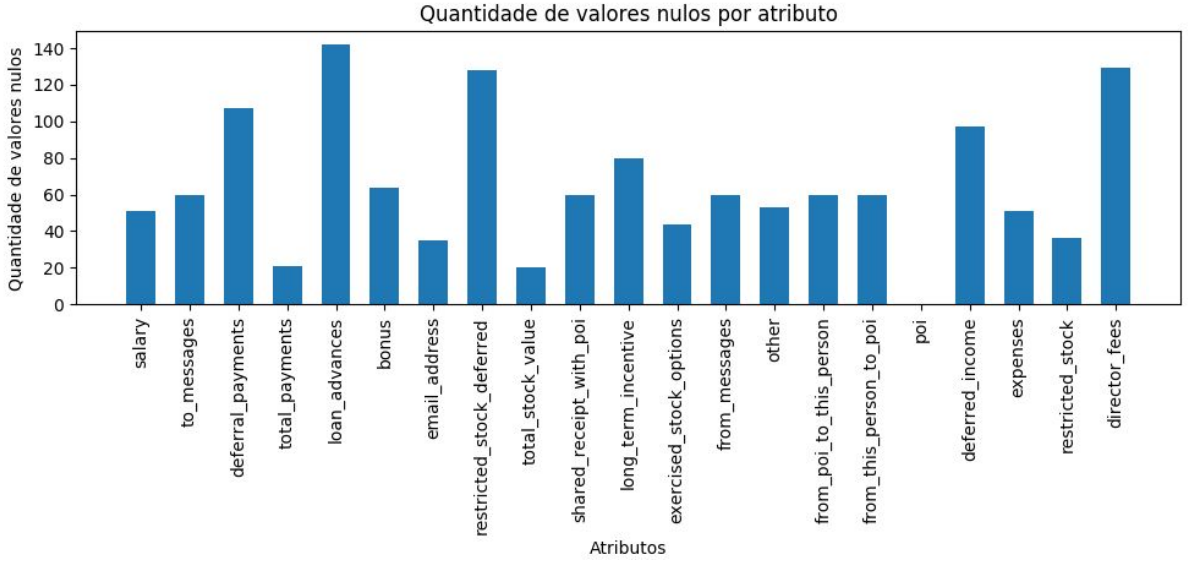
1. Resumo

O objetivo deste projeto é treinar um algoritmo de *Machine Learning* para identificar pessoas que tiveram envolvimento com o escândalo de fraude e corrupção da Enron, chamaremos estas pessoas de POIs (*Person of Interest* ou Pessoa de Interesse). Utilizaremos uma base de dados pré-processados fornecidos pela Udacity com informações financeiras e relativas a emails das pessoas.

2. Exploração dos dados

Na base de dados, possuímos 146 registros, onde 18 são marcados como sendo POI e 128 sendo não-POI. Cada um desses registros, possui 21 atributos, relacionados as finanças da pessoa, aos seus emails e um atributo que define se ela é ou não uma POI. Os atributos possuem muitos valores nulos, o que dificulta a avaliação, segue uma visualização abaixo da quantidade de dados faltantes por atributo da base de dados.



3. Investigação dos Outliers

Ao ordenar os dados em em relação ao salário, foi possível identificar o registro de TOTAL, que não é uma entrada válida na tabela, já que não representa uma pessoa, mas sim o somatório de todas elas.

Nome	Salário
TOTAL	26704229
SKILLING JEFFREY K	1111258
LAY KENNETH L	1072321
FREVERT MARK	1060932
PICKERING MAR R	655037
...	...

Removemos também o registro de LOCKHART EUGENE E, pois todos atributos (fora poi, que é falso) relativos a ele são vazios, portanto, ele não tem dados pertinentes para a análise.

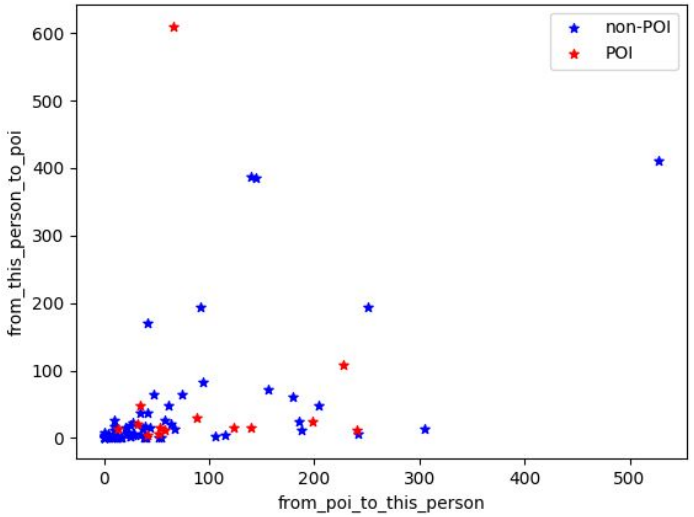
Nome	Quantidade de Atributos nulos
LOCKHART EUGENE E	20
GRAMM WENDY	18

THE TRAVEL AGENCY IN THE PARK	18
WROBEL BRUCE	18
WHALEY DAVID A	18
...	...

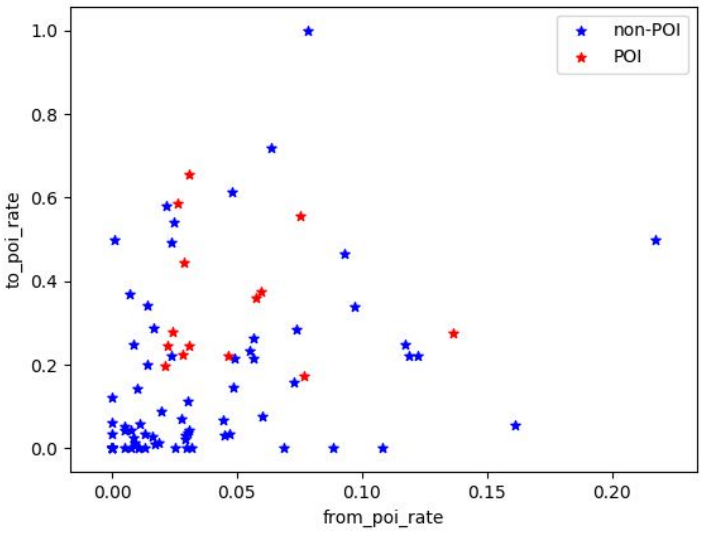
Ao verificar a listagem de nomes, também identificamos o registro inválido de THE TRAVEL AGENCY IN THE PARK que deve ser removido pois não representa uma pessoa.

4. Criação de Novas Features

Primeiramente traçamos um gráfico para checar se os atributos `from_poi_to_this_person` e `from_this_person_to_poi` tinham alguma significância visual para diferenciar POIs de não-POIs.



Como não apresentaram nenhuma divisão visual clara, criamos então dois novos atributos chamados `from_poi_rate` e `to_poi_rate`, que nada mais são que a proporção de `from_poi_to_this_person`, e `from_this_person_to_poi` pelo todo, `to_messages` e `from_messages` respectivamente.



Aparentemente os novos atributos também não possuem um poder representativo para diferenciar os POIs de não-POIs, mas eles serão removidos dos atributos durante a seleção de features caso o algoritmo detecte que não são relevantes.

Executamos o tester.py para ambos os modelos afinados com todas as features levadas em consideração, com e sem os novos atributos.

	F1 Score	**Precisão**	**Abrangência**
Random Forest com novas Features	0.27001	0.49599	0.18550
Random Forest sem novas Features	0.17692	0.29419	0.12650
Decision Tree com novas Features	0.30054	0.34454	0.26650
Decision Tree sem novas Features	0.21522	0.23789	0.19650

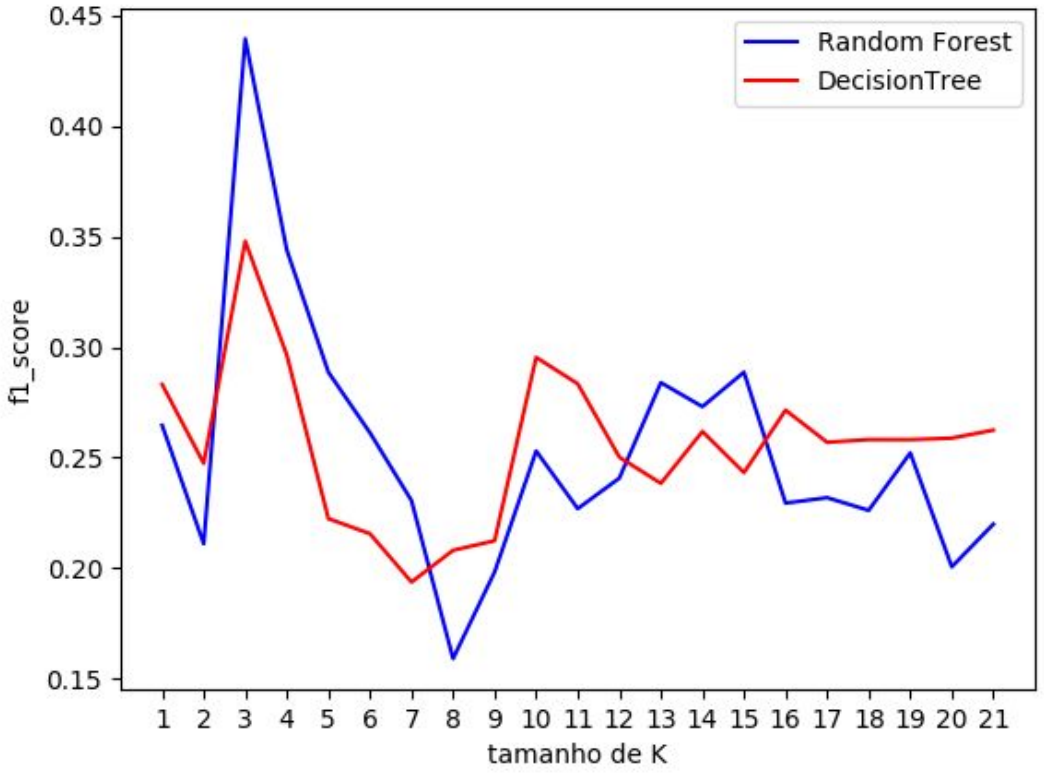
Em um âmbito geral, considerando todas as features, o incremento das novas features teve impacto positivo nos resultados.

5. Seleção de features feita de forma inteligente

Utilizamos o SelectKBest primeiramente para fazer o afinamento dos dois modelos utilizados juntamente com a variação dos parâmetros.

Para traçar um comparativo, executamos em 100 amostras, treinamentos e previsões calculando o f1_score com os modelos afinados, o resultado mostrado no gráfico abaixo é a média destes valores. O ponto mais alto do gráfico está no tamanho de K igual a 3 e em relação ao modelo de RandomForest, portanto utilizaremos esta combinação para exportar o modelo.

Pela seleção dos 3 melhores atributos com SelectKBest (sendo K = 3), iremos utilizar os atributos 'bonus', 'exercised_stock_options' e 'total_stock_value' para compor a lista de atributos, mesmo que pelo afinamento obtivemos K = 4 para RandomForest, por esta situação foi possibilitado avaliar mais casos para validar o melhor K.



6. Ajuste de escala das características feito corretamente

Não foi necessário ajuste de escala, o ajuste não afeta o desempenho dos algoritmos DecisionTree e RandomForest.

7. Escolha um algoritmo

Foram escolhidos os algoritmos RandomForest e DecisionTree, ao executar tester.py (Os classificadores foram gerados com o único parâmetro random_state=42), obtivemos os seguintes resultados:

Modelo	Acurácia	Precisão	Abrangência	F1 Score
Random Forest	0.86133	0.43421	0.13200	0.20245
Decision Tree	0.81760	0.31320	0.30850	0.31083

Observando estes resultados, Decision Tree teve uma performance melhor, inclusive já satisfazendo o requisito mínimo de ter a precisão e a abrangência maiores que 0.3.

8. Discussão da afinação de parâmetros e a performance deles

É necessário fazer o afinamento pois muitas das vezes as configurações definidas por padrão dos classificadores, não nos apresentam o melhor cenário de decisões, e para que consigamos gerar as respostas mais precisas possíveis fazemos a afinação dos parâmetros que podem melhorar a acurácia do nosso modelo, para que se adapte a situação, permitindo que as decisões feitas sejam mais efetivas.

9. Afinação (tuning) do algoritmo

Para afinar os modelos, submetemos cada um deles a 10 amostras, em que em cada amostra testamos todas as combinações dos parâmetros para cada quantidade de melhores atributos selecionados.

Para o RandomForest ao final da afinação obtivemos os seguintes parâmetros e lista melhores atributos:

melhor parâmetro: {'max_features': 'auto', 'min_samples_split': 2, 'bootstrap': False, 'min_samples_leaf': 1}
melhor lista de atributos: ['poi', 'bonus', 'exercised_stock_options', 'salary', 'total_stock_value']

Já para DecisionTree, após o afinamento obtivemos a seguinte relação:

melhor parâmetro: {'min_samples_split': 10, 'criterion': 'gini', 'max_depth': 10}
melhor lista de atributos: ['poi', 'bonus', 'exercised_stock_options', 'total_stock_value']

Então aplicando novamente o código de tester.py, obtivemos os setuintes resultados com nossos modelos afinados:

Modelo	Acurácia	Precisão	Abrangência	F1 Score
Random Forest	0.83715	0.45830	0.32150	0.37790
Decision Tree	0.81685	0.38761	0.32850	0.35562

Podemos verificar uma melhora significativa no desempenho de ambos os modelos, os atributos Precisão, Abrangência e F1 Score melhoraram para os 2 casos, já a acurácia abaixou, na próxima sessão abordaremos a justificativa do porquê abaixar a acurácia foi um fenômeno positivo.

10. Métricas utilizadas

Durante o processo de afinamento as métricas utilizadas foram relativas à abrangência e à precisão, como foram repetidos 10 amostras para cada possível combinação de parâmetros e K melhores atributos, para cada uma das amostras que satisfazia a condição de possuir abrangência e precisão maiores que 0.3, a combinação ganhava um ponto, isso garantiria que a combinação teria uma probabilidade mais alta de satisfazer a condição para qualquer conjunto de dados. Na combinação selecionada de RandomForest, 8 das 10 amostras satisfizeram a condição, e o mesmo ocorreu para DecisionTree.

A definição das métricas abrangência e precisão para este cenário são as seguintes:

Caso a previsão tenha sido de que a pessoa é POI, a precisão é a taxa de confiabilidade de que ela realmente seja uma POI.

Caso a pessoa seja uma POI, a abrangência é a confiabilidade de que o modelo irá prever esta pessoa como POI.

Devido a este fato, se tivermos uma abrangência e precisão nulas, mesmo que a acurácia seja alta, ela não conseguirá prever POIs na base. Um fator curioso de se ressaltar é que caso o modelo treinado assuma que todas as entradas sejam não-POIs, sua acurácia poderá ser alta, pois possuímos um número muito maior de não-POIs do que de POIs em nossa base, porém a abrangência e precisão seriam nulas, impedindo a previsão de qualquer pessoa como POI, isso justifica que mesmo com a acurácia mais baixa e os demais parâmetros mais altos, tivemos um ganho no afinamento dos modelos.

11. Validação

A validação é o processo que utilizamos para garantir que nosso modelo seja o mais livre possível de um enviesamento por ter sido treinado em uma amostra da base de dados com características que não representam a base como um todo . Um exemplo disso seria que caso obtivessem uma amostra que possuiria apenas não-POIs, o treinamento seria feito e todas as classificações treinadas seriam de não-POIs, se a parcela da amostra de teste também fosse completamente composta de não-POIs, obteríamos uma acurácia de 1, que é o melhor caso, porém nosso modelo não estaria devidamente treinado para a representação da base como um todo.

Em nosso projeto fizemos a validação durante o processo de alinhamento dos modelos, em que coletamos 10 amostras diferentes para que o treinamento fosse realizado, e ao comparar ambos os algoritmos e selecionar a quantidade de features mais importantes para o modelo final, geramos 100 amostras para realizar os testes e obter uma média dos resultados, garantindo assim que pela quantidade de amostras estaríamos nos distanciando da possibilidade de um enviesamento.

Todas amostras foram geradas utilizando o método train_test_split de cross_validation da sklearn, variando no atributo random_state.

12. tester.py

O tester.py utiliza o StratifiedShuffleSplit de sklearn.model_selection, ele é um cross validator que assim como outros fornece dados de treinamento e teste para o conjunto de dados, para validar o modelo persistido em my_classifier.pkl, com os dados de my_dataset.pkl utilizando as features de my_feature_list.pkl.

O cross_validator está dividindo e reembaralhando 100 vezes (passado como parâmetro) o conjunto de dados para garantir a validação do modelo e testá-lo em casos que representam de fato a base como um todo.

Como saída ele apresenta o classificador utilizado junto aos seus parâmetros e os resultados do modelo, apresentando a acurácia, abrangência, precisão, f1 score, f2 score, total de predições, número de verdadeiros positivos, falsos positivos, falsos negativos e verdadeiros negativos.

13. Performance do Modelo

O modelo final selecionado foi uma RandomForest com os seguintes parâmetros encontrados no afinamento:

```
melhor parâmetro: {'max_features': 'auto', 'min_samples_split': 2, 'bootstrap': False, 'min_samples_leaf': 1}
```

E utilizando a seguinte lista de features, decididas a partir do gráfico apresentado na seção 5, onde se deveria selecionar os 3 melhores atributos com SelectKBest (note que ‘poi’ não conta como um atributo, ele entra na lista e é o primeiro parâmetro pois é o target do modelo):

```
melhor lista de atributos: ['poi', 'bonus', 'exercised_stock_options', 'total_stock_value']
```

Ao executar tester.py, obtivemos os seguintes resultados:

Modelo	Acurácia	Precisão	Abrangência	F1 Score
Random Forest	0.85808	0.54946	0.43050	0.48276