João Vitor Deon

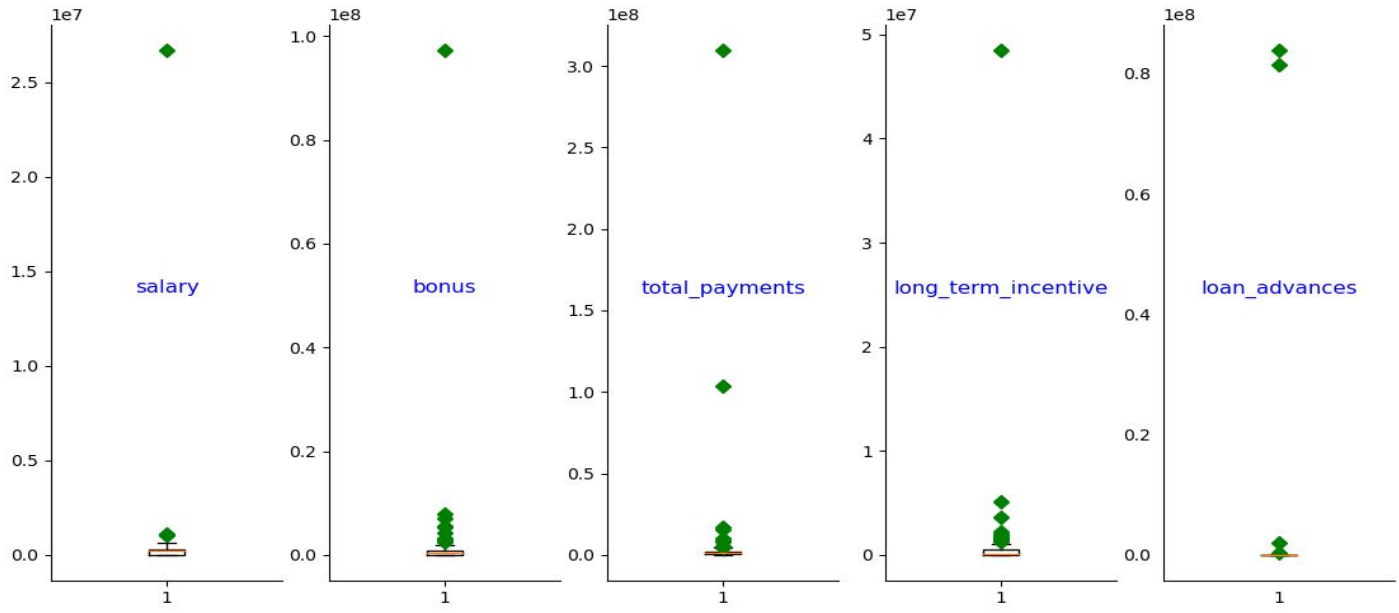
Udacity NanoDegree project 4 : Machine Learning (Enron DataBase)

1)

Esse projeto visa aplicar um modelo de Machine Learning para identificarmos possíveis pessoas envolvidas nos escândalos financeiros da empresa americana Enron.

Nós temos diversas informações sobre os funcionários e colaboradores da empresa, como dados financeiros diversos de cada pessoa, além de milhares de emails. Vamos utilizar a principal biblioteca de ML , o Sklearn.

Antes da limpeza dos dados e remoção de outliers o conjunto de dados possui 146 registros. Após algumas análises feitas com boxplots, podemos perceber alguns outliers em atributos financeiros como na figura abaixo:

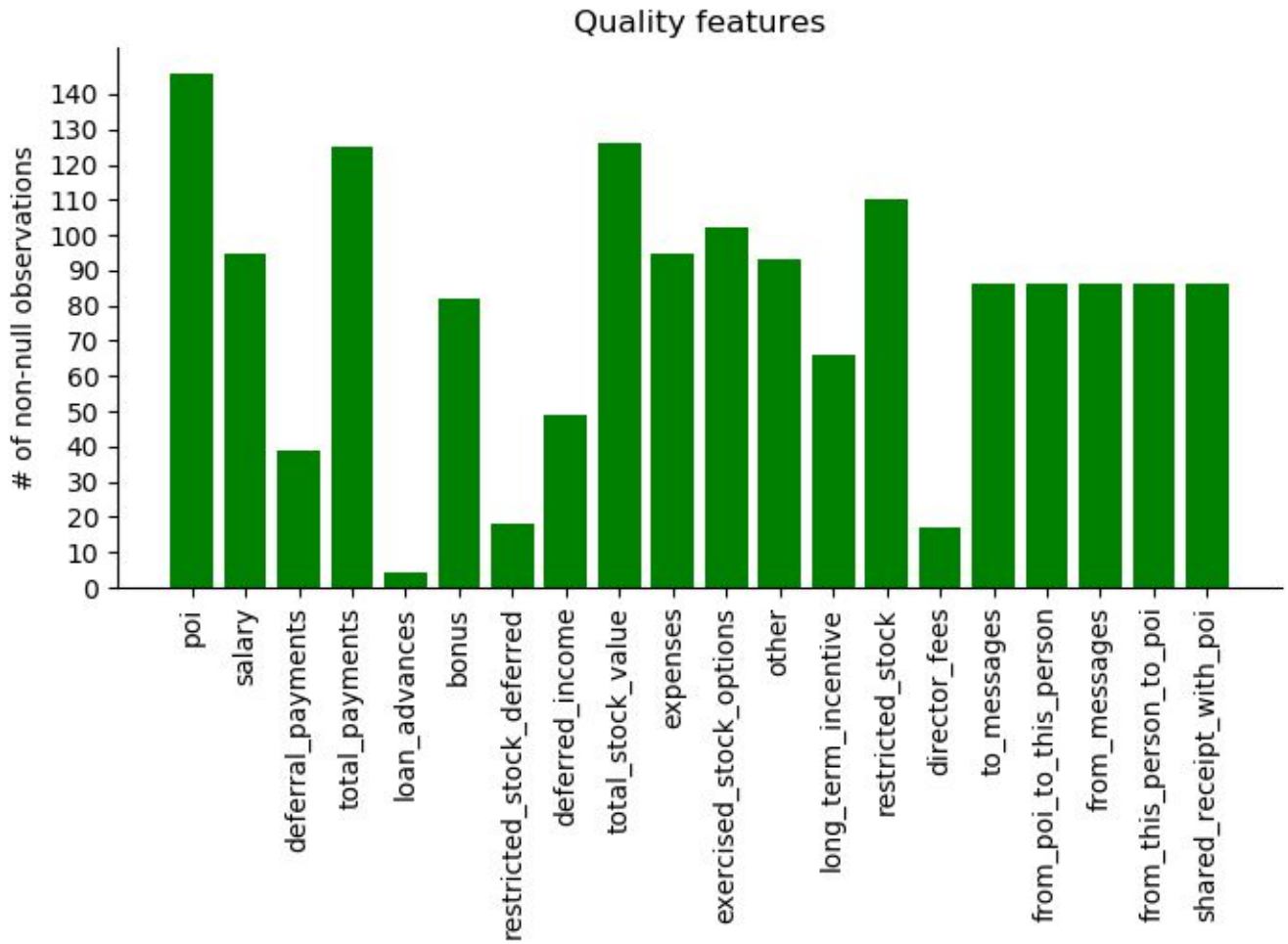


Esses outliers representavam dois registros em geral: “TOTAL” , que provavelmente é um erro de imputação de dados; e “LAY KENNETH L”, que é uma Pessoa de Interesse(POI), logo não iremos excluí-la do dataset.

Também excluímos um registro com nome “THE TRAVEL AGENCY IN THE PARK”, que também parece ser um erro de input e não acrescenta nenhuma informação para nosso modelo de ML.

'LOCKHART EUGENE E' também foi deletado por não possuir valores não-nulos.

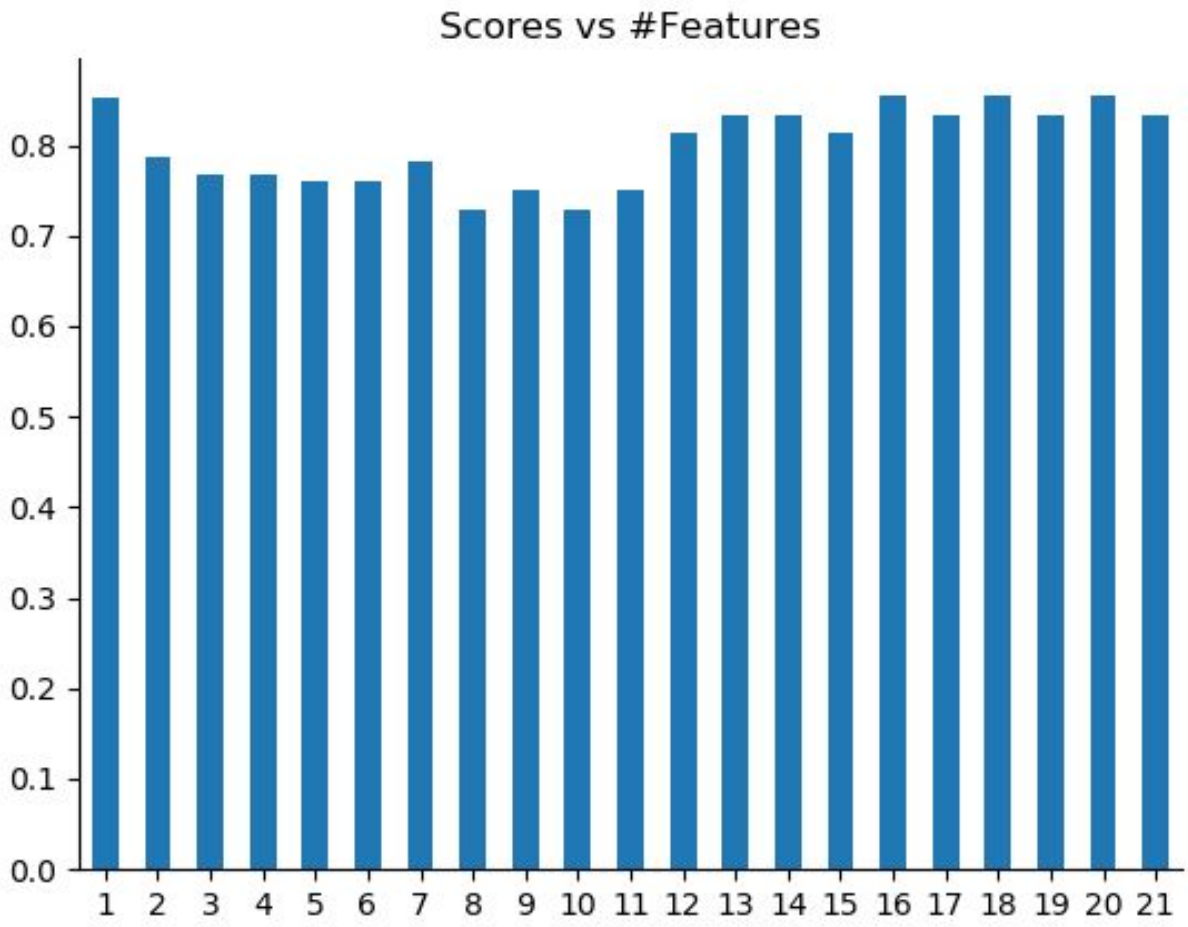
Abaixo temos algumas informações - antes de limpeza dos dados - de qualidade do nosso dataset, como número de POI’s e a quantidade de dados não nulos de cada variável:



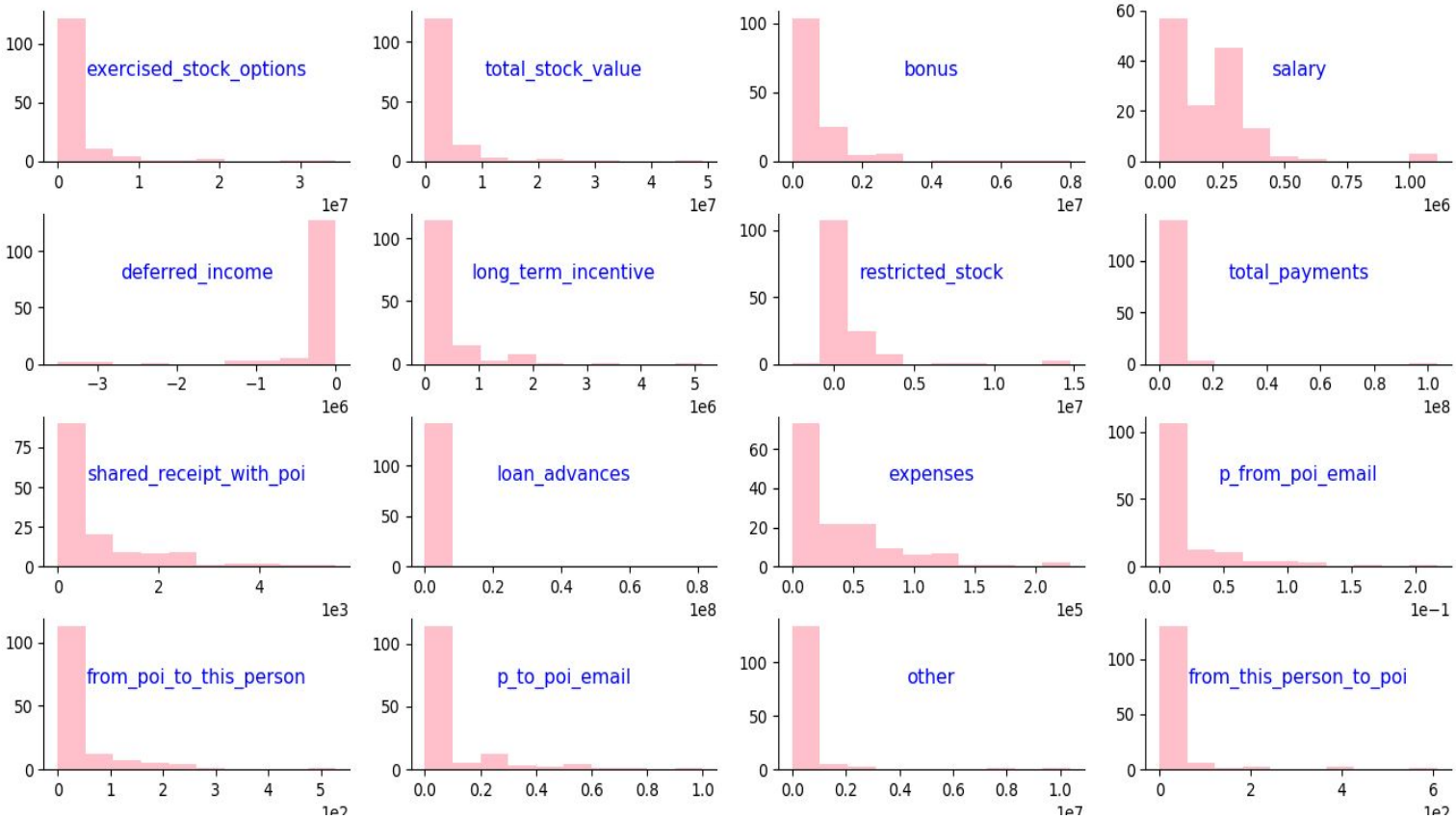
```
-------------------------

Count of POIs:  18
Count of non-POIs:  128

-------------------------
```

2)

Para descobrir o melhor número de variáveis a serem usadas, foi utilizado o algoritmo ‘SelectKBest’ para selecionar as melhores features e usando o classificador Decision Tree Classifier com todos os parâmetros padrões. O melhor desempenho (score) do classificador foi quando utilizamos as 16 melhores features:



Abaixo temos vários histogramas para analisarmos a quantidade e qualidade de dados que temos das K melhores variáveis que escolhemos com o algoritmo 'SelectKBest':



Com esses gráficos das 16 melhores features em ordem decrescente podemos notar uma quantidade grande de zeros na maioria das variáveis, o que pode no início pode nos indicar que nosso modelo não vai conseguir ter um bom êxito. Porém isso pode ser útil pois os dados que fogem da maioria da tendências - como esta de apresentar muitos zeros - podem ser Pessoas de Interesse.

Foram criadas duas novas features : "p_from_poi_email" que é uma proporção de emails recebidos de POI's dividido pelo total de emails que essa pessoa recebeu ; e "p_to_poi_email" ,que é a porporção de emails enviados a POI's dividido pelo total de emails enviados pela pessoa.

Essas duas novas features estão entre as 16 melhores variáveis.

3)

O classificador escolhido foi o ‘Decision Tree Classifier’ por apresentar melhores resultados comparados ao Random Forest Classifier e ao GaussianNB, considerando ‘accuracy’ , ‘precision’ e ‘recall’.

Para achar os melhores parâmetros dos três algoritmos citados acima, utilizei o ‘GridSearchCV’ e o resultado foi este:

```
GaussianNB:  {'var_smoothing': 0.1}
DecisionTreeClassifier:  {'criterion': 'entropy', 'min_samples_split': 25}
RandomForestClassifier:  {'criterion': 'entropy', 'min_samples_split': 10}
----------------------------------------------------------------------
```

Como veremos abaixo, dos três classificadores escolhidos, o Random Forest Classifier e o Decision Tree Classifier tiveram resultados bem parecidos, mas em geral e considerando o ‘recall’, o Decision Tree Classifier se saiu um pouco melhor.

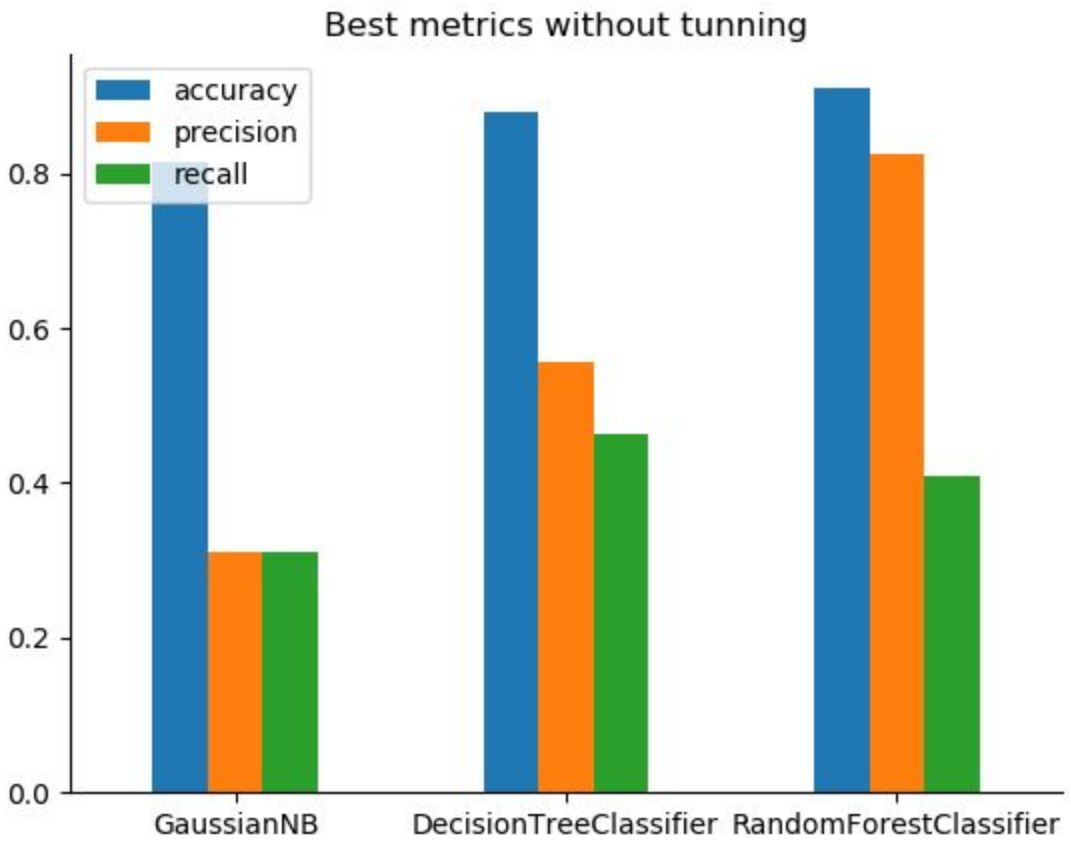
Aqui nós poderíamos escolher qualquer um desses dois, pois nas métricas ‘accuracy’ e ‘recall’ o Decision Tree Classifier se saiu um pouco melhor que o Random Forest, porém este último ganhou se compararmos a métrica ‘precision’.

4)

O ajuste de parâmetros dos algoritmos classificadores é de fundamental importância para um bom modelo de ML, pois cada algoritmo tem seus próprios métodos e meios de tentar treinar e predizer seus resultados finais. Sem o ajuste desses parâmetros estamos sujeitos aos métodos ‘mainstreams’ dos algoritmos, o que nos faz generalizar muito nosso modelo, elevando o viés na maioria das vezes.

Se queremos um bom modelo, temos que entender qual é o problema primeiro, depois ajustar os dados e parâmetros para esse problema específico.

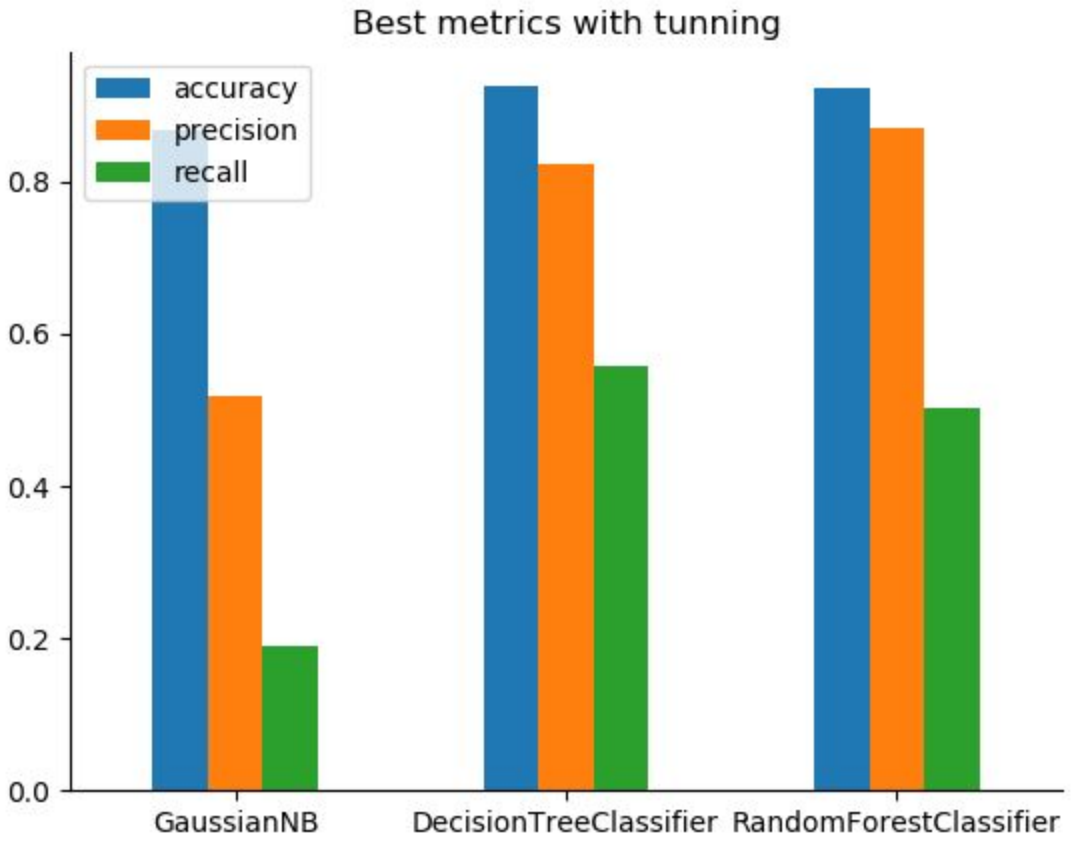
Sem o ajuste de parâmetros nós obtivemos tais resultados dos três classificadores comparados:



	accuracy	precision	recall
GaussianNB	0.815800	0.309536	0.3100
DecisionTreeClassifier	0.879000	0.555422	0.4635
RandomForestClassifier	0.909667	0.824121	0.4100

Percebemos aqui que o Random Forest Classifier obteve um bom desempenho de ‘accuracy’ e ‘precision’ mesmo sem o ajuste de parâmetros, apenas usando as 16 melhores features.

Já com o ajuste de parâmetros (mostrados na questão 3) temos os seguintes resultados dos três algoritmos de classificação:



	accuracy	precision	recall
GaussianNB	0.868400	0.51776	0.1895
DecisionTreeClassifier	0.925200	0.82327	0.5590
RandomForestClassifier	0.923667	0.87013	0.5025

Aqui temos uma melhora considerável do desempenho de todos os algoritmos, e isso foi feito utilizando o GridSearchCV, como mostrado na questão 3.

5)

Validação significa validar seu modelo usando uma parte dos dados para treinar o algoritmo e outra parte para testar e poder tirar métricas analíticas de desempenho desse modelo. Aqui temos que ter um 'tradeoff' entre viés e variância. Se ele fitar muito bem os dados de teste, ocorrerá um Overffiting provavelmente, e se ele deixar de pelo menos aprender alguma tendência nesses dados, ele irá sofrer para predizer os dados de treino, o que caracteriza um alto viés.

A validação usada nos três classificadores dentro do GridSearchCV foi o Cros Validation do próprio GridSearchCV. Porém , para descobrir as métricas dos algoritmos já tunados foi utilizado o 'StritifiedShuffleSplit' com número de folds = 1000.

O uso deste último validador é de suma importância quando trabalhamos com conjuntos de dados pequenos, e esse é o nosso caso analisando os dados da Enron.

Ele permite fazer vários testes de validação cruzada, e em que cada teste ele separa os dados em K-folds(o nome 'Stritified' vem do algoritmo StritifiedKFold que também faz uma validação de dados cruzada), mistura os dados (daí o nome 'shuffle') de teste, o que permite aumentar a confiança e proporção de POI's escolhidos em cada rodada de teste.

6)

Para avaliar o desempenho dos classificadores foram usadas 3 métricas principais: 'accuracy', 'precision' e 'recall'.

Dentro do 'StritifiedShuffleSplit' o classificador escolhido (Decision Tree Classifier) teve os seguintes resultados já mostrados acima:

ACCURACY(acurácia): 93% ,aproximadamente.

PRECISION(precisão): 82% , aproximadamente.

RECALL(revocação): 56%, aproximadamente.

Essas métricas mostram um bom desempenho do modelo em geral, e mostram um desempenho bem superior a 30% de 'precision' e 'recall' pedidos no trabalho.

Essa precisão indica que 82% das pessoas que eu classifiquei como POI' era efetivamente POI, o que significa que 18% das vezes ele indicou uma pessoa inocente como POI, o que não é tão grave para este exemplo da Enron.

Já a revocação mostra que em 56% das vezes ele previu uma pessoa como POI dentre todas as POI's. Ou seja, 44% das vezes ele prediz como não POI uma pessoa que era POI de fato, o que pode indicar um problema, visto que muitos POI's sairão como inocentes.