Projeto Final – Identificação de Fraude nos Emails da Eron

Curso: Fundamentos de Data Science II

Aluno: Filipe Pegollo

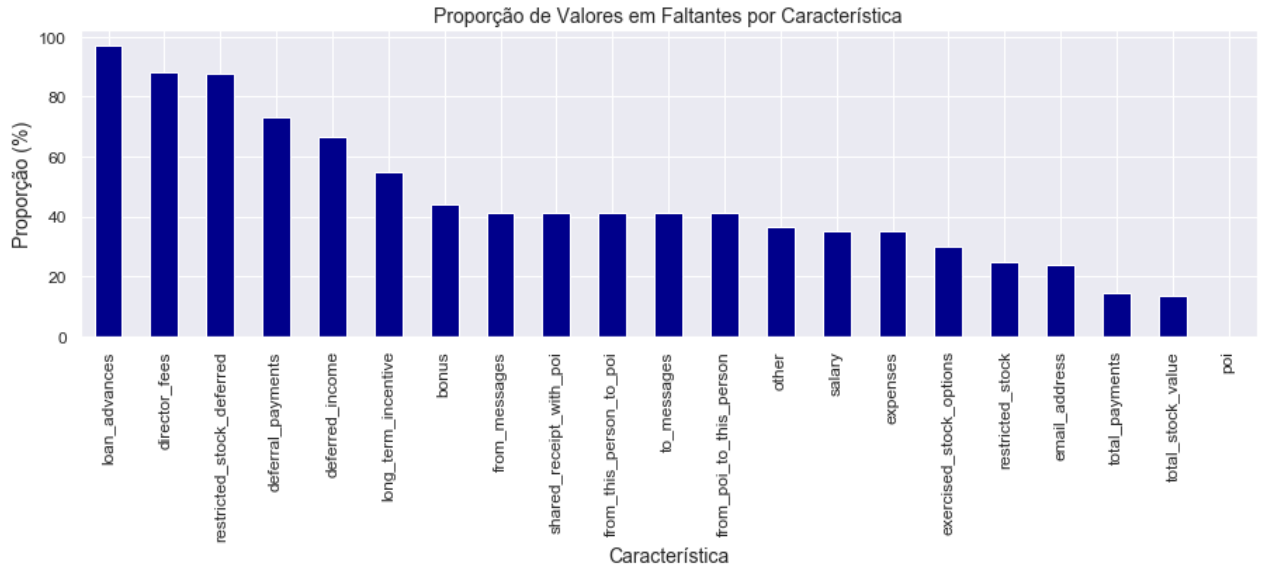
Introdução

Em 2000, Enron era uma das maiores empresas dos Estados Unidos. Já em 2002, ela colapsou e quebrou devido a uma fraude que envolveu grande parte da corporação. Resultando em uma investigação federal, muitos dados que são normalmente confidenciais, se tornaram públicos, incluindo dezenas de milhares de e-mails e detalhes financeiros para os executivos dos mais altos níveis da empresa. Este projeto tem o objetivo de analisar dados financeiros e e-mails da empresa e através de técnicas de machine learning avaliar diversas características que possam identificar pessoas com grandes chances de envolvimento.

Avaliando os Dados

Foi identificado que a base de dados possui 145 registros, não sendo nenhum duplicado, e possuem um total de 21 características contendo informações de financeiras de salários, bonus e quantidades de e-mails enviados e recebidos a POIs (Person Of Interest). Também foi identificado que há 18 registros considerados POIs e 127 Não-POIs. Para ter um panorama do dataset foi feito um levantamento do total de valores faltantes e depois calculada a proporção em relação ao total como podem ser vistos na tabela e gráfico abaixo:

Característica	Itens Ausentes	Proporção
loan_advances	142	97%
director_fees	129	88%
restricted_stock_deferred	128	88%
deferral_payments	107	73%
deferred_income	97	66%
long_term_incentive	80	55%
bonus	64	44%
to_messages	60	41%
from_poi_to_this_person	60	41%
from_messages	60	41%
from_this_person_to_poi	60	41%
shared_receipt_with_poi	60	41%
other	53	36%
salary	51	35%
expenses	51	35%
exercised_stock_options	44	30%
restricted_stock	36	25%
email_address	35	24%
total_payments	21	14%
total_stock_value	20	14%
poi	0	0%

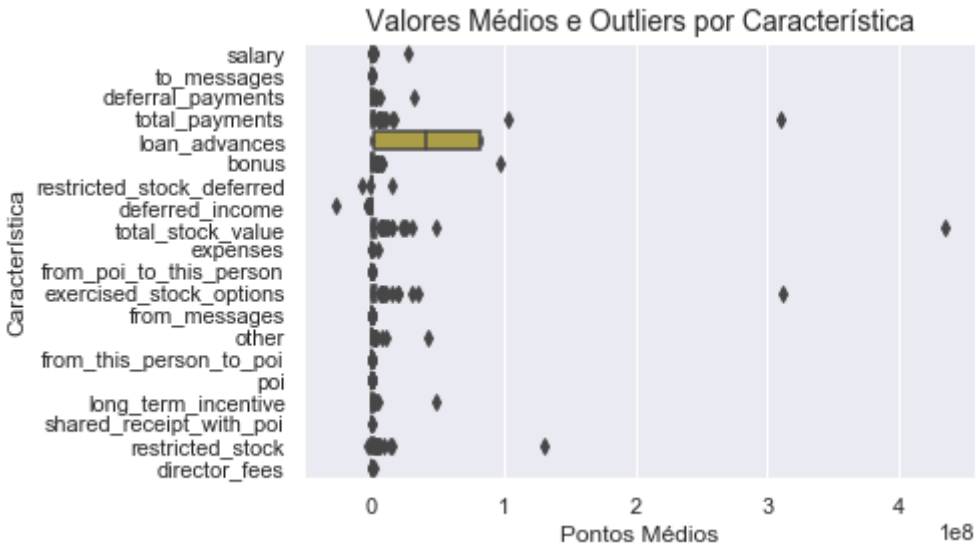


Como pode ser visto algumas características tem mais de 80% de valores faltantes e ‘loan_advances’ chegando a quase 100%, estas características tratam-se valores de ações, empréstimos. Este levantamento traz informações úteis que podem ajudar a selecionar e a entender a fase de seleção de recursos.

Identificação de Outliers

Inicialmente por haver muitas características com valores nulos, foi criado um código para verificar se havia algum registro com todos valores em branco e foi localizado um item identificado como: ‘LOCKHART EUGENE E’. Através da análise visual foi possível identificar o registro ‘THE TRAVEL AGENCY IN THE PARK’ que se trata de uma empresa e não uma pessoa.

Depois para ajudar na identificação de outliers foi criado um gráfico do tipo ‘boxplot’, que agrupa os valores médios das características e mostra pontos com valores fora da média, estes valores precisam ser verificados pois podem afetar negativamente as análises.



Conforme podemos ver no gráfico acima existem algumas características que possuem valores extremamente distantes da média, por isso foi criado um código que identificou o registro ‘TOTAL’ que na verdade não se trata de um registro mas sim do total da base de dados. Para não comprometer as análises os seguintes outliers foram removidos da base de dados:

- LOCKHART EUGENE E
- THE TRAVEL AGENCY IN THE PARK
- TOTAL

Seleção e Otimização de Características

Foram criadas três novas novas características 'prop_bonus', 'prop_from_poi' e 'prop_to_poi' que calculam a proporção de bonus em relação ao salário as proporções de mensagens enviadas e recebidas de POIs em relação ao total de mensagens. Para avaliar o desempenho das características foi criada uma função que utiliza o algoritmo K-best para ranquear e imprimir as melhores, os resultados iniciais mostraram-se favoráveis, já que das três criadas duas estavam entre as 10 melhores:

10 Melhores Características:		
Posição	Característica	Score
1	exercised_stock_options	24,82
2	total_stock_value	24,18
3	bonus	20,79
4	salary	18,29
5	prop_to_poi	16,41
6	deferred_income	11,46
7	prop_bonus	10,78
8	long_term_incentive	9,92
9	restricted_stock	9,21
10	total_payments	8,77

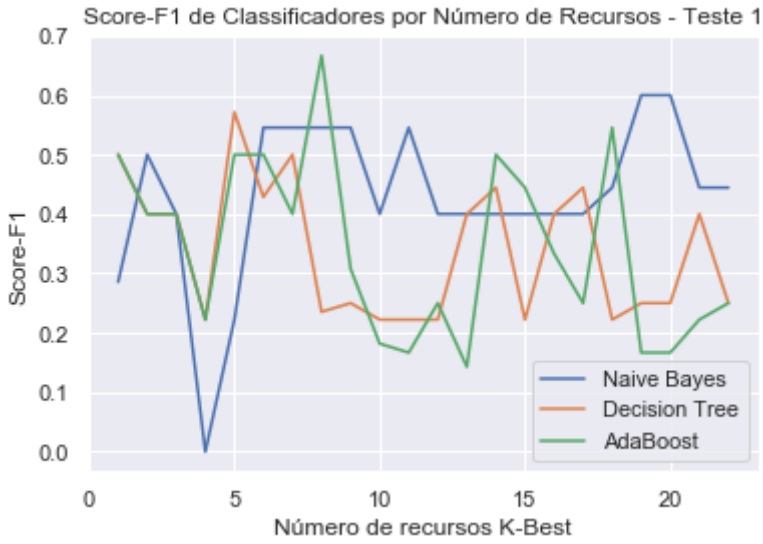
Durante os testes finais foi possível ver o impacto e ganho de performance ao utilizar as novas características:

Teste	Com novas Features	Accuracy	Precision	Recall
1	Não	0,80793	0,27021	0,25900
1	Sim	0,81385	0,39645	0,40200
2	Não	0,80507	0,26332	0,25700
2	Sim	0,81546	0,40292	0,40400
3	Não	0,80473	0,2681	0,26830
3	Sim	0,81438	0,40019	0,40400

O número de características utilizadas foi selecionado sistematicamente através da criação de um algoritmo, que mede o desempenho de cada classificador de acordo com cada número de recursos, informando o número ideal e imprimindo um gráfico para avaliação do desempenho, os classificadores testados foram: Naive Bayes, Decision Tree e AdaBoost.

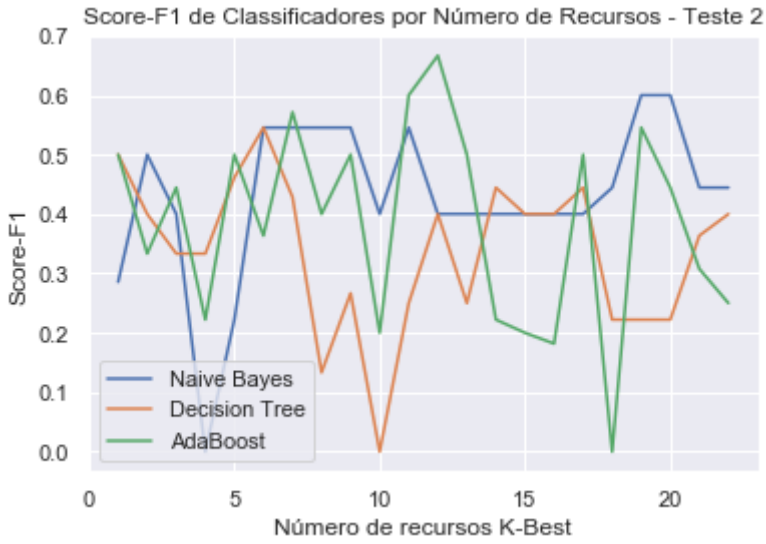
O objetivo final deste projeto é atingir pelo menos 0,3 de precisão e recall, por isso a métrica utilizada para avaliar os classificadores e número de recursos foi Score-F1, esta métrica combina valores de precisão e recall em uma pontuação equilibrada.

Teste de Classificadores 1/3:



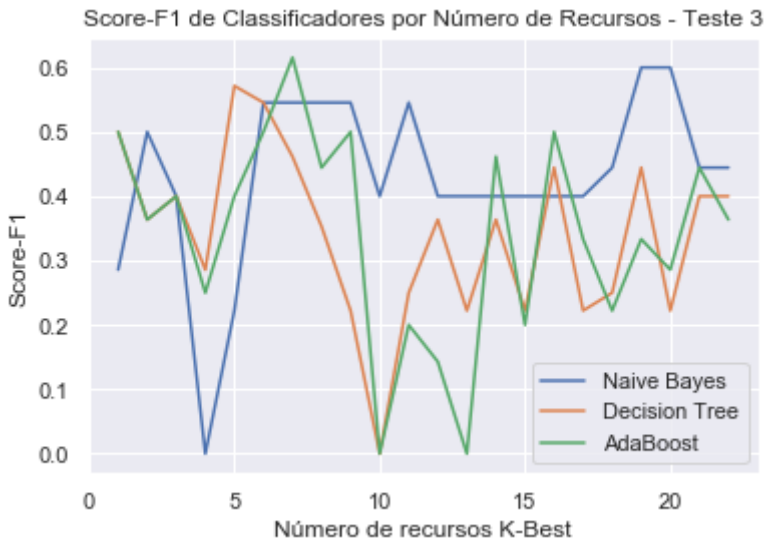
Melhor Número de Recursos e Score-F1		
Score-F1	classificador	recursos
0,600	Naive Bayes	20
0,571	Decision Tree	5
0,666	Random Forest	8

Teste de Classificadores 2/3:



Melhor Número de Recursos e Score-F1		
Score-F1	classificador	recursos
0,600	Naive Bayes	20
0,545	Decision Tree	6
0,666	Random Forest	12

Teste de Classificadores 3/3:



Melhor Número de Recursos e Score-F1		
Score-F1	classificador	recursos
0,600	Naive Bayes	20
0,571	Decision Tree	5
0,615	Random Forest	7

Melhor classificador:

Classificador	f1_score	acuracia	recursos
AdaBoost	0.666667	0.930233	12

Pelo fato do número de características ideal ser variável ele será selecionado sistematicamente durante a execução do algoritmo.

Escolha e Ajuste de Parâmetros do Classificador

Durante os testes realizados no desenvolvimento do projeto o classificador AdaBoost teve o melhor desempenho, ainda assim o Naive Bayes teve um desempenho muito semelhante a algumas vezes superior, porém o escolhido foi o AdaBoost por ter um maior potencial de ajustes de parâmetros. O AdaBoost é um classificador que se adapta criando cópias onde o peso de classificações negativas são ajustadas e pode ser utilizado em conjunto com outros classificadores, neste caso foi utilizado o AdaBoost com DecisionTree que é o classificador base padrão.

Testar diversos parâmetros é importante para que se obtenha os melhores ajustes do classificador para cada caso, por isso foi utilizado o GridSearchCV para encontrar a melhor combinação de parâmetros e recursos.

Os parâmetros testados foram:

- **base_estimator__criterion:** Mede a qualidade da divisão, gini através de impurezas e entropy através de ganho de informação.
- **base_estimator__splitter:** Escolhe os nós de divisão, best escolhe a melhor divisão ou random para divisões aleatórias.
- **n_estimators:** Número máximo de estimadores que serão criados.
- **learning_rate:** Peso de contribuição de cada modelo.

O classificador com os parâmetros ajustados ficou da seguinte forma:

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator =
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False, random_state=11, splitter='best'),
learning_rate=1, n_estimators=50, random_state=None)
```

Avaliação e Validação do Algoritmo Final

Após o ajuste de parâmetros do classificador escolhido é preciso avaliar o desempenho e validar os resultados, para evitar um erro comum de um algoritmo sobreajustado de teste que pode obter alta precisão com os dados atuais, porém pode não se adaptar com a variabilidade de novos dados.

Para validar os testes finais foram utilizados os scripts do arquivo tester.py dividindo a base de dados em dados de treinamento e dados de teste, utilizando para validação cruzada o StratifiedShuffleSplit, indicado para bases de dados com poucas observações como esta, ele é semelhante ao StratifiedKFold e ShuffleSplit, mas ao invés de embaralhar os dados apenas uma vez este embaralha antes de cada divisão e os conjuntos de teste podem se sobrepor. Ele está configurado com o parâmetro random_state=42 que define um número fixo de sequências aleatórias que são geradas.

Métricas para Avaliação de Performance

Utilizar apenas uma métrica para avaliação de um algoritmo pode não ser suficiente por apresentar um ótimo número mas ignorar outros pontos relevantes como falsos positivos, por isso é importante avaliar mais de uma métrica para ter uma visão mais detalhada. Neste projeto inicialmente foi verificada a accuracy (acurácia) que mostra o percentual de acertos gerais do classificador, mas está métrica não mostra o desempenho por classe, sendo assim foi avaliado também precision (precisão) que verifica das predições positivas quantos realmente eram. E recall que verifica o número de vezes que a classe predita aparece nos exemplos de teste.

Testes finais:

Teste	Accuracy	Precision	Recall
1	0,81385	0,39645	0,40200
2	0,81546	0,40292	0,41400
3	0,81438	0,40019	0,41400

Conclusão

Este projeto visou criar um algoritmo investigar dados para classificar pessoas e calcular as probabilidades de estarem envolvidas ou não nos escândalos de corrupção do caso Eron, atingindo precisão e recall de pelo menos 0,3. Durante os testes o algoritmo apresentou os seguintes resultados:

- **Acurácia:** A taxa média foi de 80%, no contexto deste projeto significa que o nosso algoritmo teve no geral 80% de acertos ao indicar que uma pessoa está ou não envolvida.
- **Precisão**: Os resultados apresentados ficaram entre 0,3 e 0,4 isso quer dizer que quando o nosso algoritmo classifica como positivo o envolvimento de uma pessoa tem em média um acerto de 30 a 40%.
- **Recall:** Este também ficou entre 0,3 e 0,4 o que significa que dos envolvidos reais analisados nos testes 30 a 40% são identificados pelo algoritmo.

Em relação a seleção e criação de novos recursos podemos ver que houve uma ligeira melhora na comparação dos resultados. Apesar de ter utilizado uma base de dados pequena com poucos dados para teste e treino, é possível afirmar que o projeto atingiu o seu objetivo satisfatoriamente, porém ainda assim é possível melhorar buscando a utilização de outros classificadores e técnicas de avaliação e validação formando um ciclo de constante melhora.

Fontes

Documentação Scikit-Learn:
https://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-
https://scikit-learn.org/stable/modules/cross_validation.html
https://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

Visualizing K-Means Clustering:
https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Overfitting - Sobreajuste - Wikipedia
https://pt.wikipedia.org/wiki/Sobreajuste

Repositórios do Github:
https://github.com/gapkim/Enron_Fraud/blob/master/poi_id.py
https://github.com/oliveira-marcio/ml-enron-fraud/blob/master/poi_id.py#L261
https://github.com/marcioozorio/deteccao_fraude_machine_learning_Enron/blob/master/poi_id.py
https://github.com/vyniciuss/enron_machine_learning

Identificando e Tratando Outliers - Minerando Dados
http://minerandodados.com.br/index.php/2017/11/13/tratando-outliers-python/

09 Métricas de Avaliação de Modelos - Minerando Dados
http://minerandodados.com.br/index.php/2017/10/10/cafe-com-codigo-09-metricas-de-avaliacao-de-modelos/

Métricas Comuns em Machine Learning - As máquinas que pensam
https://medium.com/as-m%C3%A1quinas-que-pensam/m%C3%A9tricas-comuns-em-machine-learning-como-analisar-a-qualidade-de-chat-bots-

Adaboost Classifier - Chris Albon
https://chrisalbon.com/machine_learning/trees_and_forests/adaboost_classifier/

AdaBoost Classifier in Python - Datacamp
https://www.datacamp.com/community/tutorials/adaboost-classifier-python

Difference Between Stratifiedkfold and Stratifiedshufflesplit - CoreDump
https://coredump.pt/questions/45969390/difference-between-stratifiedkfold-and-stratifiedshufflesplit-in-sklearn

Bibliotecas Utilizadas:

- Pandas 0.23.4
- Numpy 1.15.1
- Scikit-Learn 0.19.2
- Matplotlib 2.2.3
- Seaborn 0.9.0

Versão do Python: 3.7.0