

Machine Learning Engineer Nanodegree

Capstone Project

Eliézer F. Bourchardt

January 7st, 2018

I. Definição

Visão Geral do Projeto

A reforma política brasileira é um assunto que constantemente está na mídia. As eleições no Brasil são as mais caras do mundo, onde o sistema eleitoral, adotado pelo país desde 1945, obriga o candidato a disputar votos em uma área muito grande. Além disso, gastos elevados podem resultar em sucesso nas eleições, pela fragilidade de boa parte do eleitorado, suscetível à influência do poder econômico e das máquinas administrativas¹.

A utilização do poder econômico atrelado a eleição de candidatos é considerado “voto de cabresto”, onde no passado era uma prática explícita, hoje é realizada pela compra de votos². Para alimentar essa máquina o político se utiliza de doações realizadas a ele, favorecendo assim a corrupção³.

Todos os datasets utilizados neste projeto são disponibilizados pelo TSE⁴. Utilizarei apenas os dados da eleição de 2014, filtrando apenas candidatos a deputado federal, estadual e distrital. Também utilizarei o dataset *Electoral Donations in Brazil* disponibilizado no repositório do Kaggle por Felipe Leite Antunes.

¹ <https://www12.senado.leg.br/noticias/materias/2014/09/15/eleiassaues-no-brasil-sapso-as-mais-caras-do-mundo>

² https://pt.wikipedia.org/wiki/Voto_de_cabresto

³ <http://politica.estadao.com.br/noticias/geral/as-10-empresas-que-mais-doaram-em-2014-ajudam-a-eleger-70-da-camara,1589802>

⁴ <http://www.tse.jus.br/eleitor-e-eleicoes/estatisticas/repositorio-de-dados-eleitorais-1/repositorio-de-dados-eleitorais>

Declaração do Problema

As eleições de 2014 tiveram 21.101 candidatos a deputado estadual, federal e distrital. Destes, 1.572 foram eleitos, o que dá uma porcentagem de 7,45% de eleitos. Também houve um total de R\$ 2,4 bilhões em doações a candidatos, provenientes de pessoas físicas e jurídicas. Estes candidatos também declararam um total de R\$ 9,1 bilhões em bens⁵. Isto demonstra o peso econômico das eleições.

Também existem poucas pesquisas de intenção de votos para deputados e as realizadas normalmente não funcionam muito bem⁶. Isso acontece porque há um grande número de candidatos e porque existe um cálculo para definir os eleitos, onde muitas vezes o mais votado não é eleito⁷.

O poder econômico relacionado ao poder político é um grande entrave para a reforma política brasileira. Mesmo com alterações nas regras para a próxima eleição, é provável que esta relação mantenha-se, sendo possível utilizar o mesmo modelo, talvez com algumas alterações.

Para este problema de classificação utilizarei como dados de entrada arquivos csv (disponibilizado pelo TSE) com os dados já estruturados. Como saída esperada serão duas classes possíveis: eleito (1) e não eleito (0).

Métricas

Este é um problema de classificação em um dataset desbalanceado, onde apenas 7,45% dos dados pertencem a classe de candidatos eleitos. É necessário considerar isto para que a métrica reflita efetivamente a qualidade do modelo. Abaixo listo as duas métricas que utilizarei para avaliar os modelos.

F1 Score

$$F1 = 2 * (Precision + Recall) / (Precision + Recall)$$

O F1 Score é a média harmônica entre a Precisão e o Recall, onde uma pontuação F1 atinge seu melhor valor em 1 e pior pontuação em 0. A contribuição relativa de precisão e recall para a pontuação F1 é igual⁸.

AUC – Area Under the ROC Curve

Receiver Operating Characteristic, ou simplesmente ROC Curve, é um gráfico que ilustra o desempenho de um sistema classificador binário, pois seu limiar de discriminação é

⁵ Análise preliminar realizada no dataset.

⁶

<https://www.jornalopcao.com.br/bastidores/pesquisas-de-intencao-de-votos-para-deputado-federal-e-estadual-nao-sao-uteis-para-densidade-eleitoral-149/>

⁷ <http://g1.globo.com/politica/eleicoes/2014/noticia/2014/09/entenda-como-e-feito-o-calculo-para-eleicao-de-um-deputado.html>

⁸ http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

variado. É criado traçando a fração de positivos verdadeiros dos positivos (TPR = taxa positiva verdadeira) versus a fração de falsos positivos fora dos negativos (FPR = taxa de falso positivo), em várias configurações de limiar. TPR também é conhecido como sensibilidade, e FPR é um menos a especificidade ou taxa negativa verdadeira⁹.

A AUC mede a área sob uma curva formada pelo gráfico entre a taxa de exemplos positivos, que realmente são positivos, e a taxa de falsos positivos. O desempenho do modelo é medido em vários pontos de corte, não necessariamente atribuindo exemplos com probabilidade maior que 50% para a classe positiva, e menor, para a classe negativa

II. Analysis

Data Exploration

Todos os datasets utilizados neste projeto são disponibilizados pelo TSE¹⁰. Utilizarei apenas os dados da eleição de 2014, filtrando apenas candidatos a deputado federal, estadual e distrital. Também utilizarei o dataset *Electoral Donations in Brazil* disponibilizado no repositório do Kaggle por Felipe Leite Antunes.

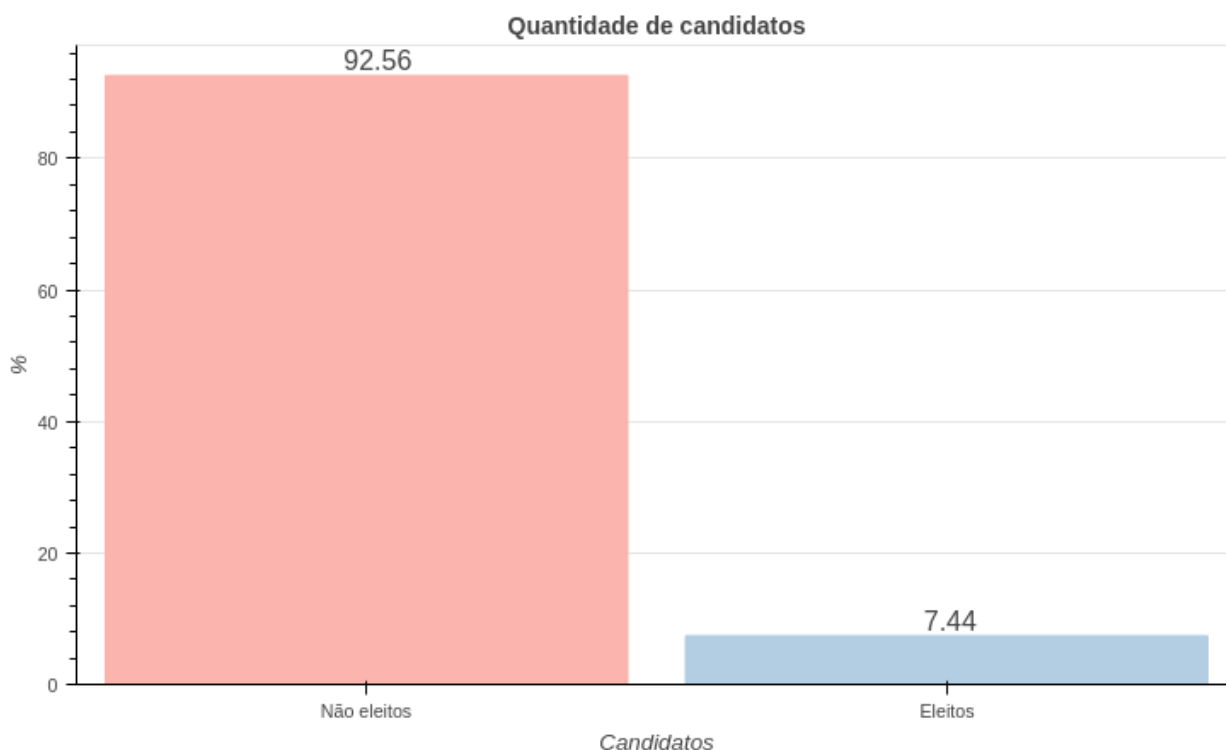
A seguir listo os datasets utilizados e suas referências:

- 1) consulta_cand_2014.zip¹¹ - Possui as informações pessoais de todos os candidatos, separado por UF. Este dataset possui os dados de 21.101 candidatos.
 - a) Dos atributos existentes, utilizarei nove, sendo eles:
 - i) Categóricos: CODIGO_CARGO, CODIGO_OCUPACAO, IDADE_DATA_ELEICAO, CODIGO_SEXO, COD_GRAU_INSTRUCAO, CODIGO_ESTADO_CIVIL, CODIGO_COR_RACA.
 - ii) Numérico: DESPESA_MAX_CAMPANHA.
 - iii) Variável alvo: DESC_SIT_TOT_TURNO - Com os seguintes valores:
 - “ELEITO POR QP”, “ELEITO POR MEDIA” = 1.
 - Total de eleitos 1.572.
 - “NÃO ELEITO”, “SUPLENTE” = 0.
 - Total de não eleitos: 19.529.

⁹ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

¹⁰ <http://www.tse.jus.br/eleitor-e-eleicoes/estatisticas/repositorio-de-dados-eleitorais-1/repositorio-de-dados-eleitorais>

¹¹ http://agencia.tse.jus.br/estatistica/sead/odsele/consulta_cand/consulta_cand_2014.zip



- 2) bem_candidato_2014.zip¹² - Possui as informações dos bens declarados pelos candidatos, separado por UF. Deste dataset irei utilizar apenas a soma dos valores declarados por cada candidato.
- 3) receitas_candidatos_2014_brasil.txt¹³ - Lista de doações aos candidatos. Utilizarei o valor das receitas doadas para cada candidato, bem como o doador e tipo de doador.
 - a) Para separar as receitas por setores econômicos utilizei a tabela de classes de CNAE, que é um arquivo tipo XLS disponibilizado pelo IBGE¹⁴. Foi necessário realizar uma limpeza nesta dataset para vincular com o dataset das listas de doações.
 - b) Foi possível mapear todos os setores econômicos, exceto um, onde foi necessário atribuir o setor manualmente.

Os datasets 1 e 2 são disponibilizados em arquivos compactados. Estes arquivos contêm os dados separados por UF. Dessa forma é necessário realizar o processo de consolidação destes arquivos em 2 datasets. Já o dataset 3 já está disponibilizado de forma consolidada.

O dataset final ficou com os 41 atributos:

['CODIGO_CARGO', 'CODIGO_OCUPACAO', 'IDADE_DATA_ELEICAO', 'CODIGO_SEXO', 'COD_GRAU_INSTRUCAO', 'CODIGO_ESTADO_CIVIL', 'CODIGO_COR_RACA', 'CODIGO_NACIONALIDADE', 'DESPESA_MAX_CAMPANHA', 'TARGET', 'VALOR_BEM', 'setor_A', 'setor_B', 'setor_C', 'setor_D', 'setor_E', 'setor_F', 'setor_G', 'setor_H', 'setor_I', 'setor_J', 'setor_K',

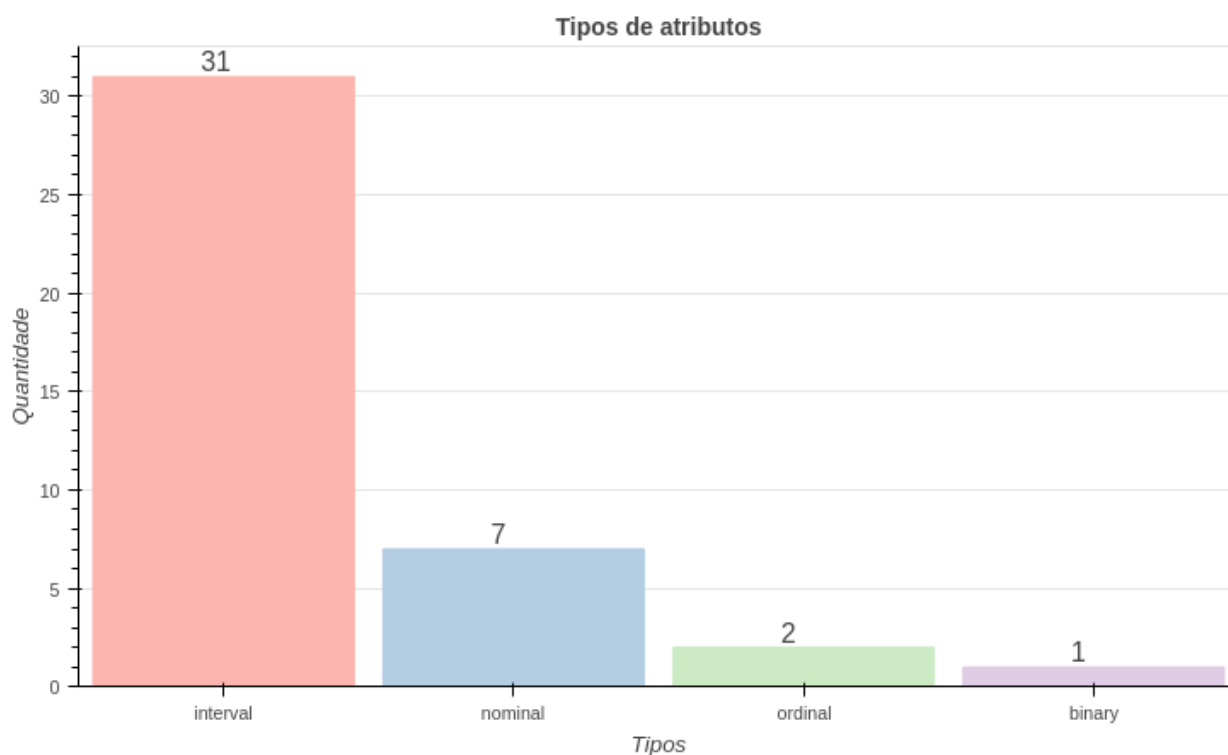
¹² http://agencia.tse.jus.br/estatistica/sead/odsele/bem_candidato/bem_candidato_2014.zip

¹³ https://www.kaggle.com/felipeleiteantunes/electoral-donations-brazil2014/downloads/receitas_candidatos_2014_brasil.txt

¹⁴ <https://concla.ibge.gov.br/classificacoes/download-concla.html>

'setor_L','setor_M', 'setor_N', 'setor_O', 'setor_P', 'setor_Q', 'setor_R', 'setor_S',
'setor_nao_identificado', 'tp_receita_aplicacao', 'tp_receita_evento', 'tp_receita_fisica',
'tp_receita_internet', 'tp_receita_juridica', 'tp_receita_nao_identificada', 'tp_receita_outro',
'tp_receita_partido', 'tp_receita_proprio', 'valor_receita']

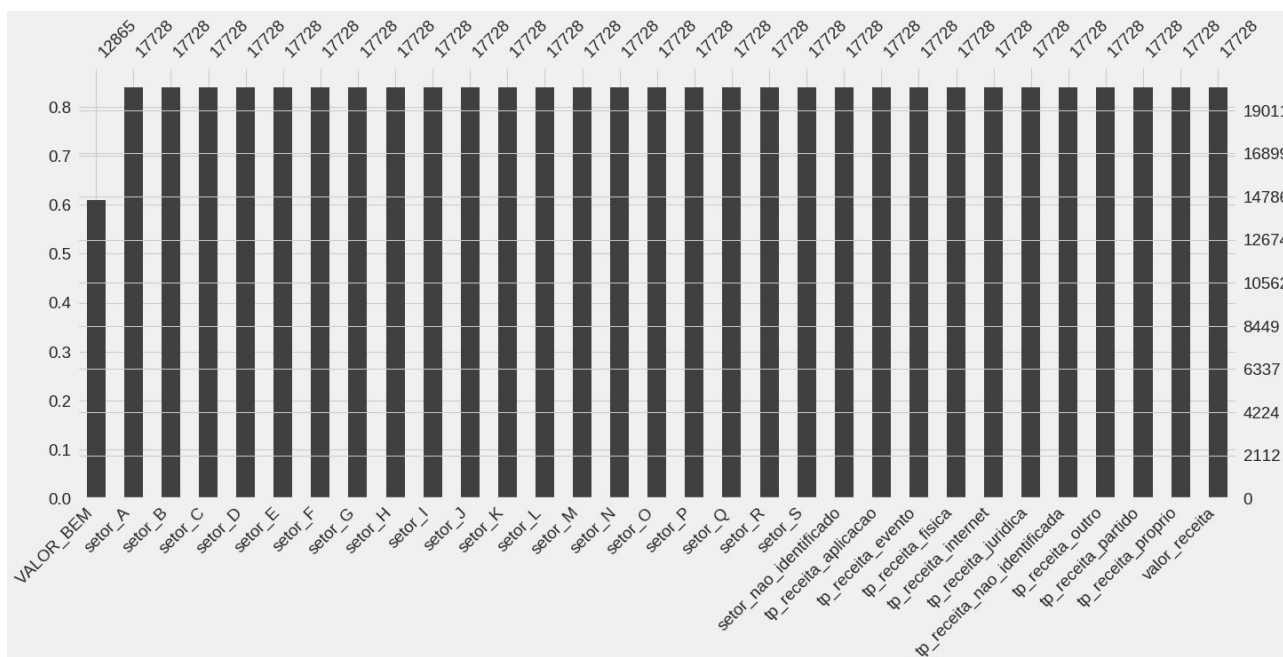
Distribuídos da seguinte forma:



Dados faltantes

Como esperado, este dataset não possui quantidade significativa de dados faltantes. O atributo que mais possui dados faltantes é o valor dos bens, significando que estes candidatos não possuem nada no nome (na teoria). Alguns poucos candidatos também não tiveram dinheiro investido de doações.

O gráfico abaixo foi gerado utilizando a biblioteca *missingno*. Nele é possível perceber que o atributo que mais havia dados faltantes era o valor declarado de bens. Um fato interessante foi que a visualização deste gráfico foi útil para detectar problemas no *merge* dos datasets, que eu não tinha percebido antes de realizar esta análise.



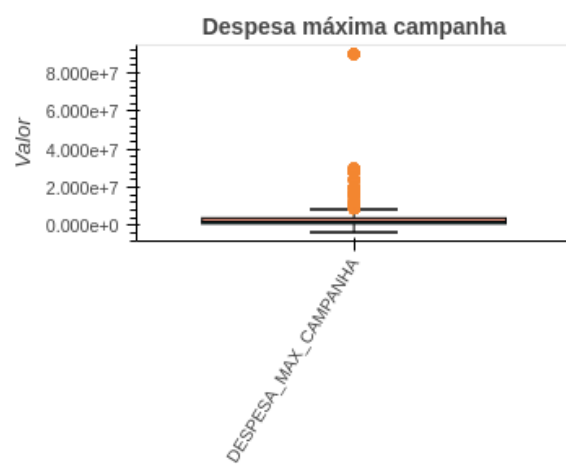
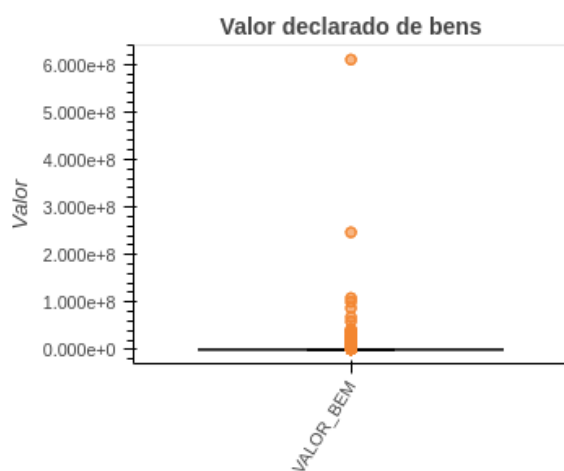
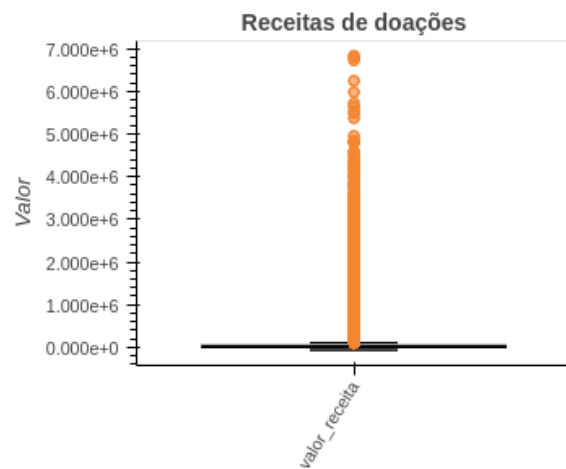
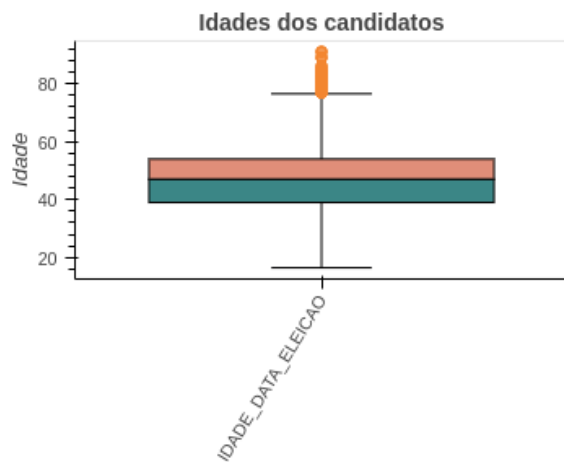
Outliers

Utilizei a função `dataframe.describe()` para ter uma visão geral de todos os atributos. Em geral o dataset possui valores dentro do esperado e alguns atributos com valores aparentemente altos.

	IDADE_DATA_ELEICAO	DESPESA_MAX_CAMPANHA	VALOR_BEM	valor_receita
count	21124.00	21124.00	21124.00	21124.00
mean	46.75	2759528.59	430926.83	114321.59
std	11.22	3933738.89	4912578.33	379515.13
min	20.00	0.00	0.00	0.00
25%	39.00	1000000.00	0.00	1020.22
50%	47.00	2000000.00	35376.29	6324.30
75%	54.00	4000000.00	283164.17	42983.44
max	91.00	90000000.00	610000000.00	6832480.98

- Candidato com a idade de 91 anos;
- Valor máximo declarado de bens R\$ 610.000.000,00;
- Valor máximo próprio utilizado na campanha R\$ 4.041.310,27;
- Valor máximo recebido do partido R\$ 5.769.682,00;
- Valor máximo recebido em doação R\$ 6.832.480,98.

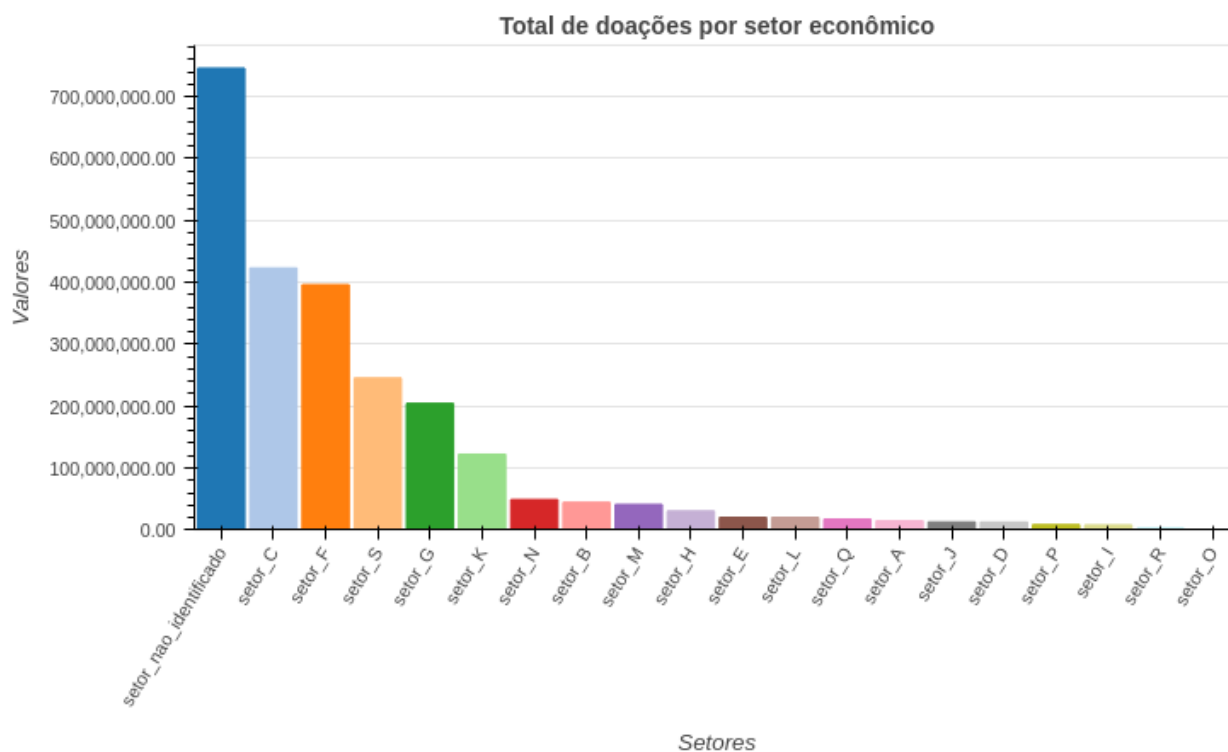
Para visualizar melhor estes atributos, utilizei o gráfico *boxplot*.



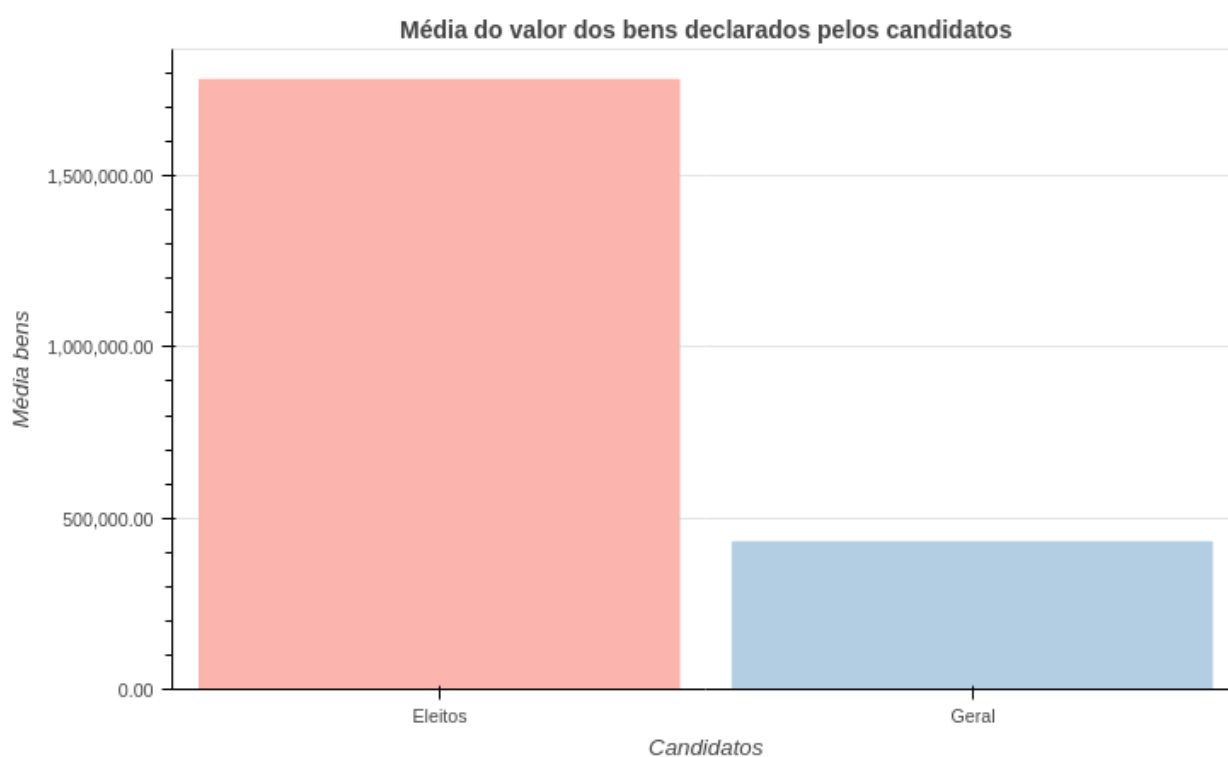
Visualização exploratória

Realizei diversos análises exploratórias para entender melhor os dados e verificar a sua relevância. A primeira análise foi verificar qual o total de doações por setor econômico, onde foi possível ver que se destaca setor não identificado, que são doações que não foram rastreadas.

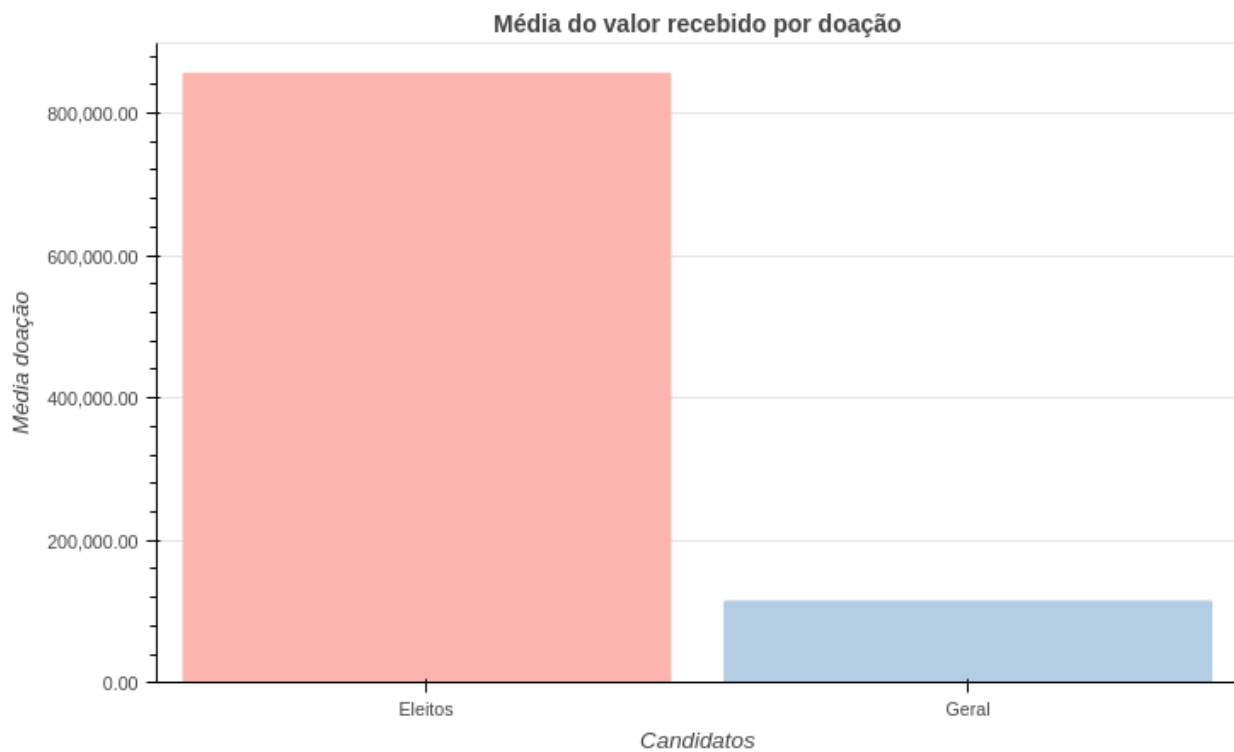
O gráfico desta análise pode ser visualizado abaixo:



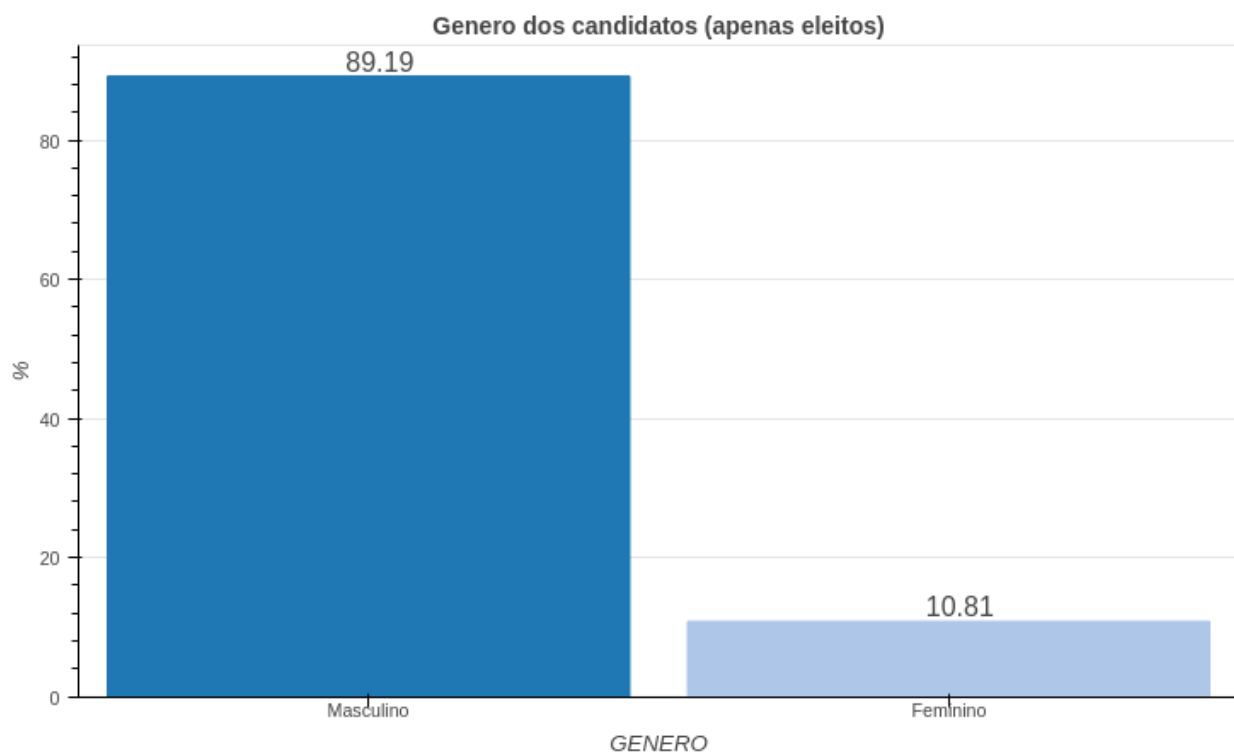
Também calculei a média de bens declarados entre todos os candidatos e os candidatos eleitos. Esta média mostrou-se superior para os candidatos eleitos, evidenciando o poder econômico na política.

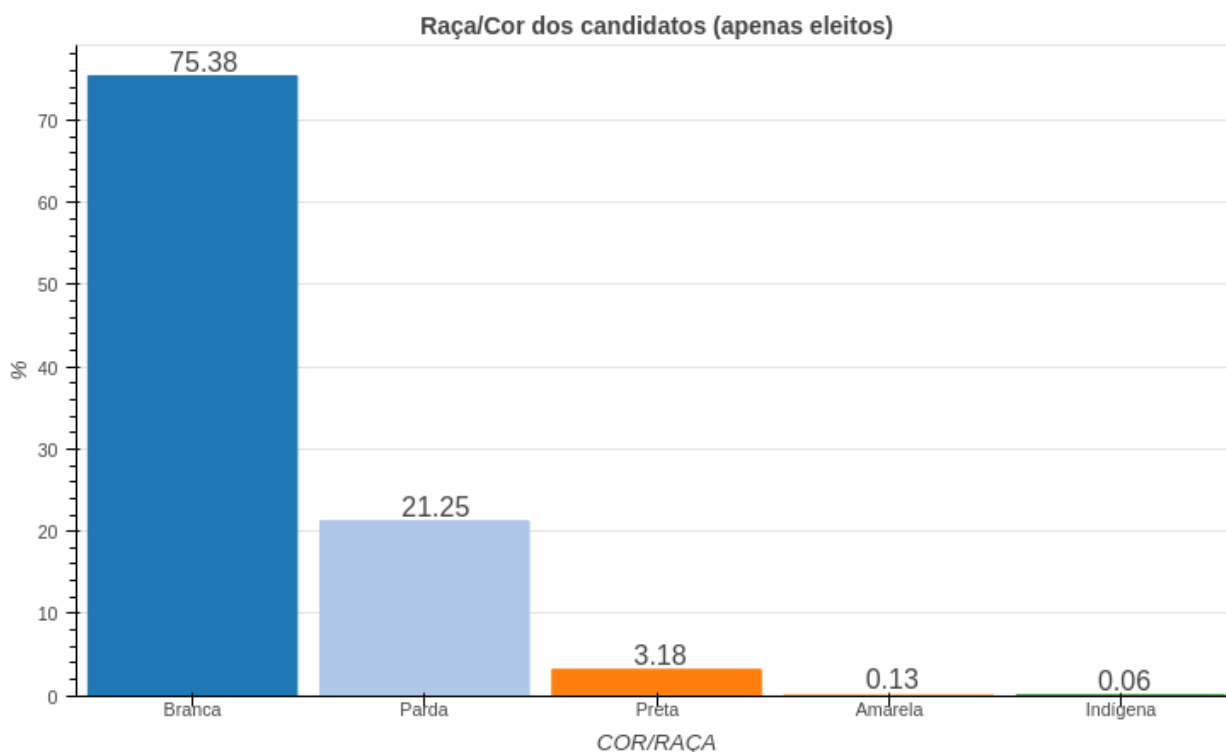


A mesma média foi calculada para o valor de doações. Mostrou a mesma relação, onde os candidatos eleitos possuem um valor de doação muito maior do que comparando com a média geral.

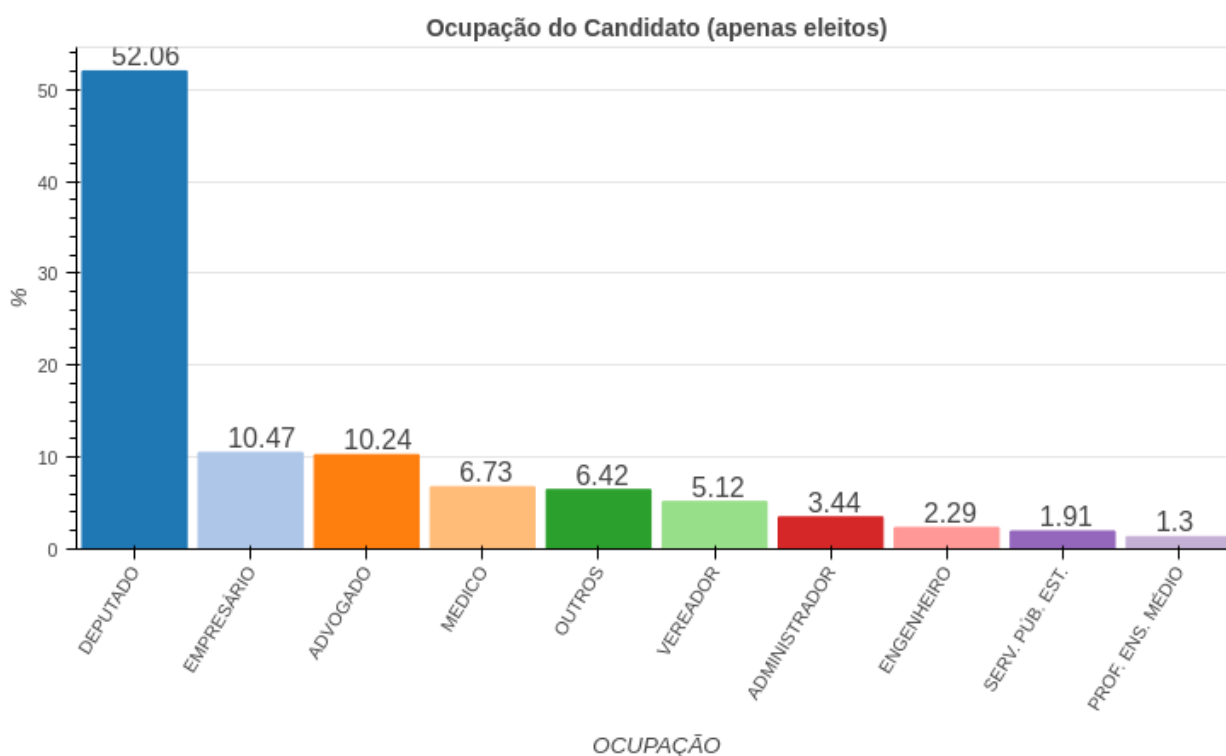


Também realizei algumas visualizações em características dos candidatos. Nestas visualizações foi possível perceber desigualdades entre gêneros e raça/cor.



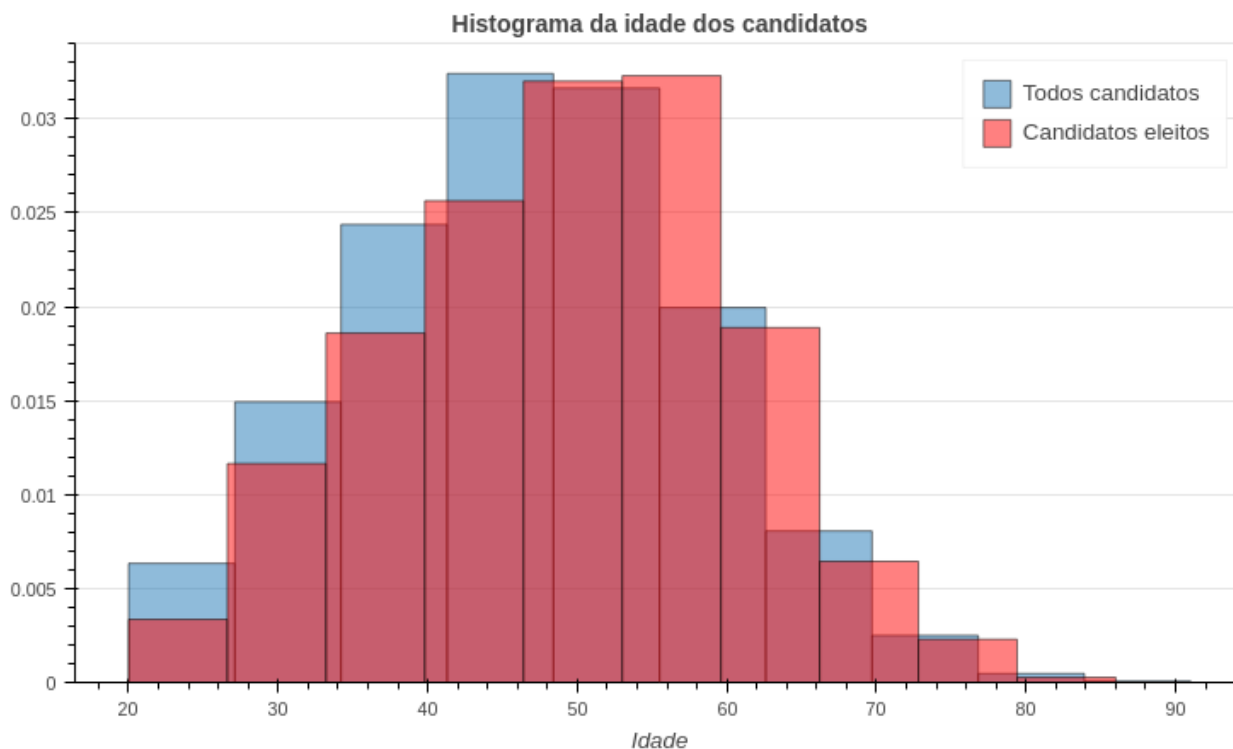


Outra análise interessante foi no tipo de ocupação dos candidatos eleitos. A maioria dos eleitos tem como profissão “deputado”. Estes dados confirmam a pouca renovação política existente.



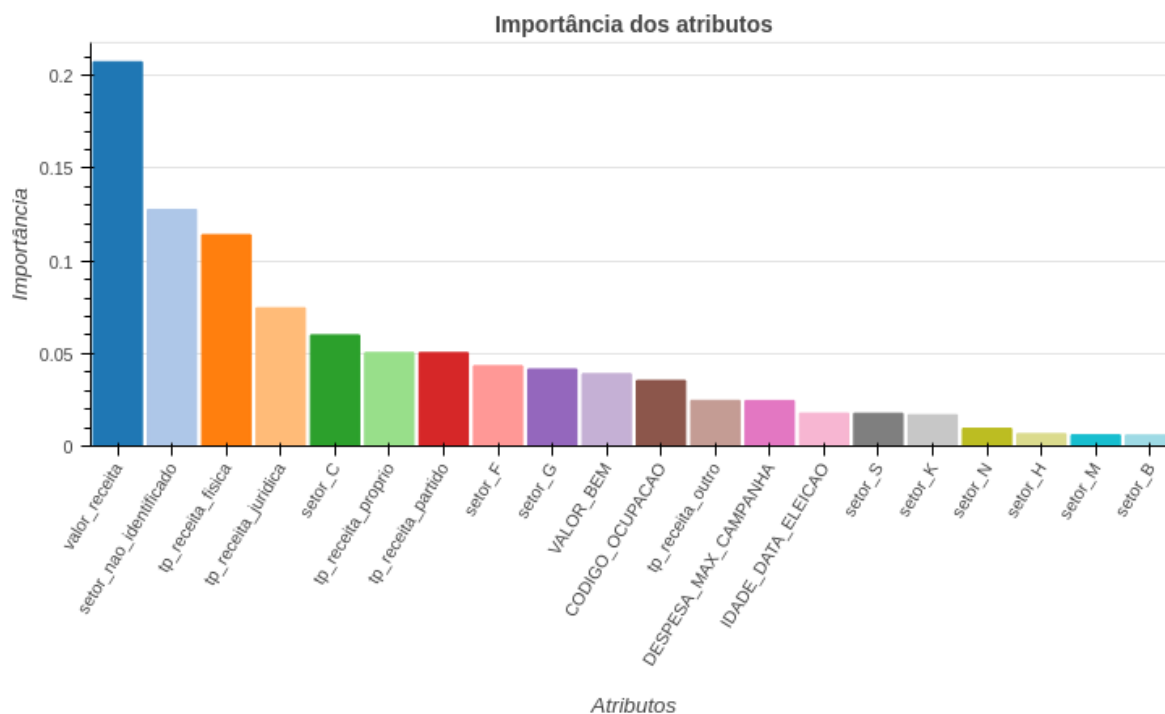
Também criei um histograma a idade dos candidatos. Em azul os candidatos eleitos sobrepondo, em vermelho, a idade dos candidatos eleitos. Para melhor visualizar os dados estão normalizados. A maior quantidade de candidatos está entre 40 e 55 anos. É

possível perceber que o histograma dos candidatos eleitos está ligeiramente deslocado para a direita, mostrando que os candidatos eleitos ficam entre 45 e 60 anos.



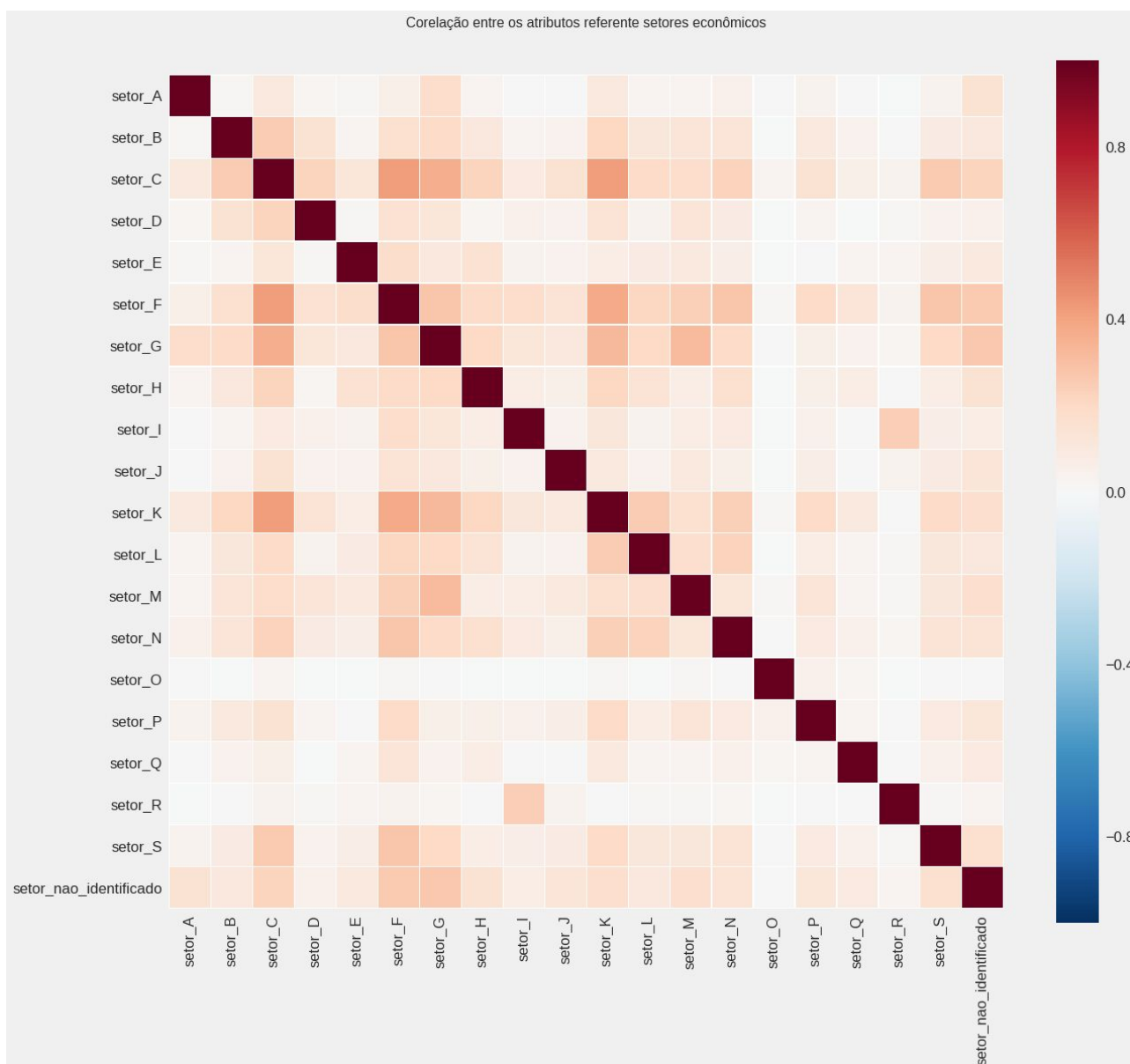
Importância dos atributos

Outra análise realizada foi a importância dos atributos para a definir se o candidato foi eleito, com o classificador Random Forest. Obtive o resultado abaixo:



Correlação dos atributos

O gráfico abaixo mostra a correlação dos atributos. Certos setores possuem alguma correlação, apesar dela não ser muito forte. Os setores C, G, F e K são os que mais parecem correlacionados. Estes setores são respectivamente: Indústria de Transformação, Comércio, Construção, Atividades financeiras.



Algoritmos e Técnicas

1. Consolidação dos datasets.

- Os dados disponíveis no site do TSE (consulta_cand_2014.zip e bem_candidato_2014.zip) estão no formato CSV, separados em arquivos distintos para cada UF. Farei a consolidação dos arquivos criando dois arquivos com os datasets.

- b. O dataset disponível no Kaggle (receitas_candidatos_2014_brasil.txt) já está consolidado em um único arquivo.
 - i. Para complementar este dataset utilizarei um dataset de CNAES, para separar as doações por setores econômicos.
- 2. Criação de um novo dataset.
 - a. Para realizar a análise e aplicar os algoritmos criarei um novo dataset com:
 - i. os dados pessoais dos candidatos;
 - ii. valor total de bens de cada candidato;
 - iii. valor total de doações para cada candidato, doações de pessoas físicas, jurídicas e empresa.
 - b. Eliminarei atributos não relevantes.
- 3. Análise exploratória.
 - a. Detecção de dados faltantes - nesta etapa eu irei verificar se há algum atributo com dados faltantes, utilizando o pacote missingno. Caso houver vou avaliar qual é a melhor maneira de completá-los.
 - b. Conforme a necessidade, para melhorar o desempenho do algoritmo, posso avaliar a necessidade de eliminar alguns atributos ou criar novos.
 - c. Analisarei todos atributos para identificar seus tipos.
 - d. Analisarei a correlação entre os atributos.
 - e. Utilizando random forest, irei verificar a importância dos atributos.
 - f. Também tenho interesse em responder algumas perguntas:
 - i. Qual a porcentagem de candidatos e candidatos eleitos?
 - ii. De todos os candidatos, como estão divididos de acordo com suas características de sexo, idade, raça/cor, profissão? E estas mesmas características quanto aos candidatos eleitos?
 - iii. A declaração de bens é uma forma de identificar o poder econômico do candidato. Qual o valor médio de bens dos candidatos e dos candidatos eleitos? E quanto ao valor doado?
- 4. Testes de algoritmos de classificação.
 - a. O atributo alvo nos dados originais possui 4 valores: “ELEITO POR QP”, “ELEITO POR MEDIA”, “NÃO ELEITO”, “SUPLENTE”. Neste projeto o que interessa é saber se o candidato foi ou não eleito. Dessa forma considerarei os dois primeiros valores como “ELEITO=1” e os dois últimos como “NÃO ELEITO=0”.
 - b. Ajuste na distribuição das classes utilizando imbalanced-learn.
 - c. Pretendo fazer testes utilizando os algoritmos:
 - i. SVM;
 - ii. KNeighborsClassifier;
 - iii. RandomForestClassifier;
 - iv. XGBoost;
 - d. Os testes serão feitos utilizando StratifiedKFold e cross_val_score, utilizando como métrica F1 Score.
- 5. Ajustes finos

- a. Com os dois melhores algoritmos acima, realizarei a otimização de hiperparâmetros para obter o melhor score.

Ferramentas e bibliotecas a utilizar: Python, Jupyter Notebook, pandas, scikit learn, seaborn, matplotlib, XGBoost e outras bibliotecas serão adicionadas se necessário.

Benchmark

Não encontrei nenhum trabalho que utilize estes conjunto de dados com o objetivo de realizar a previsão de eleição de candidatos a deputados. Para efeitos de comparação utilizarei duas pesquisas realizadas em 2014, que indicam a intenção de votos dos eleitores. Porém deve-se levar em consideração que esta pesquisa é apenas de intenção de votos, onde não leva em conta o cálculo para a eleição do deputado. Dessa forma, eu apenas analisei os primeiros colocados, se condizem com os primeiros colocados na pesquisa de intenção, para ter uma métrica para comparar.

A primeira pesquisa analisada é a de intenção de votos, do Instituto Fonte Real, para candidatos a deputados federais de Rondônia, realizada em julho de 2014. Dos 8 candidatos eleitos, 5 estavam nos primeiros lugares na pesquisa.

A segunda pesquisa é a de intenção de votos a deputado estadual de Minas Gerais, realizada em setembro de 2014 pela DataTempo. Dos 77 candidatos eleitos, 35 estavam em primeiro lugar na pesquisa.

Calculando estes dois datasets obtive as seguintes métricas:

F1 Score: 0.49 / ROC AUC: 0.73

E também utilizando Gaussian Naive Bayes como preditor base em testes iniciais, obtive as métricas:

F1 Score: 0.55 / ROC AUC: 0.73.

III. Metodologia

Pré-processamento dos dados

Merge files

Os dados disponibilizados pelo TSE (consulta_cand_2014.zip e bem_candidato_2014.zip) possuem os dados separados em arquivos distintos por UF. Para consolidar estes arquivos criei o script “merge_files_brazil.py”. Este script consolida todos os arquivos em dois arquivos CSV: “consulta_cand_2014_Brazil.csv” e “bem_candidato_2014_Brazil.csv”. Além disso ele adiciona o cabeçalho dos atributos.

Preparação do dataset de candidatos

O dataset de candidatos possui 46 atributos com os dados pessoais de todos os candidatos a seguinte cargos: PRESIDENTE, VICE-PRESIDENTE, GOVERNADOR, VICE-GOVERNADOR, SENADOR, DEPUTADO FEDERAL, DEPUTADO ESTADUAL, DEPUTADO DISTRITAL, 1º SUPLENTE, 2º SUPLENTE. Alguns destes atributos são irrelevantes para a análise, sendo assim os removi, resultando em 25 atributos.

Depois disso foi mantido apenas os dados dos candidatos a deputados (CODIGO_CARGO = [6, 7, 8]), com candidatura deferida ou deferida com recurso. Totalizou 21.124 candidatos, sendo 1.572 (7,44%) eleitos.

Preparação do dataset de candidatos

O dataset de bens dos candidatos possui um total de 12 atributos. Foi apenas considerado o atributo VALOR_BEM, agrupando por candidatos. Em seguida realizei o merge com o dataset dos candidatos.

Preparação do dataset de doações

Doações indiretas são quando o candidato recebe a doação de terceiros e os atributos chamados “(...) doador originário” possuem os dados do doador que fez a doação. Foi substituído aos dados do doador a informação do doador originário.

Realizei a limpeza e ajuste no dataset “Subclasses CNAE 2.2 - Estrutura.xls” disponibilizado pelo IBGE.

Com o dataset CNAE preparado, localizei os setores econômicos de cada doação. Fiz isso realizando um merge com base no atributo “Setor econômico do doador”. Apenas um setor econômico não foi localizado, onde eu atribui o setor manualmente de acordo com a descrição. As doações em que não havia informação do setor econômico eu atribui o valor “setor_nao_identificado”.

Cada setor econômico identificado de acordo com a classificação do setor econômico, realizei a soma agrupando pelo setor, para depois utilizar a função *pivot* tendo como índice o número sequencial do candidato, permitindo assim realizar o merge com o dataset de candidatos.

Também realizei a soma de acordo com o tipo da origem da receita: pessoa física, pessoa jurídica, outros, partido, internet, não identificada, recurso próprio, aplicação e eventos. Depois de realizar o *pivot*, fiz o merge com o dataset de candidatos.

Por fim realizei a soma de todas as doações, agrupando por sequencial de candidato e gerei um arquivo “dados_tratados.csv” com o dataset final para ser utilizado.

Implementation

Dados faltantes e outliers

Os datasets utilizados não possuem dados faltantes. Apenas há dados onde o candidato não declarou bens ou não houve doação, estes dados estão como nulos. O mais coerente neste caso é substituí-los por zeros.

Os outliers detectados não foram removidos. Eles não são informações erradas e possuem dados significativos para este projeto. Por exemplo, no valor da receita dentro dos outliers há 1.475 candidatos eleitos, o que equivale a 93,82% dos candidatos eleitos. Valor declarado de bens, equivale a 51,78% dos candidatos eleitos.

Separação do dataset em conjunto de treino e teste

A separação dos dados em treino e teste deve ser feito para que se tenha um conjunto de dados para validar o resultado do processo de aprendizado de máquina. Os dados de testes devem ser separados para que em nenhuma hipótese ser utilizado no processo de treinamento. Caso isso acontecer o modelo terá um ótimo score, porém em outros dados o score será significativamente baixo, pois o modelo fica super ajustado.

Antes de separar os conjuntos, foi separado o mesmo conjunto de dados da pesquisa que foi utilizado para cálculo do benchmark. Estes dados não serão utilizados no treinamento para validação do modelo final.

Para realizar esta separação do restante dos dados dois datasets de treino e teste foi utilizado *train_test_split*, deixando 25% dos dados reservados para testes.

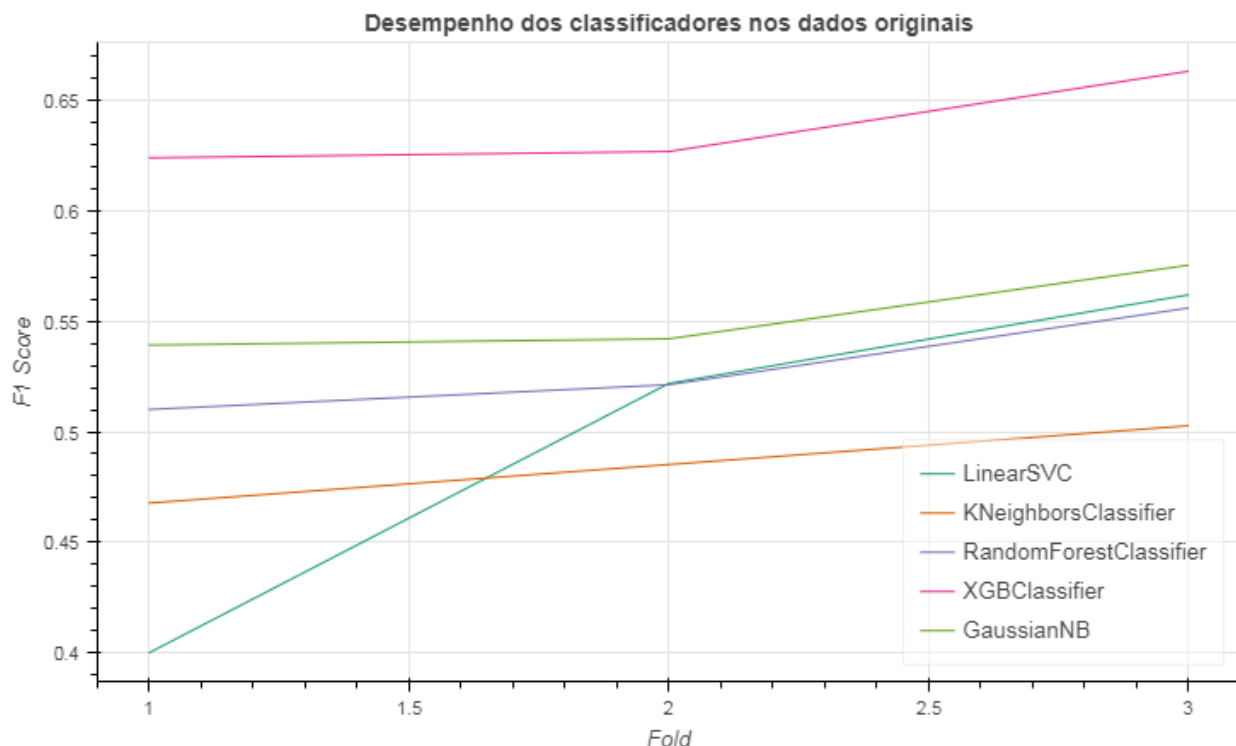
Testes com alguns algoritmos

Realizei alguns testes iniciais com LinearSVC, KNeighborsClassifier, RandomForestClassifier e XGBClassifier. Estes testes iniciais servem para termos uma base de qual algoritmo possui a melhor performance. O classificador GaussianNB, foi utilizado como benchmark e foi incluído nestes testes como referência.

Para realizar estes testes utilizei a função “*cross_val_model*”, com a utilização de *StratifiedKFold*, divide os dados de treino novamente em treino e teste, realizando várias divisões (*n_folds*). Esta função utiliza *Score F1* para validar o modelo em cada conjunto de dados e resulta a média simples para avaliação geral do algoritmo.

Conforme o gráfico abaixo, LinearSVC e KNeighborsClassifier obtiveram o pior F1 Score, já RandomForestClassifier e XGBClassifier obtiveram melhores resultados. Estes dois classificadores tiveram resultados bem similares.

Em todos os testes que eu realizei, LinearSVC foi o que teve o resultado mais intermitente, em algumas vezes ficava melhor que KNeighborsClassifier ou RandomForestClassifier, outras vezes seu resultado ficava com F1 Score abaixo de 0,35.



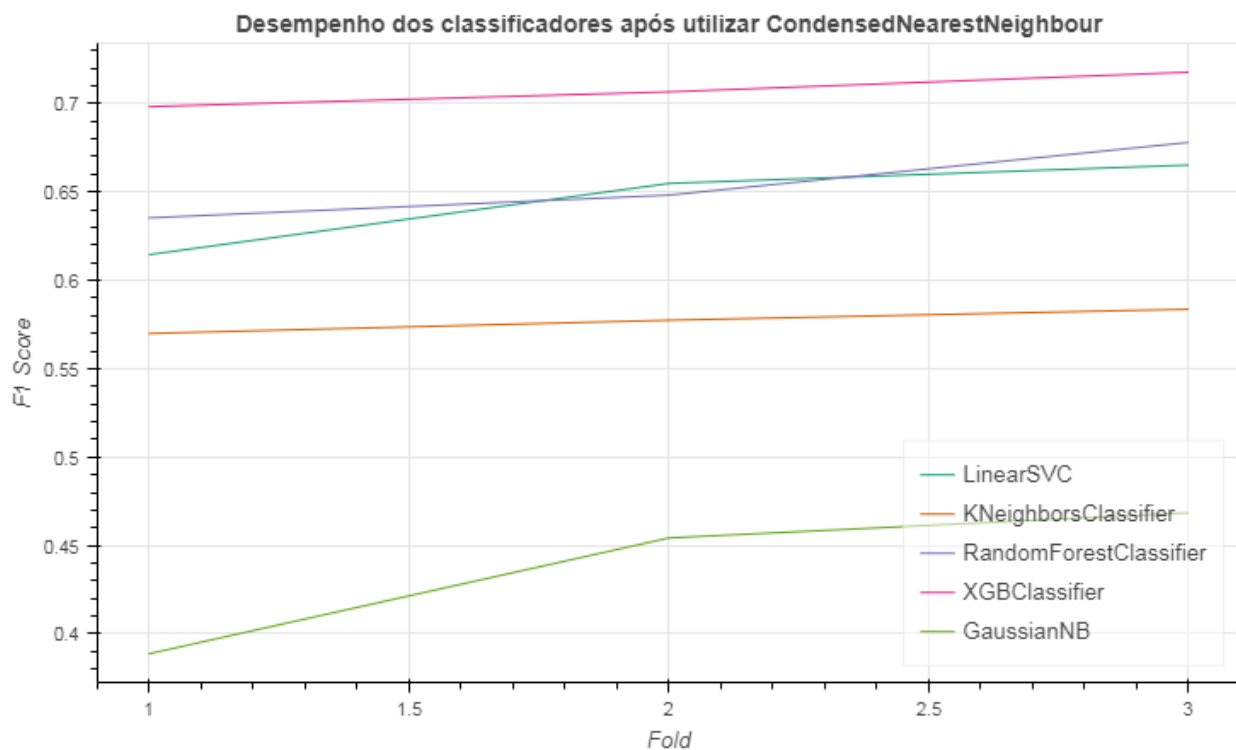
Dados desbalanceados

Dados desbalanceados são quando um tipo de classe se sobrepõe a outra em quantidade de elementos. Neste dataset a classe de candidatos eleitos correspondem a 7,44% dos dados.

Há duas formas de contornar este problema, aumentando ou reduzindo a quantidade de elementos de uma classe ou outra (*under_sampling* ou *over_sampling*).

A biblioteca *imblearn* possui alguns métodos como *RandomOverSampler*, *RandomUnderSampler*, *NearMiss* e *CondensedNearestNeighbour*. Realizei testes com todos estes métodos e o desempenho dos classificadores nos três primeiros métodos foram inferiores comparados com os dados originais (causando *overfitting* ou *underfitting*). O que apresentou melhor resultado foi *CondensedNearestNeighbour*, em contrapartida é o que mais tempo leva para realizar o treinamento. Este método utiliza o *nearest neighbour* para reduzir a quantidade de dados no dataset.

Este método reduziu de 15.007 amostras para 2.195, o que corresponde a 14,62% dos dados originais. No gráfico abaixo é possível ver que o desempenho dos classificadores melhoraram (exceto GaussianNB):



Refinamento

A parte de ajuste fino dos parâmetros é importante para obter o melhor resultado que o modelo pode oferecer. Neste trabalho, utilizo duas abordagens: `RandomizeSearchCV` e `GridSearchCV`¹⁵.

GridSearchCV

Neste processo é definido um conjunto de parâmetros que se deseja testar o modelo. Em seguida é usada validação cruzada para avaliar a melhor combinação possível. Entretanto esta estratégia possui um elevado custo computacional. Por exemplo, a busca de 10 valores de parâmetros diferentes para cada um dos quatro parâmetros exigirá 10.000 ensaios de validação cruzada, o que equivale a 100.000 ajustes do modelo e 100.000 conjuntos de previsões se a validação cruzada de 10 vezes estiver sendo usada.

RandomizeSearchCV

Em um processo de pesquisa aleatório, é pesquisado apenas um subconjunto aleatório dos valores dos parâmetros fornecidos. Isso permite o número de diferentes combinações de parâmetros que são testados, o que pode alterar dependendo do tempo computacional disponível.

¹⁵ <http://blog.kaggle.com/2015/07/16/scikit-learn-video-8-efficiently-searching-for-optimal-tuning-parameters/>

É certamente possível que o RandomizedSearchCV não encontre um resultado tão bom quanto o GridSearchCV, mas frequentemente ele encontra o melhor resultado (ou algo muito próximo) em uma fração do tempo que GridSearchCV teria tomado, muitas vezes superando GridSearchCV.

RandomizeSearchCV no modelo RandomForest

Optei por utilizar RandomizeSearchVC no modelo RandomForest, utilizando 100 interações. Obtive os seguintes parâmetros:

```
{'bootstrap': False, 'class_weight': None, 'criterion': entropy, 'max_depth': 30, 'max_features': 8, 'min_samples_leaf': 7, 'min_samples_split': 9, 'n_estimators': 59}
```

GridSearchCV no modelo XGBoost

No modelo XGBoost utilizei a abordagem descrita em “Complete Guide to Parameter Tuning in XGBoost”¹⁶, onde o autor utiliza GridSearchCV para descobrir os melhores parâmetros, utilizando algumas passos:

- 1) Ajuste do *learning rate* e número de *estimators*;
- 2) Ajuste do *max_depth* e *min_child_weight*;
- 3) Ajuste do *gamma*.
- 4) Ajuste do *subsample* e *colsample_bytree*.
- 5) Ajuste do *reg_alpha*.
- 6) Redução do *Learning Rate*.

Realizando estes passos obtive os seguintes parâmetros:

```
{'learning_rate':0.01, 'n_estimators':1007, 'max_depth':3, 'min_child_weight':8, 'gamma':0, 'subsample':0.9, 'colsample_bytree':0.6, 'reg_alpha':10}.
```

IV. Resultados

Avaliação e Validação de Modelos

Com o dataset de treino foi obtido o seguinte F1 Score:

RandomForest: 0.73. XGBoost: 0.72.

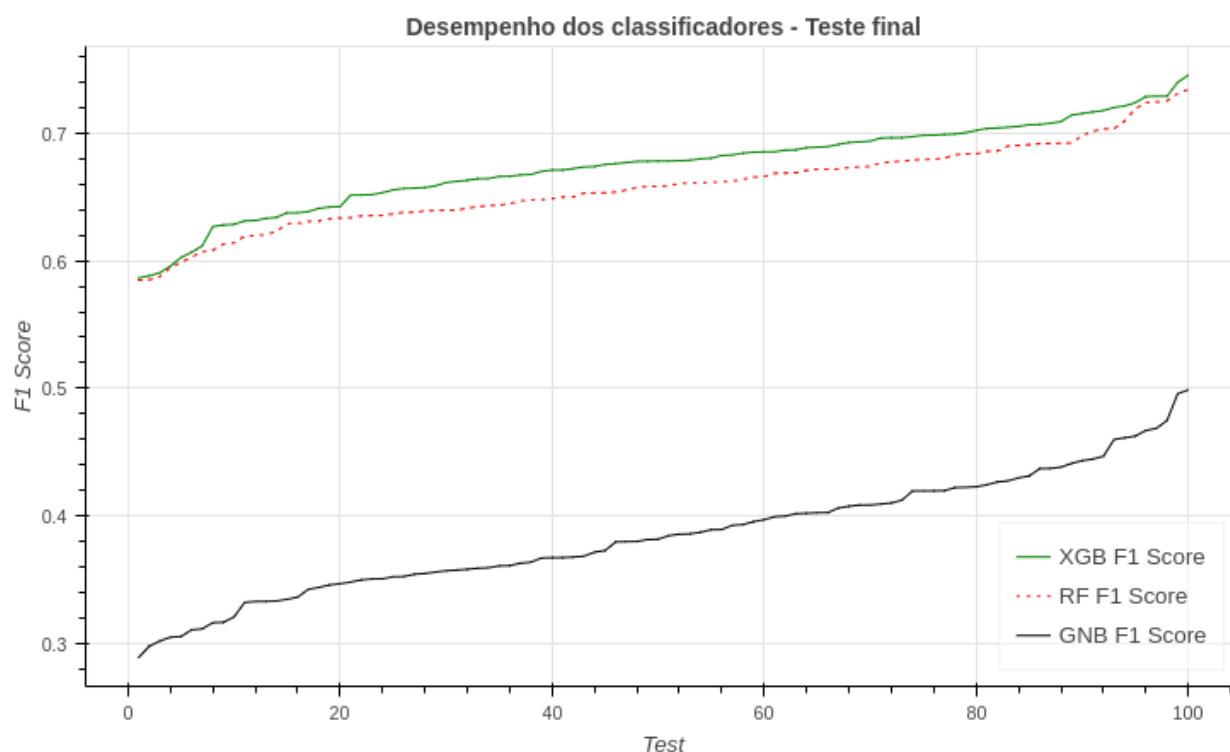
Como esperado, o desempenho nos dados de testes foi um pouco inferior. Na tabela abaixo é possível visualizar e comparar os resultados do Score F1 para cada modelo, utilizando o dataset de testes. Também é possível comparar o resultado utilizando o ajuste fino e sem utilizar.

¹⁶ <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>

	RandomForest		XGBoost	
	Sem ajustes	Com ajustes	Sem ajustes	Com ajustes
F1 Score	0.58	0.67	0.67	0.68
ROC AUC	0.80	0.86	0.85	0.86

No dataset de testes, a diferença do desempenho entre o modelo ajustado e não ajustado se mostrou mais evidente com RandomForest. Já com XGBoost houve pouca melhoria. Os dois modelos mostraram com um desempenho muito parecido, com XGBoost se mostrando um pouco melhor. Apesar da pouca margem de diferença, o melhor modelo é XGBoost. Para comparação, o modelo utilizado no benchmark (Gaussian Naive Bayes), nos dados de testes obteve o F1 score de 0.40 e ROC AUC de 0.63.

Para testar o modelo em seu limite, utilizei *train_test_split* para obter 100 conjuntos aleatórios de 10% dos dados de testes. No gráfico abaixo é possível observar o desempenho de cada classificador.



A média do F1 Score para Random Forest ficou 0.66 e XGBoost 0.67. Já Gaussian Naive Bayes 0.39.

Justificativa

Foram definidos dois benchmarks para este projeto.

No primeiro benchmark foi calculado com os dados de uma pesquisa de intenção de votos. Para comparar com este benchmark utilizei os dados separados previamente, que não fizeram parte do treino dos modelos e também não fizeram parte dos testes.

No segundo benchmark foi utilizado Gaussian Naive Bayes como um preditor básico. Na tabela abaixo é possível comparar cada resultado.

	Pesquisa	Naive Bayes	XGBoost
F1 Score	0.49	0.55	0.69
ROC AUC	0.73	0.73	0.94

Pode-se observar que o modelo XGBoost teve um resultado superior comparando tanto F1 Score como ROC AUC. Isso confirma também que este modelo pode ser utilizado como uma solução final.

V. Conclusão

Free-Form Visualization

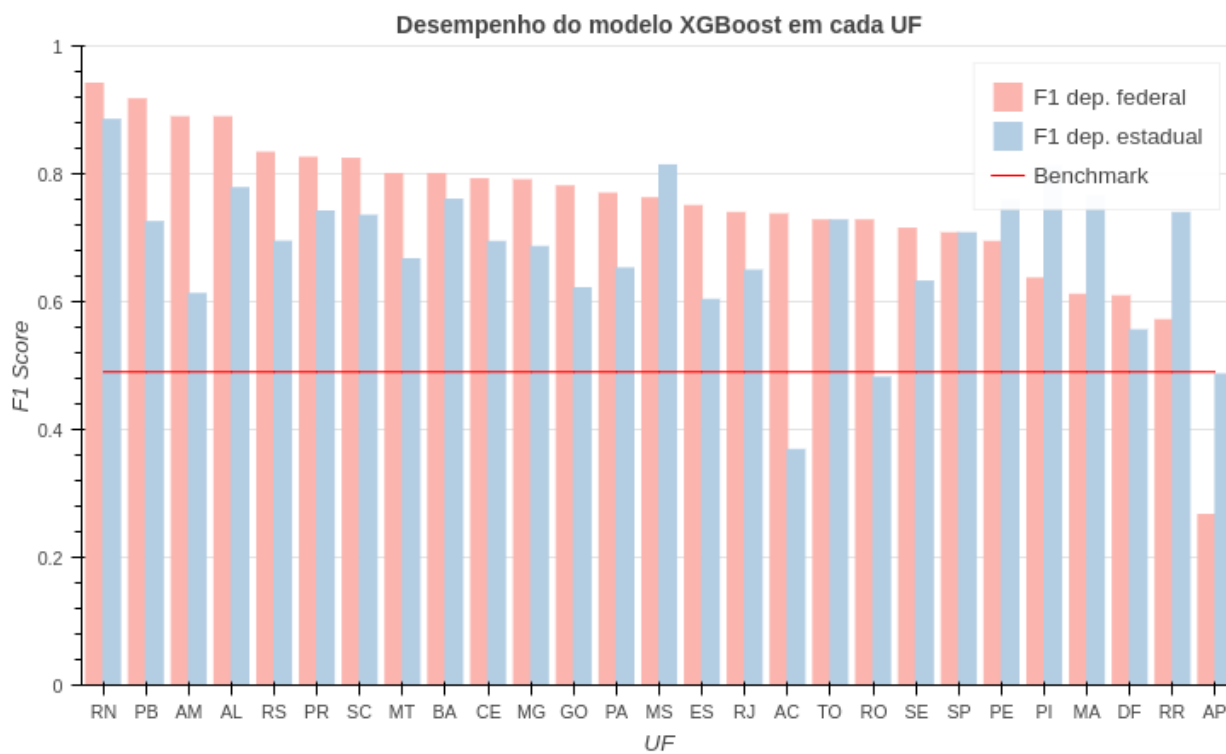
No gráfico abaixo é possível visualizar o F1 Score do modelo, aplicado em diferentes estados. Para deputado estadual a média dos scores ficou em 0,62 e para deputado federal em 0,69.

Comparando o desempenho entre os dois cargos, é possível perceber que deputado federal, em geral, teve um desempenho melhor que para deputado estadual, com exceção de AC, MS e MA. Em outros casos, o desempenho ficou bem parecido.

A linha vermelha mostra o desempenho do Score F1 do benchmark da pesquisa (0,49). Na maioria das UF o classificador teve um desempenho melhor, exceto alguns casos, por exemplo AP, DF, SE e AC.

É interessante lembrar que são raras as pesquisas de intenção de votos para candidatos a deputados. Um dos motivos é que a intenção de votos não está correlacionada diretamente com os candidatos vencedores. Outro motivo é que existem muitos candidatos e uma pesquisa teria que abranger uma área muito grande, elevando os custos da pesquisa.

Com base na pesquisa realizada e no cálculo do seu Score F1, este modelo se mostra superior, onde é possível ter uma noção se determinado candidato irá ou não vencer de acordo com suas características (claro que há outros fatores envolvidos que não foram abordados neste trabalho).



Reflexão

Parte 1 - Preparação dos dados e testes iniciais.

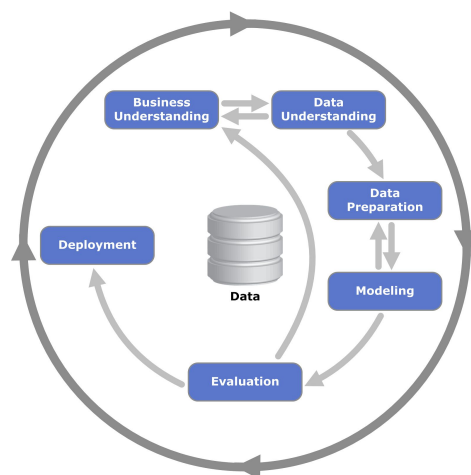
Ao escolher este problema para resolver fiz questão de escolher algo que utilizasse dados reais para sua resolução, onde necessitasse utilizar várias fontes de dados e que fosse algo desafiador.

O passo inicial foi testar se este problema poderia ser resolvido com Machine Learning. Procurei então por datasets que pudesse utilizar. O primeiro dataset disponível no Kaggle com os dados das doações aos candidatos já estava consolidado, facilitando o processo. Porém os datasets disponíveis no TSE não estavam, sendo o primeiro desafio escrever um script para consolidá-los.

Então quando tive os primeiros dados que poderiam ser utilizados, fiz um protótipo inicial e testei o classificador Gaussian Naive Bayes. O resultado deste classificador utilizei como benchmark. Também procurei pesquisas de intenção de votos para calcular o seu Score F1. Encontrei apenas duas pesquisas e foi onde descobri que não existem pesquisas para candidatos a deputados, por serem ineficientes e caras.

O segundo desafio na preparação dos dados foi a divisão das doações em setores econômicos, onde utilizei a tabela XLS de CNAE disponível pelo IBGE. Sua estrutura obrigou realizar um trabalho adicional para permitir sua utilização.

Com a etapa de processamento dos dados concluída, iniciei o processo de exploração dos dados. Foi aí que percebi que o processo não era linear. Em vários momentos, voltei na etapa de preparação dos dados para realizar ajustes necessários no dataset. Esta forma não linear é o processo CRISP de mineração de dados, onde devemos gastar o máximo de tempo possível na compreensão dos negócios/mini-ciclo de compreensão dos dados, até que tenhamos uma definição concreta e específica do problema que estamos tentando resolver¹⁷.



CRISP-DM Process diagram¹⁸

Parte 2 - Análise exploratória

A fase de exploração dos dados permite ter uma compreensão dos dados. Além disso, nesta fase foi onde descobri problemas que não tinha detectado anteriormente, na fase de preparação dos dados. Caso tivesse pulado diretamente para os modelos, estes problemas não seriam descobertos, possivelmente diminuindo o desempenho do modelo final.

Em um primeiro momento utilizei a biblioteca Matplotlib para gerar os gráficos para visualização. É muito fácil e rápido gerar gráficos utilizando pandas¹⁹, isso agiliza o processo de análise. Já a biblioteca Bokeh²⁰ possui vários recursos e apresenta um resultado interessante, assim alterei todos gráficos para esta biblioteca.

¹⁷ Data Science para Negócios - Foster Provost & Tom Fawcett, página 183.

¹⁸ Kenneth Jensen (Own work) [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>)], via Wikimedia Commons: <https://itsalocke.com/blog/crisp-dm-and-why-you-should-know-about-it/>

¹⁹ <https://datasciencelab.wordpress.com/2013/12/21/beautiful-plots-with-pandas-and-matplotlib/>

²⁰ <https://bokeh.pydata.org/en/latest/>

Parte 3 - Modelos

Aqui comecei a testar os modelos LinearSVC, KNeighborsClassifier, RandomForestClassifier e XGBClassifier para ver qual possuía o melhor desempenho. Para resolver o problema de classe desbalanceadas testei o pacote imbalanced-learn. Testei os métodos RandomOverSampler, RandomUnderSampler, NearMiss e por último CondensedNearestNeighbour, que melhorou o desempenho do classificador.

Com os dois modelos que tiveram melhores resultados iniciei o processo de ajuste de melhores parâmetros. Esta foi uma das partes difíceis e sensíveis, pois em certos momentos o modelo não apresentava bons resultados. Em um primeiro momento tentei otimizar os melhores parâmetros implementando busca aleatória com *hyperopt*²¹²². Não satisfeito com o resultado, para o modelo RandomForest utilizei *RandomizedSearchCV*²³ e para XGBoost utilizei *GridSearchCV*, seguindo uma série de passos²⁴ para obter a melhor configuração. Com os parâmetros ajustados, nos testes realizados foi possível perceber que houve melhorias no classificador.

Parte 4 - Testes

É nesse momento que devemos fazer o papel de questionador. Não basta o modelo apresentar um bom resultado durante o treinamento. É necessário levá-lo ao limite, testando em dados não vistos anteriormente.

A primeira coisa a se fazer é utilizar os dados de testes para avaliar o modelo. Os dois modelos escolhidos tiveram um desempenho parecido ao final. Para escolher o melhor modelo optei por realizar 100 experimentos, dividindo o dataset de testes em pequenos pedaços aleatórios contendo 10% dos dados. Assim foi possível perceber graficamente que XGBoost levou uma pequena vantagem em comparação com RandomForest.

Também utilizei os dados da pesquisa para efetuar os testes. Estes dados da pesquisa também não estavam presentes durante o treinamento. Nesse caso, o desempenho do classificador também se mostrou superior.

Por fim, motivado pela curiosidade, separei todos os dados por UF e apliquei o modelo separadamente por tipo de cargo (deputado federal e estadual). Neste teste foi possível perceber que o modelo não é eficiente em sua totalidade, porém na sua maioria o desempenho se mostrou melhor que o Benchmark.

Neste trabalho em nenhum momento utilizei dados de intenção de votos, nem informações de partidos políticos. Foi possível realizar a classificação dos candidatos apenas com as suas características e informações financeiras, permitindo o modelo ser

²¹ <http://sbern.com/2017/05/03/optimizing-hyperparameters-with-hyperopt/>

²² https://github.com/felipeeeantunes/udacity_live/blob/master/porto_seguro.ipynb

²³ http://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html

²⁴ <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>

utilizado como referência ao invés da utilização de pesquisa tradicional. Fica em aberto a possibilidade de aplicar este modelo nos dados da eleição geral de 2018.

Melhorias

Este foi um trabalho que tive uma grande satisfação em concluir, pois consegui realizar várias etapas do trabalho de um cientista de dados para chegar até o modelo final. Entretanto não existe um modelo perfeito e eu acredito que este trabalho pode e deve ser melhorado. Abaixo listo algumas melhorias que podem ser feitas²⁵:

- Pesquisar novos datasets, incluindo de eleições passadas e a de 2018, após concluída.
- Criar modelos para candidatos a prefeitos e vereadores.
- Relacionar os candidatos a reeleição com os dados da candidatura anterior para testar se os gastos realizados durante o mandato refletem em sua reeleição.
- Verificar a relevância de cada candidato na internet (notícias, twitter, etc).
- Verificar a teia de doações a candidatos e como isso reflete na eleição.
- Utilizar aprendizagem não supervisionada para criar agrupamentos de tipos de empresas para criar novos atributos, não se restringindo a tabela IBGE.
- Ao invés de prever se o candidato será ou não eleito, calcular a probabilidade de ser ou não eleito.
- Criação de uma interface disponível na internet para simulação de resultado de eleição.

²⁵ Fique livre em utilizar este projeto como referência para novos trabalhos. Qualquer trabalho derivado obrigatoriamente deve ser público e disponível no GitHub. Fique a vontade para enviar um e-mail para eliezerfb@gmail.com a respeito de qualquer dúvida, sugestão ou para informar sobre novos resultados.