Detectando questões duplicadas: Quora Questions Pairs

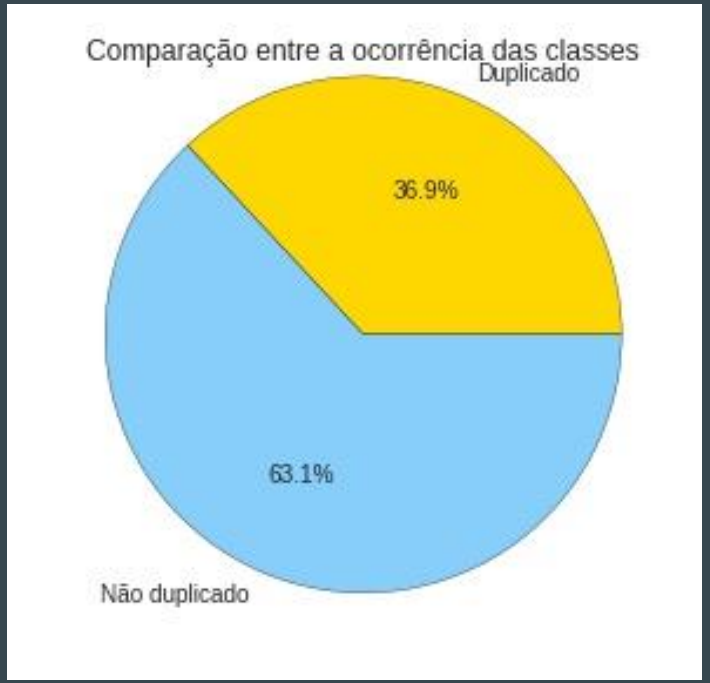
...

Airine Carmo e Christian Cardozo

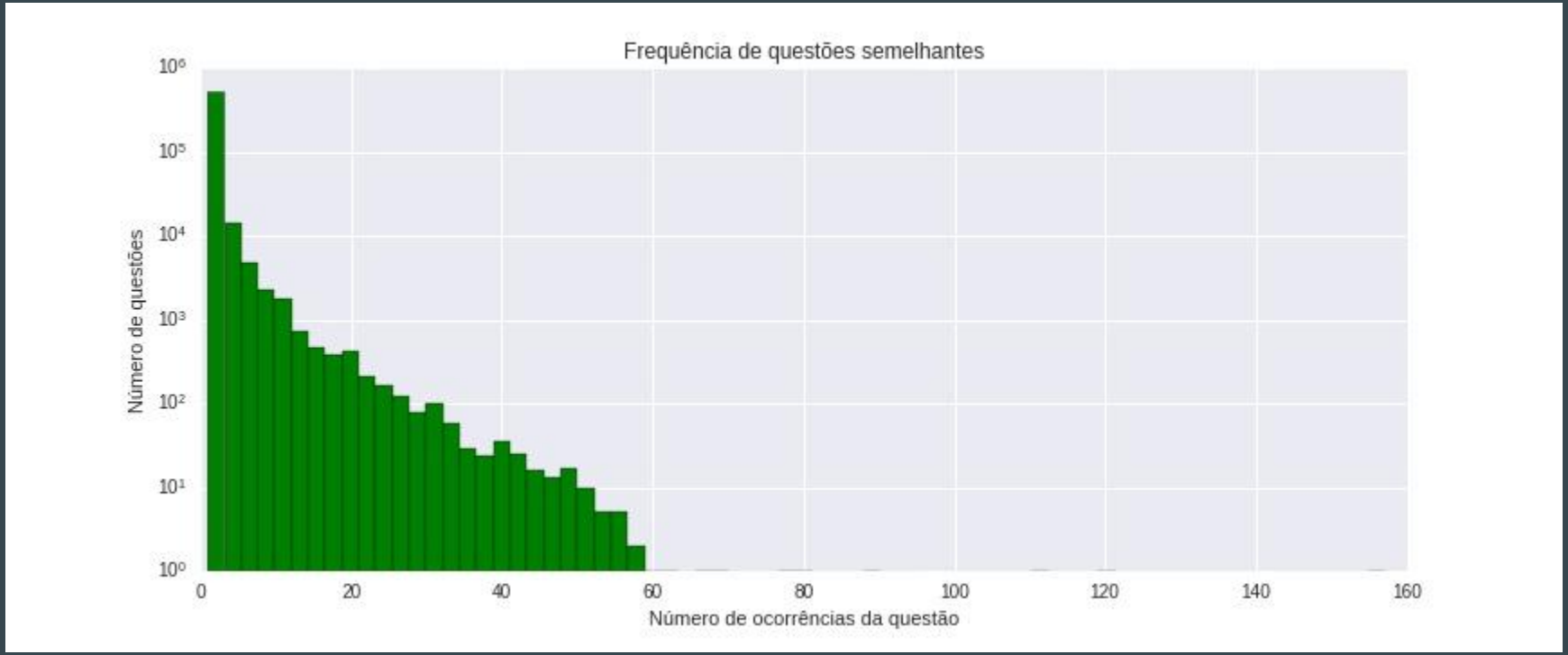
Sumário

Análise dos dados

- Conjunto de dados do treinamento
- 404.290 registros
- 255.027 registros são da classe ‘0’
- 149.263 são da classe ‘1’.



Análise dos dados



Pré-processamento

1. Transformação e Tokenização
2. Remoção de Stop Words
3. Stemming
4. Construção da matriz de presença e n-gramas
5. Redução de dimensionalidade
6. Combinando features

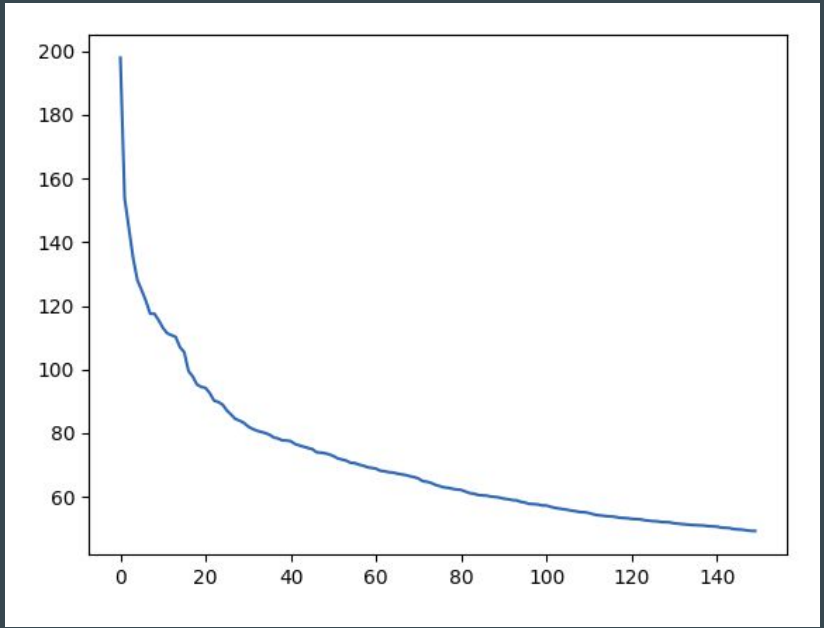
Construção da matriz de presença e n-gramas

- Foram geradas todas as combinações presentes no texto de 1-gramas, 2-gramas e 3-gramas
- Foram geradas 3 matrizes de presença onde:
  - A primeira matriz, X1, se refere à presença de n-gramas na primeira questão do par.
  - A segunda, X2, se refere à presença de n-gramas na segunda questão do par.
  - A terceira, X3, é feita utilizando a presença de n-gramas que aparecem nas duas perguntas.

Matriz	Linhas	Colunas
X1	404290	1.616.073
X2	404290	1.675.423
X3	404290	24.197

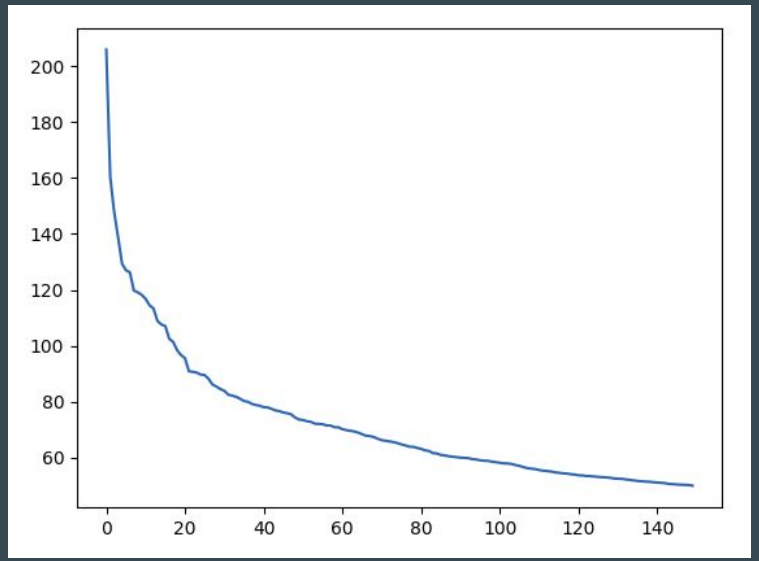
Redução de dimensionalidade (SVD)

- Mostra os primeiros 150 autovalores da matriz diagonal Sigma gerada pelo SVD das matrizes X1, X2 e X3.
- Foi definido um ponto de corte K=50 para todas as matrizes.
- SVD de X1, X2 e X3 gera 3 matrizes que concatenamos em uma matriz, X', com 150 colunas

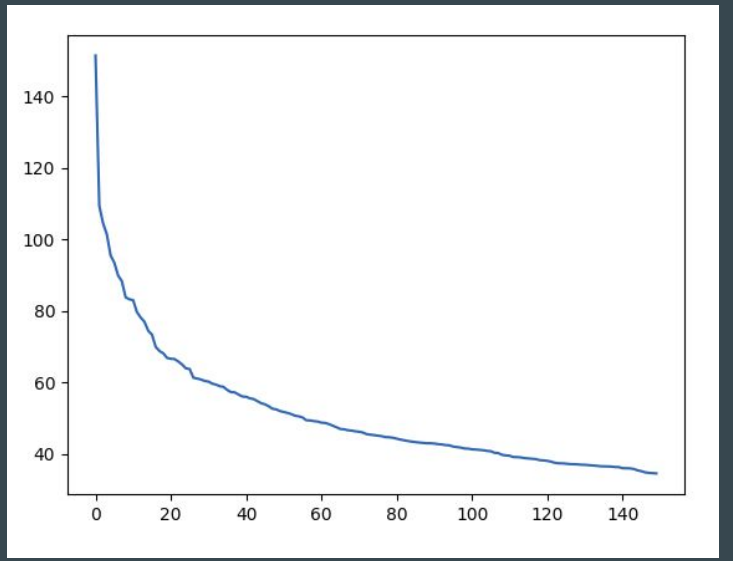


Valores da matriz diagonal Sigma de X1 gerada pelo SVD

Redução de dimensionalidade (SVD)



Valores da matriz diagonal Sigma de X2 gerada pelo SVD



Valores da matriz diagonal Sigma de X3 gerada pelo SVD

Combinando features

Além das 150 geradas no SVD, são geradas 6 features adicionais sendo:

1. Diferença de quantidade de tokens entre as questões dividido pela quantidade de tokens na questão 1
2. A diferença de quantidade de tokens entre as questões dividido pela quantidade de tokens na questão 2
3. Distância de Jaccard entre os conjuntos de tokens das questões
4. Distância de Leveinshtein entre os conjuntos de tokens das questões
5. Quantidade de tokens que aparecem nas duas perguntas dividido pela quantidade de tokens na questão 1
6. Quantidade de tokens que aparecem nas duas perguntas dividido pela quantidade de tokens na questão 2

Totalizando 156 features para a criação da matriz final com 404.290 linhas.

Experimento preliminar

Experimentos empíricos mostraram que construir a matriz X’ com as features semânticas após o SVD melhora o resultado.

Foram realizadas com as seguintes premissas:

- Executar com 50% da base: 202.145 exemplos
- Criar uma matriz X auxiliar com menos dimensões, executando o SVD com ponto de corte igual à 15.
- Executar XGBoost com os parâmetros: learning_rate=0.15, n_estimators=170 e max_depth=6.

Features semânticas	Log loss	Acurácia
Não	0.434926784513	78.26%
Concatenadas antes do SVD	0.409501988533	79.6%
Concatenadas após o SVD	0.378485486853	81.98%

Experimentos

Foram utilizados três modelos de aprendizado de máquina sendo eles:

- Naive Bayes do scikit-learn
- Redes neurais do scikit-learn
- Xgboost

Seguindo estes parâmetros, para todos:

- Utilizando o conjunto de exemplos dividido em 10% para validação e 90% para treinamento.
- Método de K-Fold Cross Validation com K=3
- Duas métricas de avaliação sendo: acurácia e log loss.

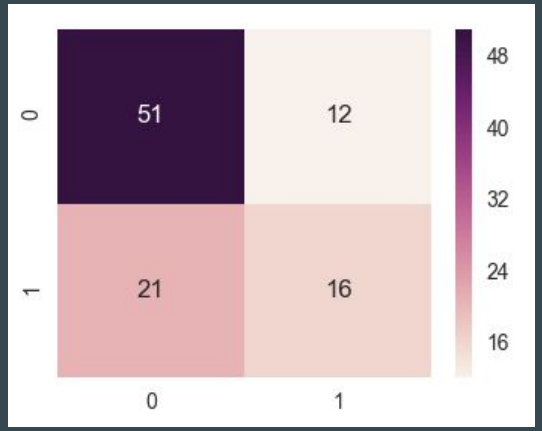
Naive Bayes

Naive Bayes Gaussiano		
	Acurácia	Log loss
Conjunto de treinamento	66.9841% (+/- 0.0569%)	5.9239 (+/- 0.0180)
Conjunto de validação	67.0459%	5.9146

Naive Bayes Bernoulli		
	Acurácia	Log loss
Conjunto de treinamento	62.2177% (+/- 0.0519%)	0.6449 (+/- 0.0010)
Conjunto de validação	61.8466%	0.6482

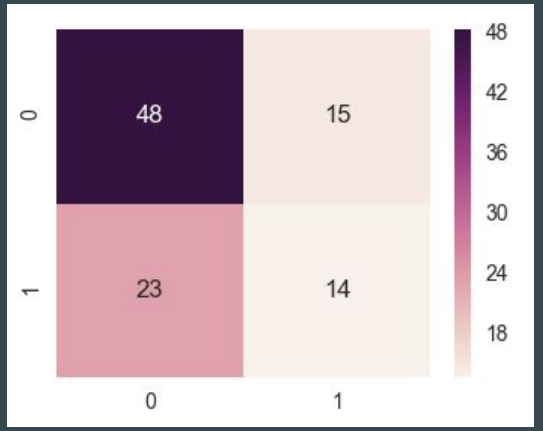
Naive Bayes Multinomial		
	Acurácia	Log loss
Conjunto de treinamento	66.7395% (+/- 0.0649%)	0.5955 (+/- 0.0003)
Conjunto de validação	66.4275%	0.5965

Naive Bayes - matriz de confusão



Naive Bayes Gaussiano

Naive Bayes Multinomial



Naive Bayes Bernoulli

Onde o eixo horizontal são as classes preditas pelos classificadores, e o eixo vertical, as classes reais no dataset.

Rede Neural

Foram testadas as seguintes variações:

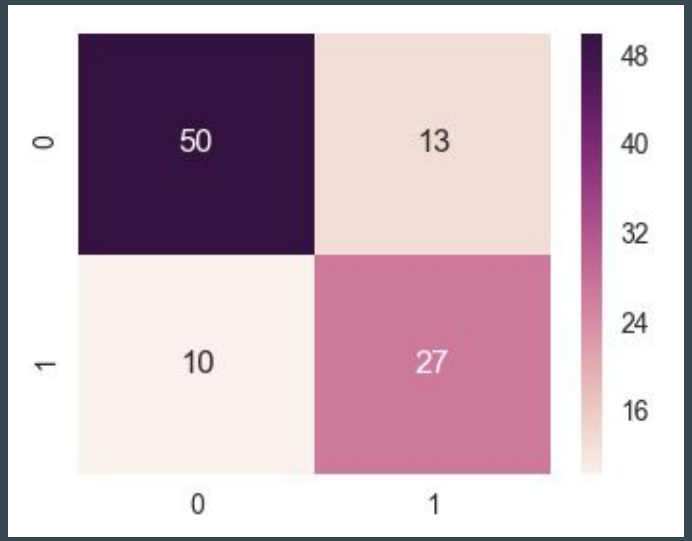
- Quantidade de camadas: 1 e 2
- Quantidade de neurônios por camada: 10, 30 e 50
- Função de ativação: relu, tangente hiperbólica e logística
- Ajuste de pesos: adam, lbfgs e sgd

Melhor desempenho

Quantidade de camadas	2
Quantidade de neurônios por camada	50
Função de ativação	relu
Ajuste de pesos	adam

	Acurácia	Log loss
Conjunto de treinamento	76.669% (+/- 0.2157%)	0.4497 (+/- 0.0018)
Conjunto de validação	76.5267%	0.4538

Rede Neural - matriz de confusão



Onde o eixo horizontal são as classes preditas pelos classificadores, e o eixo vertical, as classes reais no dataset.

Xgboost

XGBoost é uma biblioteca de gradiet boosting otimizada e muito eficiente

Parâmetros iniciais:

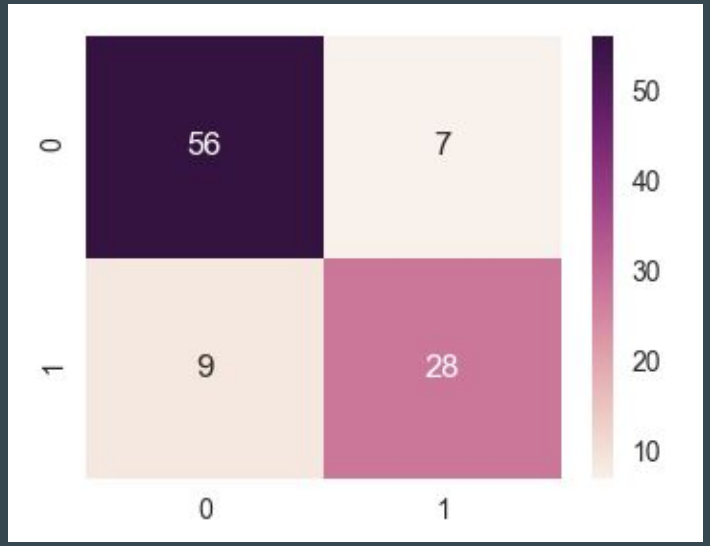
- learning_rate: 0.15
- n_estimators: 170
- max_depth: 6

Melhor desempenho

learning_rate	0.15
n_estimators	1000
max_depth	10

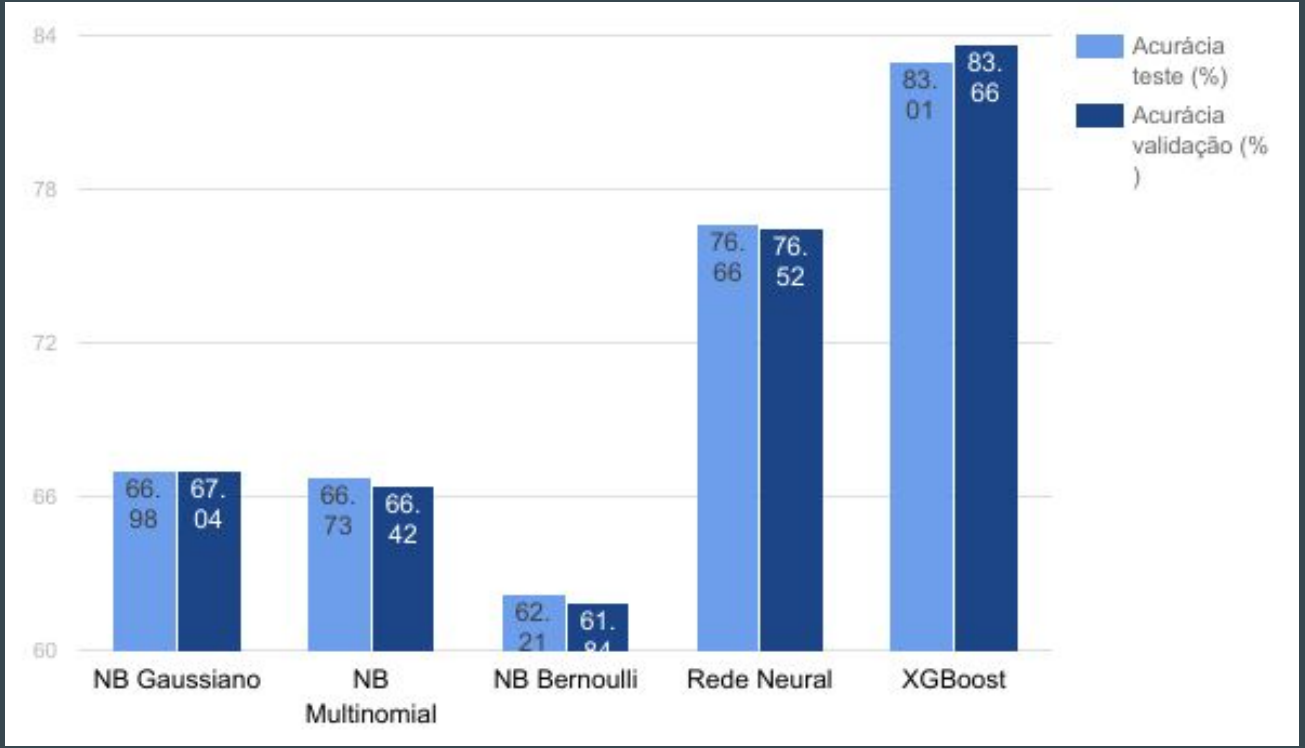
	Acurácia	Log loss
Conjunto de treinamento	83.0129% (+/- 0.1033%)	0.3822 (+/- 0.0018)
Conjunto de validação	83.6676%	0.3613

Xgboost - matriz de confusão

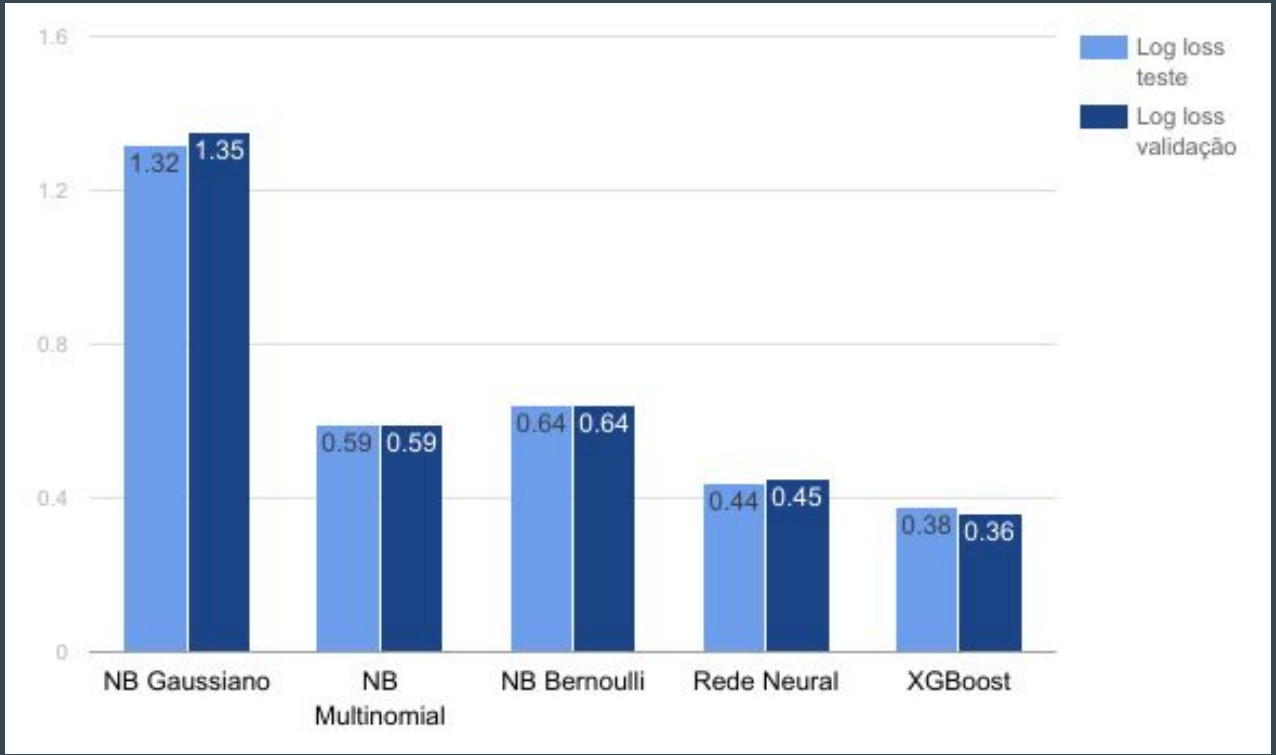


Onde o eixo horizontal são as classes preditas pelos classificadores, e o eixo vertical, as classes reais no dataset.

Resultados e considerações finais



Resultados e considerações finais



Referências

Quora Questions Pairs, 2017. Disponível em: https://www.kaggle.com/c/quora-question-pairs. Acesso em: 15 de mai. 2017.

Kaggle, 2017. Disponível em: https://www.kaggle.com/. Acesso em: 15 de mai. 2017.

Quora, 2017. Disponível em: https://www.quora.com/. Acesso em: 15 de mai. 2017.

Python Software Foundation, 2017. Disponível em: https://www.python.org/. Acesso em: 15 de mai. 2017.

Scikit-learn: Machine Learning in Python, 2017. Disponível em: http://scikit-learn.org/stable/. Acesso em: 15 de mai. 2017.

Natural Language Toolkit, 2017. Disponível em: http://www.nltk.org/. Acesso em: 15 de mai. 2017.

GitHub, 2017. Disponível em: https://github.com/chriiscardozo/QuoraQuestionPair/. Acesso em: 15 de mai. 2017.

Stemmers, 2017. Disponível em: http://www.nltk.org/howto/stem.html. Acesso em: 15 de mai. 2017.

XGBoost Documents, 2017. Disponível em: https://xgboost.readthedocs.io/en/latest/. Acesso em: 15 de mai. 2017.

Complete Guide to Parameter Tuning in XGBoost (with codes in Python), 2017. Disponível em: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/, 2017. Acesso em: 15 de mai. 2017.

Jaccard index, 2017. Disponível em: https://en.wikipedia.org/wiki/Jaccard_index. Acesso em: 15 de mai. 2017.

Leveinshtein distance, 2017. Disponível em: https://en.wikipedia.org/wiki/Levenshtein_distance. Acesso em: 15 de mai. 2017.