "pKd"

Realizei o mesmo processo que com o "vina".
Peguei minhas 30 variáveis, e rodei com o svmLinear.

1.6.1 Resultados

RMSE = 8.431763
RRSE = 4.31719
E meu gráfico ficou:

Figure 15: svmLinear



1.7 Redução de Atributos com PCA

Fiz os PCA de novo pra verificar as novas variáveis, com um threshold de 0.1, e ele me retornou 77 novas variáveis, sendo elas: "$F58+NN3+NN4+NN29+NN51+NN62+NN219+NN233+NN234+NN285+NN313+Chi4v+NumBridgeheadAtoms+PEOE_VSA6+SMR_VSA6+SlogP_VSA5+S1+S5+S10+F2+F3+F43+F44+F45+F46+F47+F49+F50+F51+F55+F56+NN28+NN50+NN61+NN161+NN232+NN284+nBondsM+n4Ring+nF8Ring+nT4Ring+nT8Ring+nTG12Ring+n4HeteroRing+nF8HeteroRing+nT4HeteroRing+nT8HeteroRing+nTG12HeteroRing+SlogP_VSA7+EState_VSA1+nG12Ring+nF10Ring+nF11Ring+nF12Ring+nFG12Ring+nT10Ring+nT11Ring+nT12Ring+nF10HeteroRing+nF11HeteroRing+nF12HeteroRing+nT10HeteroRing+nT11HeteroRing+nT12HeteroRing+NumSaturatedRings+NumSpiroAtoms+LabuteASA+PEOE_VSA9+SMR_VSA1+VSA_EState6+MQNs_topology_counts_rgIO+NN9+NN45+NN47+NN52+NN60+NN100+NN106+NN229+NN231+PEOE_VSA10+SMR_VSA2+SlogP_VSA6$".

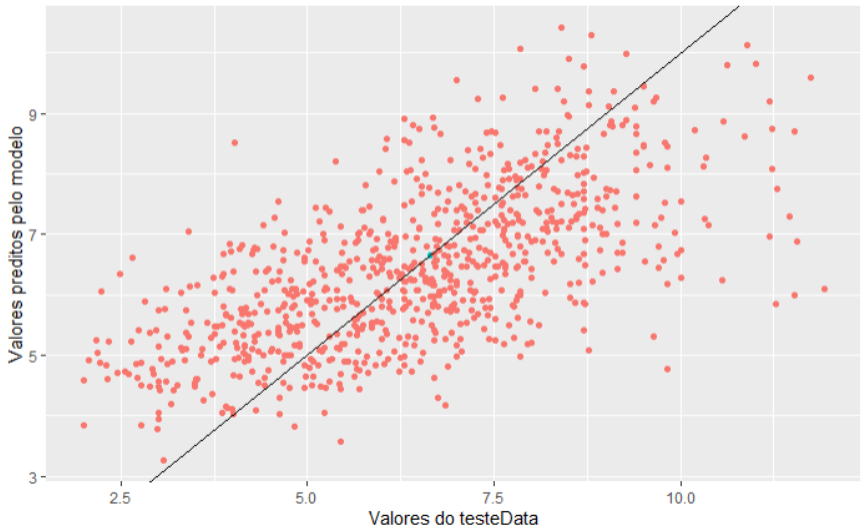
1.7.1 Regressão Linear com "svmLinear"

Fiz o treinamento com regressão linear com essas 77 variáveis, utilizando o método svmLinear e meu resultado foi:

1.7.1.1 Resultados

RMSE = 1.493205.
RRSE = 0.7645434.
E meu gráfico ficou:

Figure 16: svmLinear



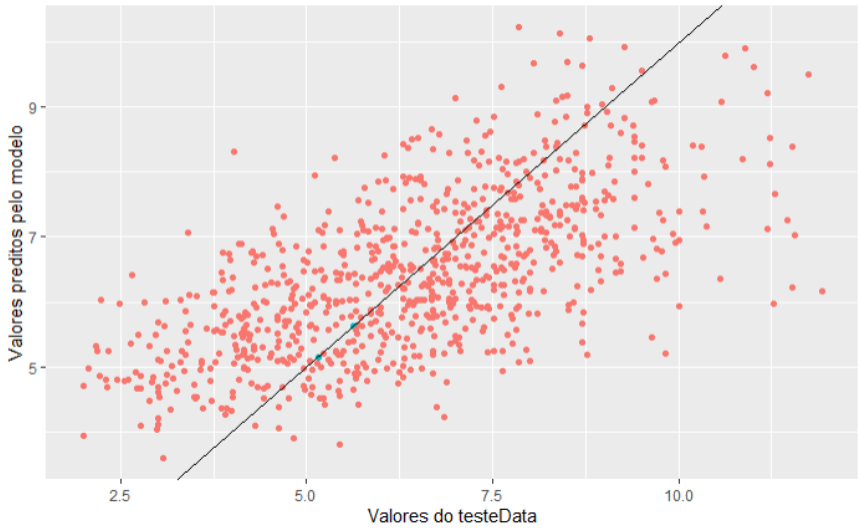
1.7.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.7.2.1 Resultados

RMSE = 1.497195.
RRSE = 0.7665864.
E meu gráfico ficou:

Figure 17: enet



1.7.3 Rede Neural

E rodei também com uma Rede neural resultado com 2 camadas ocultas de 50 neurônios cada, procurei na literatura algo sobre quantas camadas ocultas usar, e achei pouca coisa, achei na verdade um artigo falando que o ganho a partir de "n" camada é bem baixo, então muitas vezes não é viável computacionalmente utilizar muitas camadas, usei threshold de 0.1.

1.7.3.1 Resultados

RMSE = 2.224602.
RRSE = 1.13903.
E meu gráfico ficou:

Figure 18: Rede Neural

1.8 Redução de Atributos com ANOVA F-Value

Então resolvi fazer a seleção de atributos com outros métodos do estilo "feature importance", procurei na literatura e encontrei um método utilizado em variáveis experimentais, achei que se encaixava com o dataset, o nome do método é "ANOVA F-value".

Apliquei ele, e normalmente esses outros métodos ao contrario do pca, eles precisam de um "k" variáveis importantes, resolvi utilizar k = 77, pois foi isso que o PCA me retornou e é melhor pra comparar os resultados.

As variáveis que ele me retornou foram:
"$NN1 + NN2 + NN47 + NN60 + NN74 + NN89 + NN90 + NN97 + NN100 + NN105 + NN117 + NN194 + NN215 + NN216 + NN254 + NN278 + NN294 + NN296 + nBonds + nBondsS + nRing + n4Ring + nFRing + nF5Ring + nFG12Ring + nTRing + nT4Ring + nTG12Ring + n4HeteroRing + nF5HeteroRing + nFG12HeteroRing + nT4HeteroRing + nTG12HeteroRing + BertzCT + Chi2n + Chi3n + Chi4n + MolMR + NumSpiroAtoms + PEOE_VSA9 + EState_VSA1 + MQNs_topology_counts_r3 + MQNs_topology_counts_r4 + InertialShapeFactor + F2 + F6 + F7 + F8 + F9 + F10 + F11 + F12 + F13 + F14 + F15 + F16 + F17 + F18 + F19 + F20 + F21 + F22 + F23 + F24 + F25 + F26 + F27 + F28 + F29 + F30 + F31 + F32 + F33 + F34 + F35 + F49 + F56$".

Um pouco diferente das do PCA.

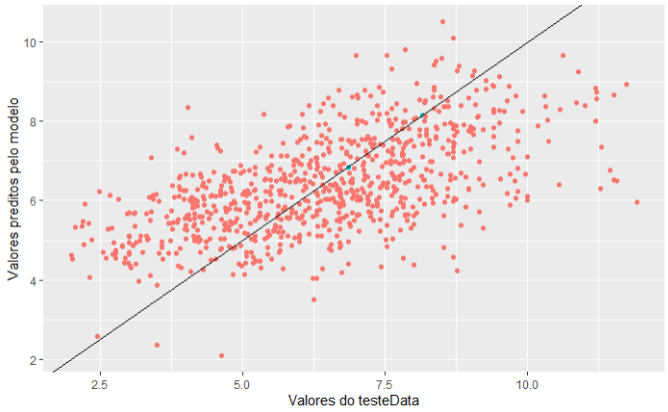
1.8.1 Regressão Linear com "svmLinear"

Fiz o treinamento com regressão linear com essas 77 variáveis, utilizando o método svmLinear e meu resultado foi:

1.8.1.1 Resultados

RMSE = 1.578801.
RRSE = 0.80837.
E meu gráfico ficou:

Figure 19: svmLinear



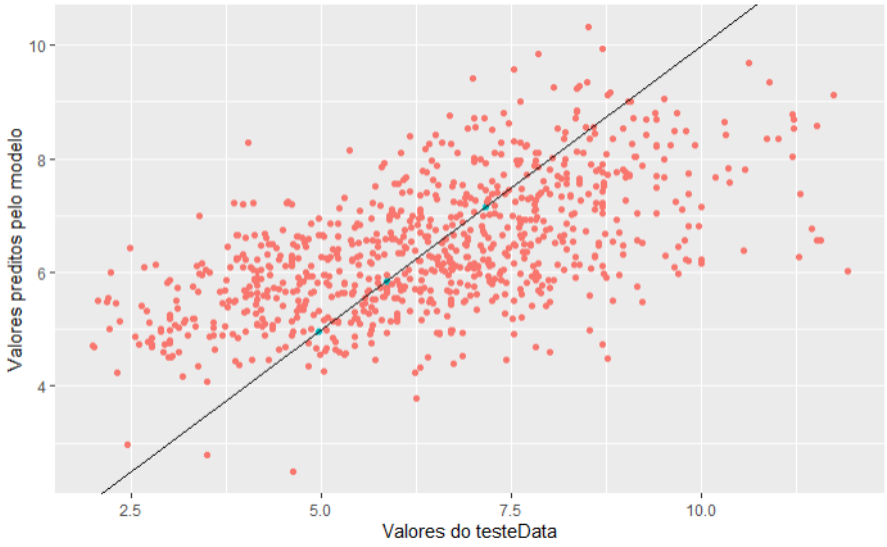
1.8.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.8.2.1 Resultados

RMSE = 1.577322.
RRSE = 0.8076125.
E meu gráfico ficou:

Figure 20: enet



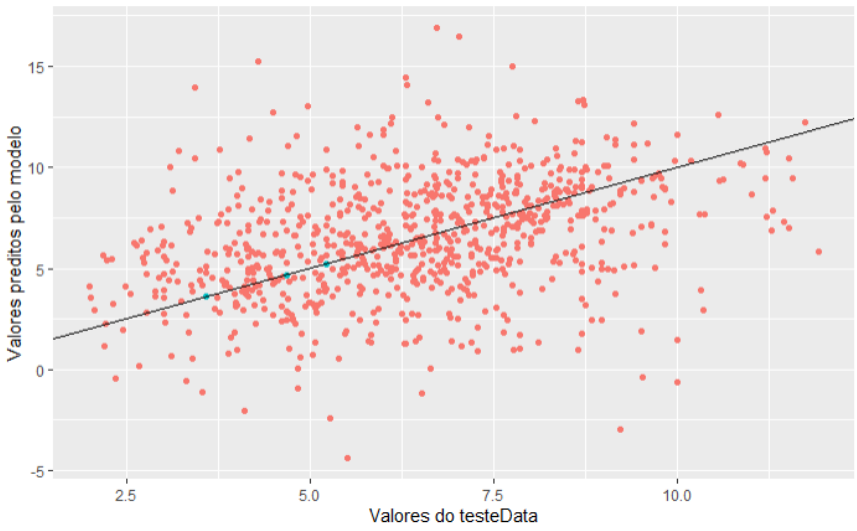
1.8.3 Rede Neural

E rodei também com uma Rede neural resultado com 2 camadas ocultas de 50 neurônios cada e threshold de 0.5, pois com 0.1 ele não convergia.

1.8.3.1 Resultados

RMSE = 2.868762
RRSE = 1.46885
E meu gráfico ficou:

Figure 21: Rede Neural



1.9 Redução de Atributos com Random Forest Regression

Como eu variáveis continuas, com o *Random Forest* normal não deu pra fazer.
Rodei com ele e ele me retornou as 77 variáveis, em ordem de importância.
As variáveis que ele me retornou foram:
"$EState_VSA1 + MolLogP + I + vina + F23 + VSA_EState1 + F24 + F30 + S + PEOE_VSA3 + G + NN61 + F25 + F33 + NN11 + F2 + M + NN174 + H + Y + EState_VSA8 + F48 + D10 + D + A + T + D32 + NN5 + F + D34 + W + NN77 + D6 + F20 + S2 + SMR_VSA3 + F28 + VSA_EState3 + S8 + F35 + NN161 + S5 + Q + VSA_EState2 + R + VSA_EState7 + D28 + NN25 + NN115 + N + P + S10 + K + F49 + SMR_VSA6 + F4 + F19 + D12 + C + Chi4n + S1 + SlogP_VSA7 + Kappa1 + D8 + E + NN28 + NN163 + NN30 + S7 + VSA_EState4 + F41 + F10 + D30 + L + VSA_EState9 + NN180$".

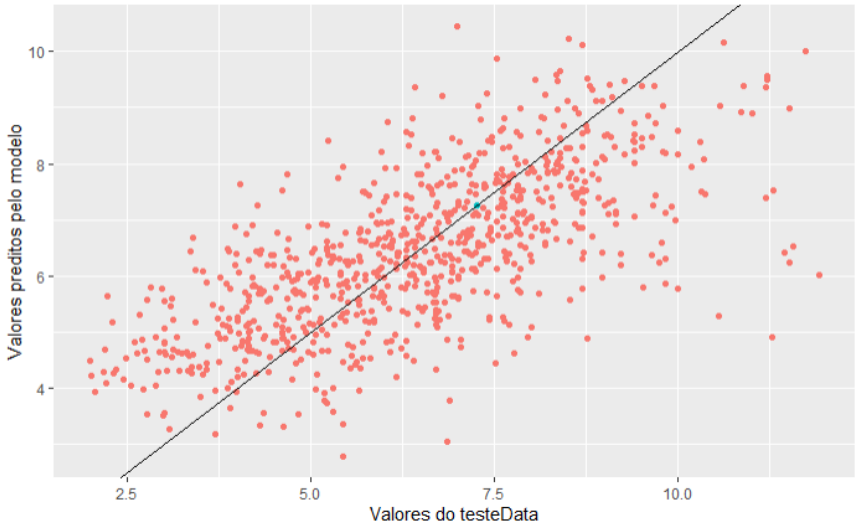
1.9.1 Regressão Linear com "svmLinear"

Fiz o treinamento com regressão linear com essas 77 variáveis, utilizando o método "svmLinear" e meu resultado foi:

1.9.1.1 Resultados

RMSE = 1.419529
RRSE = 0.7268206.
E meu gráfico ficou:

Figure 22: svmLinear



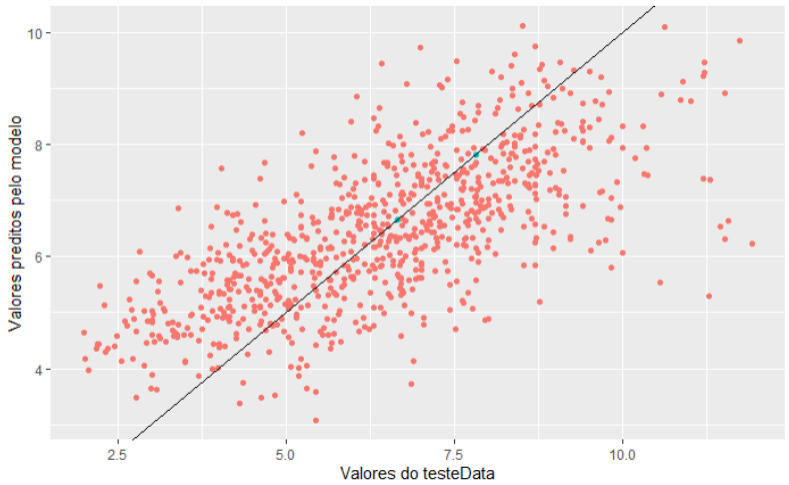
1.9.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.9.2.1 Resultados

RMSE = 1.414396.
RRSE = 0.7241922.
E meu gráfico ficou:

Figure 23: enet



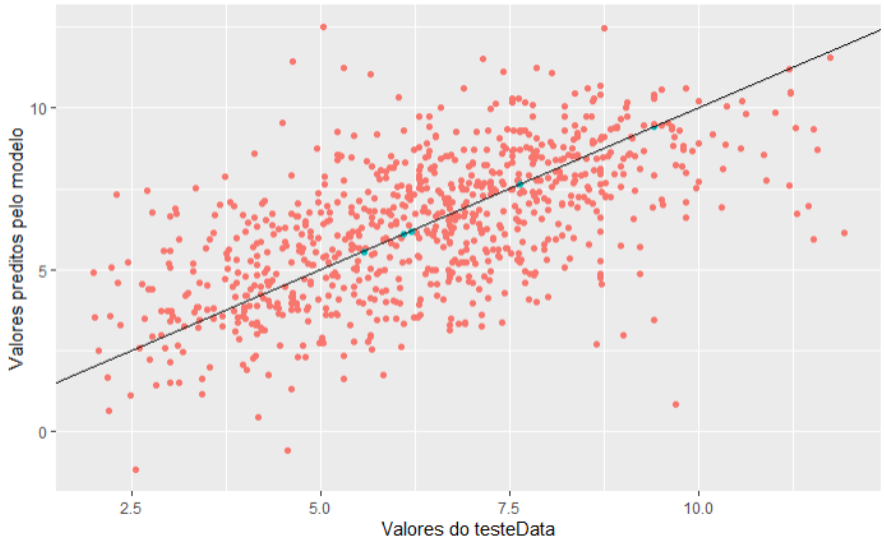
1.9.3 Rede Neural

E rodei também com uma Rede neural com 2 camadas ocultas de 50 neurônios cada e threshold de 0.1.

1.9.3.1 Resultados

RMSE = 1.85728.
RRSE = 0.9509557.
E meu gráfico ficou:

Figure 24: Rede Neural



1.10 Tentativa de obter melhores modelos

Como o *Random Forest Regression* me retornou o melhor resultado, eu decide partir dele pra tentar criar modelos mais precisos.

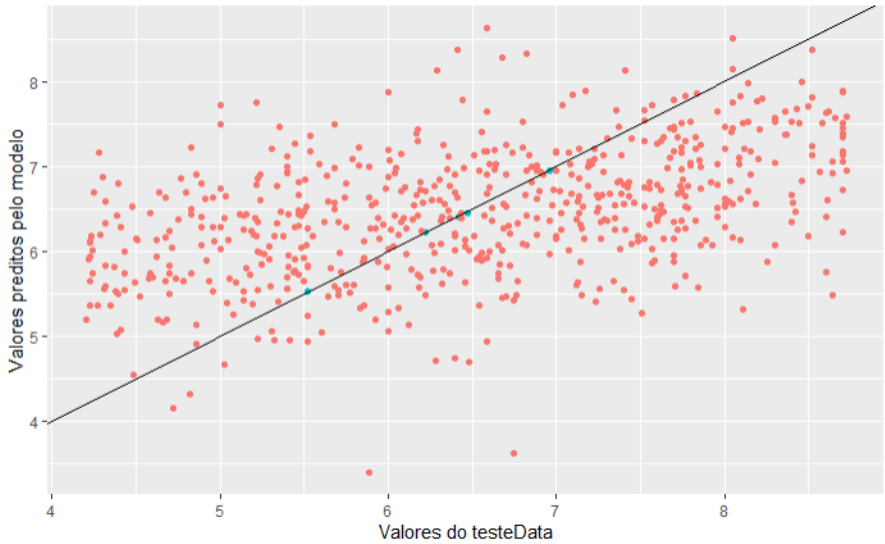
O que eu fiz foi ver em que faixa de dados o modelo convergia bem, decide que era em torno de 4.2 e 8.75

1.10.1 Regressão Linear com svmLinear

1.10.1.1 Resultados

RMSE = 1.081036.
RRSE = 0.8819661.
E meu gráfico ficou:

Figure 25: svmLinear



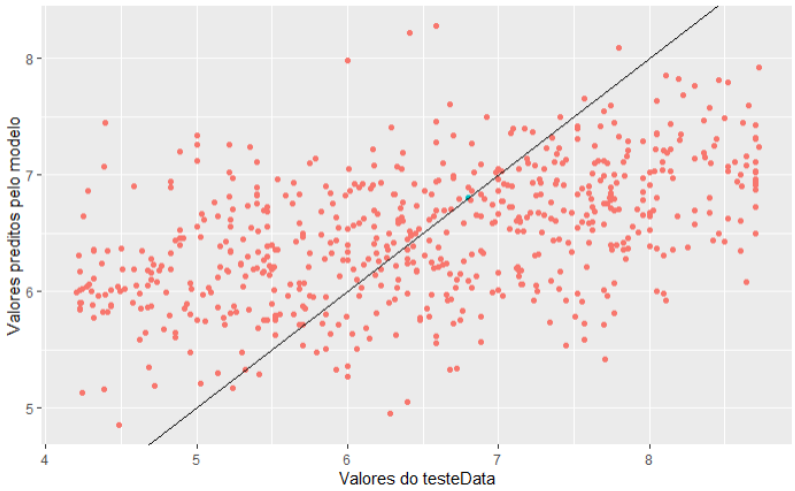
1.10.2 Regressão Linear com "enet"

Utilizando o método "enet" e meu resultado foi:

1.10.2.1 Resultados

RMSE = 1.069475.
RRSE = 0.872534.
E meu gráfico ficou:

Figure 26: enet



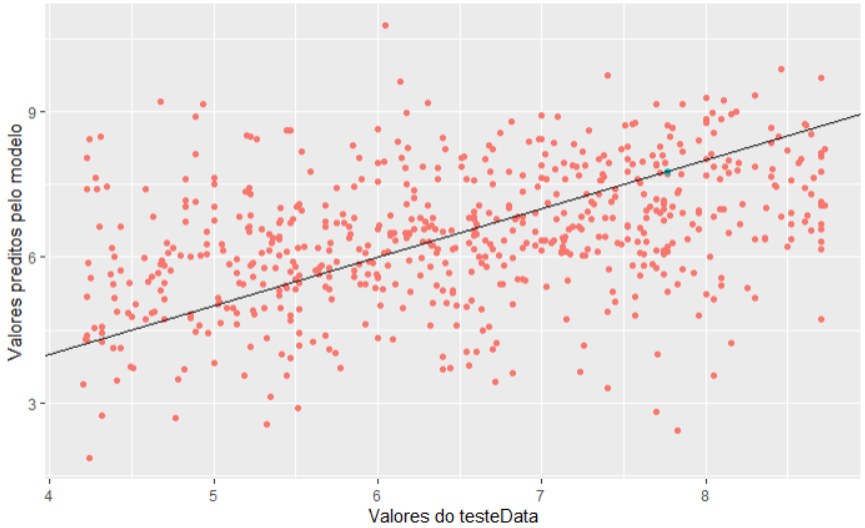
1.10.3 Rede Neural

E rodei também com uma Rede neural com 2 camadas ocultas de 50 neurônios cada e threshold de 0.1.

1.10.3.1 Resultados

RMSE = 1.459911.
RRSE = 1.191073.
E meu gráfico ficou:

Figure 27: Rede Neural



2 Dataset Fifa

Como o *dataset* utilizado anteriormente era difícil de entender, e a proposta do exercício era pegar um dataset que tínhamos conhecimento, então fiz com esse também.

Inicialmente o dataset é muito grande, ele tem dimensão inicial de 18208 x 84, e com dados fora de padrão, e também com muito valores faltantes. Sua atributos eram "ID Name Age Photo Nationality Flag Overall Potential Club Club Logo Value Wage Special Preferred Foot International Reputation Weak Foot Skill Moves Work Rate Body Type Real Face Position Jersey Number Loaned From Contract Valid Until Height Weight LS ST RS LW LF CF RF RW LAM CAM RAM LM LCM CM RCM RM LWB LDM CDM RDM RWB LB LCB CB RCB RB Crossing Finishing HeadingAccuracy ShortPassing Volleys Dribbling Curve FKAccuracy LongPassing BallControl Acceleration SprintSpeed Agility Reactions Balance ShotPower Jumping Stamina Strength LongShots Aggression Interceptions Positioning Vision Penalties Composure Marking StandingTackle SlidingTackle GKDiving GKHandling GKKicking GKPositioning GKReflexes Release Clause".

Como objetivo dessa analise, eu pretendo estimar o *"Value"* do jogador utilizando classificação. Eu decidi fazer com classificação, primeiramente porque eu usei regressão e redes neurais no anterior, então quis fazer diferente, e também porque eu tinha "classes" muito desbalanceadas, tem muitas instancias de jogadores com baixo valor e poucos com valores altos, então meus jogadores com valor alto seriam os dados discrepantes, mas eu não quero retirar eles, quero que ele consiga classificar, e com regressão, a curva de aprendizado dele seria linear, e quando chegasse em valores altos a curva de aprendizado não iria conseguir acompanhar o salto do gráfico pra cima, então se eu categorizar meus *"Value"* eu consigo fazer um aprendizado mais abrangente (por mais que eu diminua minhas possibilidades).

Primeiramente, eu removi diversos atributos que não contribuíam no meu modelo, como "photo, flag, body type, real face" entre vários outros.

Depois eu comecei a padronizar algumas variáveis, como por exemplo, minha variável resposta "*Value*" tinha K e M, então eu transformei em valores números, entre varias outras variáveis, outras variáveis que não tinham como padronizar eu simplesmente exclui elas.

Ainda assim, o meu dataset continua muito grande, então fui pra uma analise mais detalhada, e olhei como se comportavam os valores baixos, percebi que todos as instancias que tinha o valor menor que 3000 eram muito parecidas, praticamente iguais, mudavam pouca coisa, então eu decidi que iria excluir essas instancias, me restaram 1583 instancias, então eu reduzi 91% do meu dataset. Então meu dataset ficou 1582x44.

O meu ultimo tratamento nos dados foi criar classes de classificações, criei valores, Entre 5700 e 8500, Entre 8500 e 11500, Maior que 11500 e Menor que 5700.

Treinei, e a matriz de confusão foi:

Table 1: **Matriz de Confusão**

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	133	6	0	0
Entre 5700 e 8500	1	100	9	0
Entre 8500 e 11500	1	10	22	3
Maior que 11500	8	8	13	82

E minhas medidas de qualidade ficaram.

Figure 28: Medidas de Qualidade

```
               Accuracy : 0.851
                 95% CI : (0.8121, 0.8846)
    No Information Rate : 0.3611
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7919

 Mcnemar's Test P-Value : 0.0001529
```

Figure 29: Medidas de Qualidade das Classes

```
Statistics by Class:

                     Class: Entre 5700 e 8500 Class: Entre 8500 e 11500
Precision                            0.9091                    0.61111
Recall                               0.8065                    0.50000
F1                                   0.8547                    0.55000
Prevalence                           0.3131                    0.11111
Detection Rate                       0.2525                    0.05556
Detection Prevalence                 0.2778                    0.09091
Balanced Accuracy                    0.8848                    0.73011
                     Class: Maior que 11500 Class: Menor que 5700
Precision                            0.7387                 0.9568
Recall                               0.9647                 0.9301
F1                                   0.8367                 0.9433
Prevalence                           0.2146                 0.3611
Detection Rate                       0.2071                 0.3359
Detection Prevalence                 0.2803                 0.3510
Balanced Accuracy                    0.9357                 0.9532
```

2.1 Analise do resultados

Da pra ver que a maioria dos erros foram nas classes próximas, isso se da pelo corte seco, o limiar entre cada classe é 1, então se o calculo der 8501 ele iria entrar na classe 3, mas poderia também entrar na 2, isso se resolveria em partes, se eu utilizasse um sistema de classificação Fuzzy, pois ai eu teria a pertinência de ser de tal classe ou não, como eu não sei fazer em R e acho que nem é o foco da disciplina, eu deixei assim mesmo.

O erro mais evidente é o erro da classe maior que 11500, isso se deu por conta de termos poucos exemplos em que o valor é muito alto, e os exemplos que tem são *outliers*, então entendo que é difícil para o modelo, pegar esses poucos casos muito alto, e transformar em valores aceitaveis que se encaixam no modelo.

3 Link para os códigos em Python

https://colab.research.google.com/drive/1v8csTdspsn03R3dSlLzRXGNNw6izYZ0V?usp=sharing

4 Link para os datasets

https://drive.google.com/file/d/1iEmNeD5Wcxf0BTyDTkFgjD5xWV3PVdOi/view?usp=sharing