NANODEGREE ENGENHEIRO DE MACHINE LEARNING

PROJETO FINAL

Arthur Pimenta

30 de maio de 2019

DEFINIÇÃO

VISÃO GERAL DO PROJETO

Processamento de língua natural (PLN) é uma subárea da ciência da computação, inteligência artificial e da linguística que estuda os problemas da geração e compreensão automática de línguas humanas naturais e está rapidamente se tornando uma habilidade essencial para as organizações modernas ganharem vantagem competitiva. Fez-se uma ferramenta essencial para muitas novas funções de negócios, de chatbots, sistemas de perguntas, análise de sentimentos, detecção de ameaças, compreensão de documentos e análises em conteúdo não estruturado. Em uma plataforma de perguntas podem haver várias com a mesma intenção fazendo com que os solicitantes passem mais tempo procurando a melhor resposta para a pergunta e com que os redatores sintam que precisam responder a várias versões da mesma pergunta.

Neste projeto apresentamos um modelo de classificação para encontrar a similaridade entre duas perguntas no qual utilizaremos o conjunto de dados fornecido pelo desafio Kaggle "Quora Question Pairs" e bibliotecas como Gensim para o processamento de texto.

DESCRIÇÃO DO PROBLEMA

Utilizando técnicas de processamento de linguagem natural nosso objetivo neste projeto é prever quais dos pares de perguntas fornecidas contêm duas perguntas com o mesmo significado, seguindo uma abordagem clássica nosso trabalho será decomposto em várias etapas: 1. Extrair informações significativas do texto bruto e decompor o conteúdo de cada pergunta; 2. Feature Engineering; 3. Treinar um classificador capaz de determinar a similaridade entre os pares de perguntas; 4. Submeter o resultado para a plataforma Kaggle.

MÉTRICAS

A submissão é avaliada através da função *log-loss* entre os valores previstos e a verdade básica, a mesma leva em conta a incerteza de sua previsão com base em quanto ela varia do rótulo real.

Fig. 1 - Função de *Log Loss*

$$-(y\log(p)+(1-y)\log(1-p))$$

Para este caso binário *p* é a probabilidade do exemplo pertencer à classe, e *y* é o valor real da variável dependente.

ANÁLISE

EXPLORAÇÃO DOS DADOS

O conjunto de dados da competição *Quora Question Pairs* contêm um par de perguntas e um rótulo de verdade marcado por especialistas humanos, definindo se os pares de perguntas são duplicados ou não. É interessante notar que esses rótulos são subjetivos, o que significa que nem todos os especialistas podem concordar se o par de perguntas é duplicado.

O conjunto de dados têm 404290 pares de perguntas rotuladas sendo que 111780 perguntas aparecem mais de uma vez, portanto num total de 808580 utilizaremos apenas 537933 para treinar o classificador. Em todo o conjunto temos 3 dados faltantes que precisam ser tratados, julgando pelo tamanho do conjunto de dados essas entradas podem ser removidas.

Fig. 2 - Amostra dos dados

id	qid1	qid2	question1	question2	is_duplicate
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0

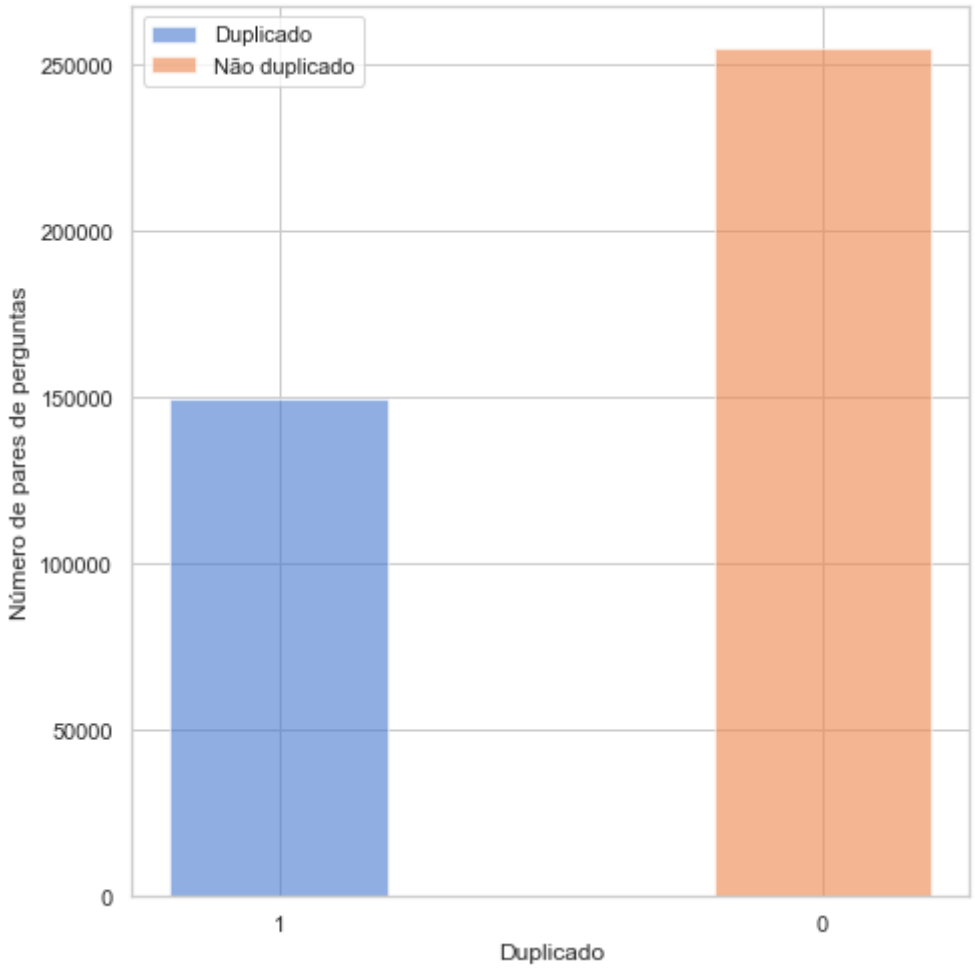
Os campos que constituem o conjunto de dados são:

- **id:** identificação única para cada par de perguntas.
- **qid{1, 2}:** identificação única para cada pergunta no par.
- **question{1, 2}:** conteúdo de texto bruto de cada pergunta do par.
- **is_duplicate:** rótulo que estamos tentando prever, diz se as duas perguntas são duplicadas (1) ou não (0).

VISUALIZAÇÃO EXPLORATÓRIA

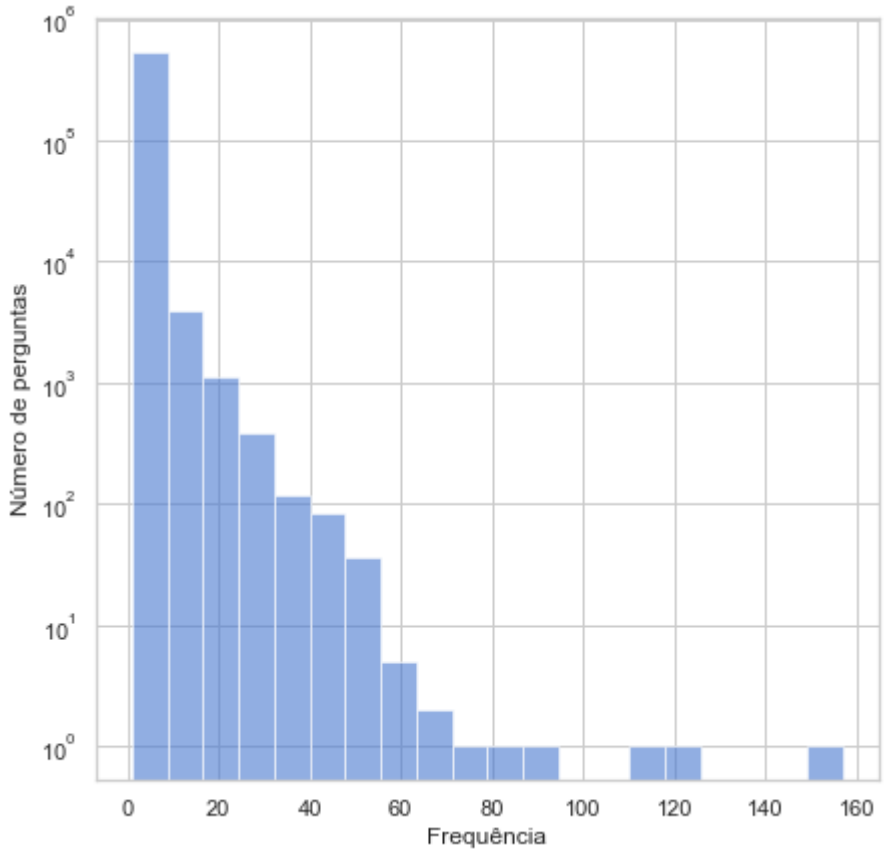
A *Fig. 3* mostra como o conjunto de dados é desequilibrado, sendo 36,92% dos pares de perguntas classificados como duplicados, a maioria dos algoritmos de aprendizado de máquina funcionam melhor quando o número de amostras em cada classe é aproximadamente igual, neste caso iremos aplicar a técnica de reamostragem *oversampling* para equilibrar as classes.

Fig. 3 - Número de observações (pares de perguntas) por classe.



Em um conjunto de dados de perguntas é de se esperar que algumas sejam mais frequentes que outras, através da *Fig. 4* podemos observar que a maioria das perguntas aparecem com baixa frequência, para ser mais preciso 20,77% das perguntas se repetem mais de uma vez, sendo que as perguntas duplicadas não farão parte do conjunto de treino para o classificador.

Fig. 4 - Número de perguntas por frequência de aparição



ALGORITMOS E TÉCNICAS

O classificador é o XGBoost (eXtreme Gradient Boosting) que se tornou particularmente popular por tem sido o algoritmo vencedor em várias competições recentes no Kaggle, é um algoritmo baseado em árvores com aumento de gradiente onde o aumento de gradiente é um algoritmo de aprendizagem supervisionada que tenta prever com precisão uma variável de destino.

Antes de executar o XGBoost, devemos definir três tipos de parâmetros: *general parameters, booster parameters and task parameters*.

- **General parameters:** referem-se a qual booster que estamos usando para fazer o reforço, geralmente árvore ou modelo linear, utilizaremos árvore.

- **Booster parameters:** dependem de qual booster você escolheu, baseado na escolha anterior utilizaremos os seguintes parâmetros *eta* e *max_depth.*

- **Learning task parameters:** decidem sobre o cenário de aprendizado. Por exemplo, as tarefas de regressão podem usar parâmetros diferentes com tarefas de classificação. Como já mencionado estamos diante de uma tarefa de classificação binária avaliada com a função de log loss, para isso precisamos determinar os parâmetros: *objective* e *eval_metric*.

- **Command line parameters:** referem-se ao comportamento da versão CLI do XGBoost.

Antes de treinar o classificador precisamos extrair informações úteis dos pares de perguntas e convertê-las em variáveis de entrada (vetores) para o treinamento, processo chamado de vetorização.

BENCHMARK

Uma referência para este projeto será o Kernel Data Analysis & XGBoost Starter (0.35460 LB) submetido na plataforma Kaggle onde obteve uma pontuação de 0.35460 na tabela de classificação, não esperamos superar este resultado.

METODOLOGIA

PRÉ-PROCESSAMENTO DE DADOS

Nesta etapa as perguntas foram transformados para a forma numérica. O conteúdo de cada uma é decomposto em termos e a frequência de cada palavra reduzindo o vocabulário e tornando os dados menos esparsos, característica conveniente para o processamento computacional. Utilizando as biblioteca Gensim e NLTK os dados de texto foram processados da seguinte forma:

1. Converter todas as letras para minúscula
2. Substituir caracteres de pontuação por espaços
3. Remover dígitos
4. Remover caracteres não alfabéticos
5. Remover os caracteres de espaço em branco repetidos
6. Remover *stopwords*, porém as palavras *what*, *which*, *who*, *whom*, *when*, *where*, *why* e *how* foram mantidas por acreditarmos em suas relevâncias para o problema
7. Remover palavras com comprimento menor que 3
8. *Stemming*

Fig. 5 - Amostra de dados da *Fig. 2* após as etapas acima

id	qid1	qid2	question1	question2	is_duplicate
0	1	2	what step step guid invest share market india	what step step guid invest share market	0
1	3	4	what stori kohinoor koh noor diamond	what would happen indian govern stole kohinoor koh noor diamond back	0
2	5	6	how increas speed internet connect us vpn	how internet speed increas hack dn	0
3	7	8	why mental lone how solv	find remaind when math math divid	0
4	9	10	which on dissolv water quikli sugar salt methan carbon oxid	which fish would surviv salt water	0

9. Converter uma coleção de documentos de texto em uma matriz de contagens de *token*
10. Transformar uma matriz de contagem para uma representação normalizada de *tf-idf*

FEATURE ENGINEERING

Feature engineering é uma parte muito importante no desenvolvimento de soluções para problemas de processamento de linguagem natural, sabendo disso as seguintes variáveis foram criadas para melhorar a capacidade de predição do classificador:

- **dist:** distância euclidiana entre os vetores dos pares de pergunta
- **q{1,2}_words:** número de palavras presente em cada pergunta dos pares
- **q{1,2}_len:** número de caracteres presente em cada pergunta dos pares
- **word_share:** percentual de palavras em comum entre os pares de pergunta
- **word_share_weight:** percentual do peso das palavras em comum entre os pares de pergunta baseado na vetorização normalizada do *tf-idf.*

Fig. 6 - Pares de perguntas pela distância euclidiana, após o pré-processamento dos dados calculamos a distância utilizado os vetores resultante para cada par de pergunta e como era previsto os pares duplicados se concentram na região de menor distância euclidiana entre as perguntas.

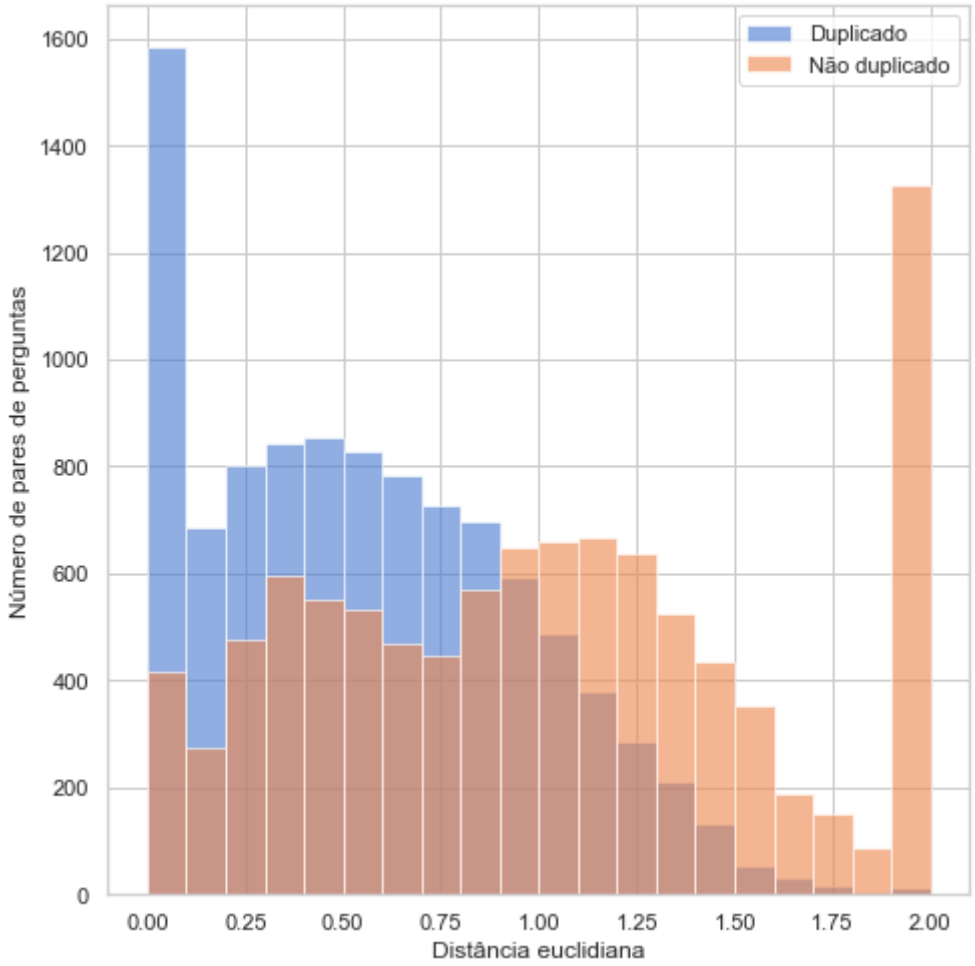
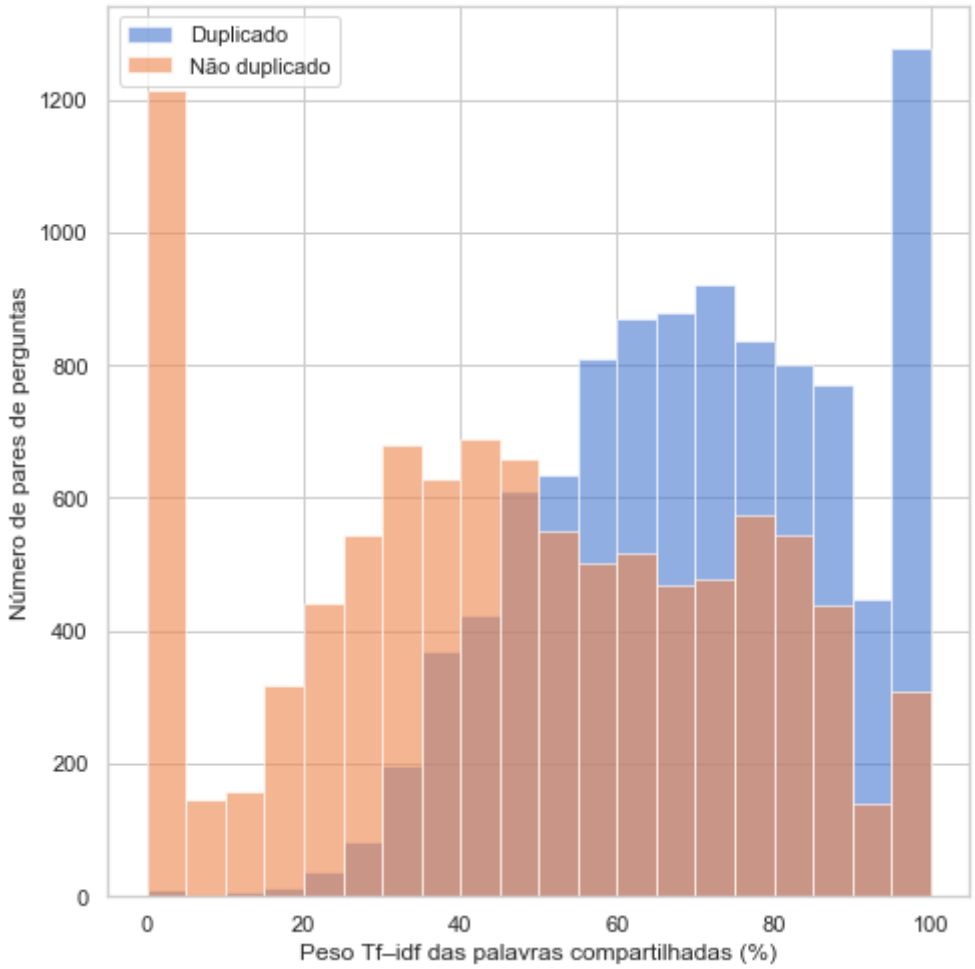


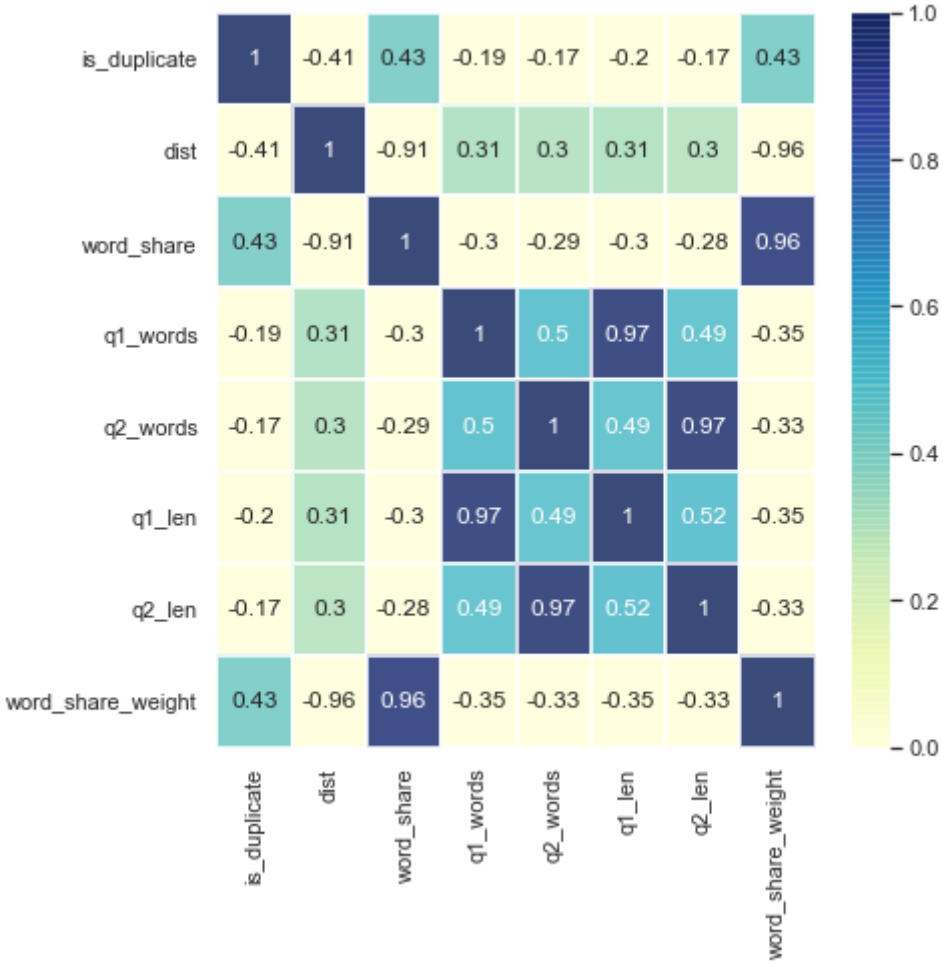
Fig. 7 - Pares de perguntas pelo peso percentual das palavras compartilhadas, partindo do pressuposto que as perguntas duplicadas teriam mais palavras em comum do que as não duplicadas, utilizando o peso das palavras (features) definido pela vetorização no pré-processamento para calcular qual o peso das palavras compartilhadas entre as perguntas do par e confirmando nossa teoria os pares duplicados se concentram mais perto do 100% onde todas as palavras de peso estão presentes em ambas as perguntas.



Por fim, calculamos a correlação das variáveis criadas para entendermos como elas estão relacionados entre si e com a variável de destino. A correlação pode ser positiva (aumento no valor da variável aumenta o valor da variável de destino) ou negativo (aumento no valor da variável diminui o valor da variável de destino).

O mapa de calor da *Fig. 8* facilita a identificação de quais variáveis estão mais relacionados à variável de destino.

Fig. 8 - Mapa de calor da correlação das variáveis criadas durante o *feature engineering*, podemos observar que 43% da variável destino pode ser explicada pelas variáveis criadas *word_share* e word_share_weight, porém ambas apresentam uma correlação de 96% isso significa que podemos facilmente remover uma das duas do nosso conjunto de treinamento sem sofrer grandes efeitos no resultado final.



IMPLEMENTAÇÃO

Os requisitos de software para a implementação são os seguintes:

- Python 3.x
- numpy >= 1.16.2
- pandas >= 0.22.0
- seaborn >= 0.9.0
- scipy >= 1.0.0
- scikit-learn >= 0.19.1
- xgboost == 0.90

- nltk >= 3.2.5
- gensim >= 3.7.3

Inicialmente para combater o desequilíbrio do conjunto de dados e gerar novas amostras da classe sub-representada utilizamos a estratégia (naive random over-sampling) mais simples que é gerar novas amostras por amostragem aleatória com a substituição das amostras atuais disponíveis, em seguida o classificador XGBoost foi treinado utilizando os dados pré processados sem considerar as novas variáveis (features engineered), e por fim, o mesmo modelo foi treinando considerando as novas variáveis e ambos avaliados a partir da submissão das predição do conjunto de dados de teste na plataforma Kaggle.

Os hiperparâmetros do modelo final foram os seguintes:

- **Booster parameters:**
  - eta**:** 0.02 (taxa de aprendizado)
  - max_depth**:** 4 (profundidade máxima de uma árvore)
- **Learning task parameters:**
  - objective**:** binary:logistic (classificação binária)
  - eval_metric**:** logloss (métrica)

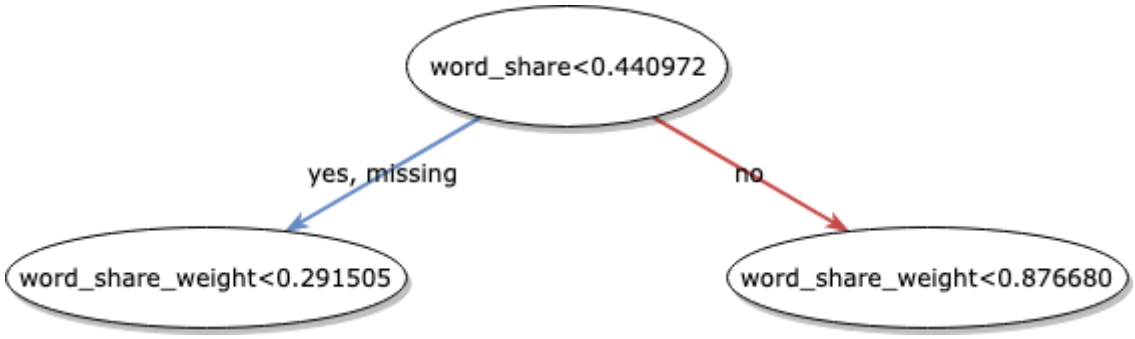
REFINAMENTO

Como mencionado na seção anterior o modelo inicialmente foi treinado sem as variáveis descritas na seção features engineering no qual recebemos um score de **0.53800** ao submetermos os resultados, em ordem de aperfeiçoar o modelo incluímos depois as variáveis (features engineered) além de explorarmos a flexibilidade e variedade de hiperparâmetros do XGBoost para ajustar o modelo a um cenário mais apropriado que resultou em um score de **0.45549** no Kaggle. Dentre os hiperparâmetros nós otimizamos os seguintes *booster parameters* utilizando o *Grid Search*:

- **eta**
  - Torna o modelo mais robusto diminuindo os pesos em cada etapa.
- **max_depth**
  - Utilizado para controlar o overfitting
  - Uma profundidade maior permitirá que o modelo aprenda relações muito específicas para uma determinada amostra

Fig. 9 - Entrada da primeira árvore de decisão do modelo, uma única árvore não é bom em prever, mas é muito interpretável, os ramos nos diz o "porquê" de cada previsão. É interessante notar como as variáveis adicionadas (features engineered) se tornaram os nós de entrada no modelo e

considerando o aumento da avaliação recebida pelo modelo refinado é notável a importância do feature engineering para o problema.



RESULTADOS

MODELO DE AVALIAÇÃO E VALIDAÇÃO

No geral o resultado foi satisfatório, embora não exploramos por completo os hiperparâmetros e outras possíveis novas variáveis por conta do custo computacional, contudo o modelo performou bem obtendo uma precisão de 80.72% nos dados de validação e os hiperparâmetros escolhidos foram os que entregaram o melhor resultado dentre todas as combinações testadas, em seguida para testar a robustez do modelo otimizado os mesmo foi avaliado pela submissão dos resultados enviados à respectiva competição no Kaggle no qual recebeu um score de **0.45549** em classificar mais de 2 milhões de pares de perguntas presentes no conjunto de dados de teste.

JUSTIFICATIVA

Se compararmos a avaliação final recebida com a do projeto de Benchmark e considerarmos a diferença de experiência existente pode-se considerar que o objetivo foi atingido. A avaliação final não supera a nossa referência (0.35460), sendo que o ideal é o mais próximo possível de zero, mas podemos dizer que é encorajadora e que as decisões tomadas foram no mínimo consistentes.

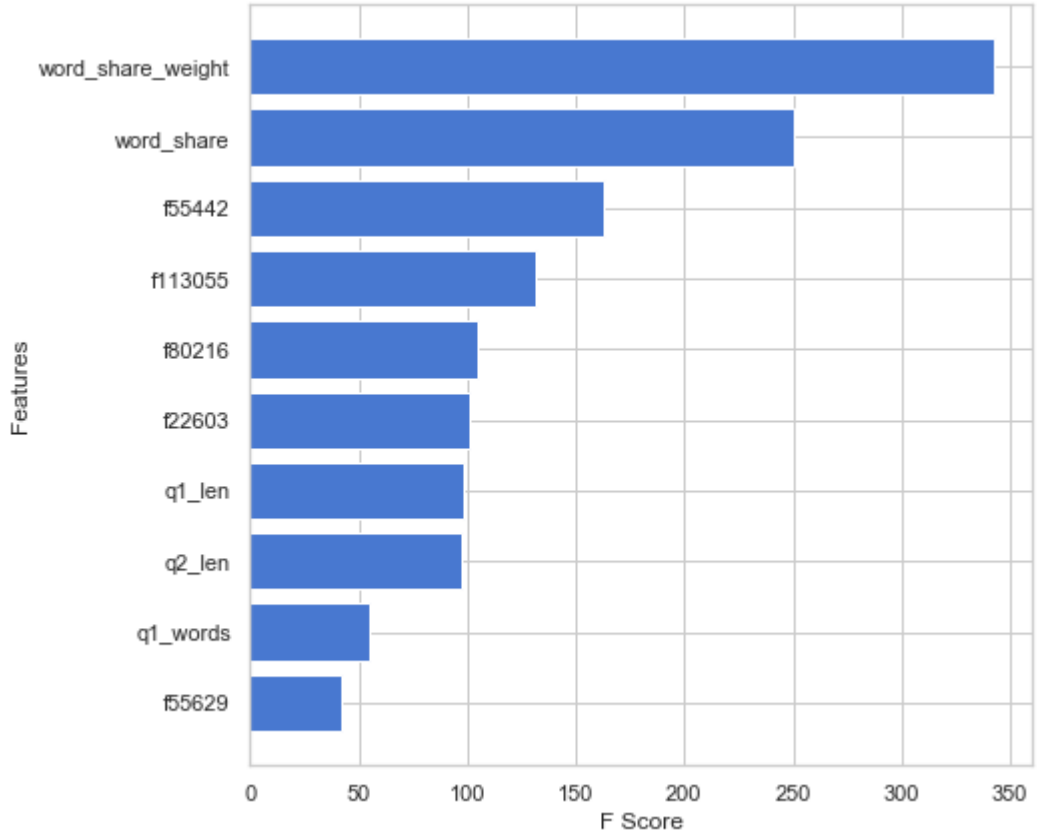
Para entendermos o quão bom é este resultado em termos práticos é um pouco mais complicado, precisa-se levar em conta as diferentes classes e o desequilíbrio dos dados, de modo geral este valor é uma medida de incerteza da precisão.

CONCLUSÃO

FORMA LIVRE DE VISUALIZAÇÃO

A partir dos ganhos descritos na seção de refinamento percebemos a importância em utilizar boas técnicas de feature engineering e a capacidade de extrair informações significativa a partir dos dados de texto não estruturados, que não é uma tarefa fácil, sabendo utilizamos uma característica oferecida pelo XGBoost que nos permite visualizar a importância de cada variável no modelo tornando-se um ótimo complemento para analisarmos a qualidade das variáveis criadas na etapa de feature engineering.

Fig. 10 - Feature importance, classificação em ordem decrescente de importância da variáveis do modelo, onde das seis criadas apenas uma não se encontra dentre as dez mais importantes segundo o critério de avaliação do modelo, ou seja, praticamente todas as variáveis extraídas do feature engineering tiveram um impacto positivo na capacidade de predição do classificador.



Além disso podemos utilizar do gráfico acima para reduzir a dimensionalidade do problema removendo as variáveis redundantes ou irrelevantes. Reduzir o número de variáveis por meio da seleção de recursos garante que o treinamento precise de menos memória e poder computacional, o que implica em um menor tempo de treinamento e pode ajudar a reduzir as chances de overfitting. A simplificação dos dados de treinamento também torna o modelo mais fácil de interpretar, o que pode ser importante ao justificar a tomada de decisões no mundo real.

REFLEXÃO

O desenvolvimento do projeto pode ser dividido em algumas etapas:

- Entendimento dos dados e do problema
- Técnicas de processamento de linguagem natural para extração de informações
- Escolha do algoritmo de aprendizado supervisionado
- Treinamento e otimização do modelo
- Submissão e avaliação dos resultados

Dentre os inúmeros algoritmos e técnicas de aprendizado de máquina disponíveis foi desafiador entender suas características, vantagens, desvantagens e diferentes aplicações para melhor adaptarmos a nossa tarefa.

Além disso, uma das dificuldades na implementação do modelo foi a limitação computacional, por exemplo, a variável *distância euclidiana* descrita na seção features engineering não foi utilizada para treinar o classificador final devido ao elevado custo computacional para ser calculada, no conjunto de dados de teste levou em média mais de uma hora para ser processada o que inviabilizou sua entrada visto que para otimizar o modelo o mesmo precisava ser executado múltiplas vezes.

MELHORIAS

Além dos métodos para representar palavras de maneira numérica aqui apresentados existem diversos outros que podem entregar resultados melhores como, por exemplo, o Word2vec que é uma estrutura de rede neural para gerar tais representações numéricas chamadas de *word embedding*. Tal método foi introduzido pela primeira vez no artigo *Efficient Estimation of Word Representations in Vector Space* por Mikolov et al., 2013 e provou ser muito bem sucedido em conseguir a incorporação de palavras que poderia ser usada para medir semelhanças sintáticas e semânticas entre elas. Assim, abrimos um leque de opções para melhorar o nosso resultado a partir do modelo Word2vec pré-treinado disponibilizado pela Google, possibilitando, por exemplo, a criação de novas variáveis como a Word Mover's Distance (WMD) que utiliza do Word2vec para avaliar a distância entre dois documentos de maneira significativa, independentemente de terem ou não terem palavras em comum.

Técnicas mais sofisticadas de reamostragem também devem ser consideradas para melhor a solução, pois como citado na seção de implementação utilizamos neste projeto a técnica mais simples de sobre-amostragem (naive random over-sampling), uma ótima opção seria o SMOTE (Synthetic Minority Over-sampling Technique). Em termos simples, ele olha para o espaço de recurso para os pontos de dados da classe minoritária e considera seus k vizinhos mais próximos.

REFERÊNCIAS

Richert, W., & Coelho, L.P. Building Machine Learning Systems with Python. Local de publicação: Packt Publishing Ltd., 2013.

Matt J., Yu Sun, Nicholas I. Kolkin and Kilian Q. Weinberger. From Word Embeddings To Document Distances, *Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130*.

Susan Li. Finding Similar Quora Questions with BOW, TFIDF and Xgboost, *Towards Data Science*, 2018. Disponível em:

<https://towardsdatascience.com/finding-similar-quora-questions-with-bow-tfidf-and-random-forest-c54ad88d1370>

Jardeson L. N. Barbosa, João P. Albuquerque V., Roney L. S. Santos, Gilvan V. M. Jr., Mariana S. Muniz, Raimundo S. Moura. Introdução ao Processamento de Linguagem Natural usando Python. *III Escola Regional de Informática do Piauí*, 2017.