Identificar fraude no Email da Enron

Visão geral do Projeto:

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.

Neste projeto desenvolvemos o indentificador de person of interest(POI) utilizando os algoritmos de machine learning, baseado em dados financeiros e e-mails fornecido conforme o link. Uma pessoa de interesse (POI) é o funcionário indiciado, fecharam acordos com o governo ou testemunharam em troca de imunidade no processo.

Enron Submission Free-Response Questions:

1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?

O objetivo deste projeto é escolher uma combinação de features dos antigos funcionários da Enron e um algoritmo apropriado de machine learning para prever se essa pessoa é considerada uma pessoa de interesse (POI) ou não. Este é um problema de classificação supervisionada, pois estamos tentando prever o resultado discreto que é binário (0 e 1). Recebemos um conjunto de dados em que temos a informação se a pessoa é uma pessoa de interesse (POI) ou não. A finalidade é obter as previsões mais precisas quando aplicarmos nossos algoritmos de machine learning nos testes.

Dataset:

- Número total de pessoas no conjunto de dados: 146
- Número total de pessoas de interesse (POI) no conjunto de dados: 18
- Número total de pessoas de não interesse (non POI) no conjunto de dados: 128
- Cada pessoa possui 21 features

Quantidade total de valores faltantes nos features: 1358

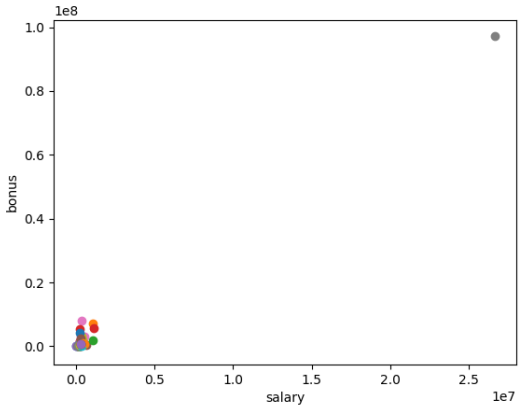
- ('salary', 51)
- ('to_messages', 60)
- ('deferral_payments', 107)
- ('total_payments', 21)
- ('long_term_incentive', 80)
- ('loan_advances', 142)
- ('bonus', 64)
- ('restricted_stock', 36)
- ('restricted_stock_deferred', 128)
- ('total_stock_value', 20)
- ('shared_receipt_with_poi', 60)

- ('from_poi_to_this_person', 60)
- ('exercised_stock_options', 44)
- ('from_messages', 60)
- ('other', 53)
- ('from_this_person_to_poi', 60)
- ('deferred_income', 97)
- ('expenses', 51)
- ('email_address', 35)
- ('director_fees', 129)

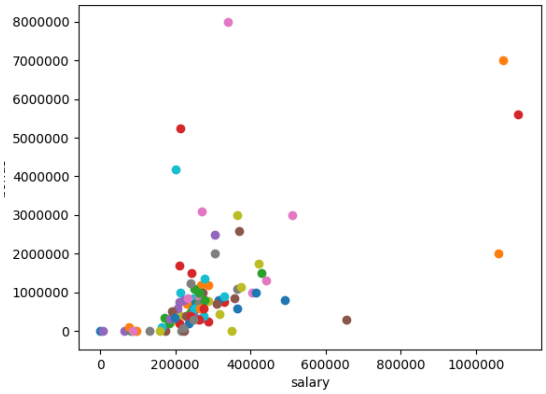
Podemos verificar que 'deferral_payments', 'long_term_incentive', 'restricted_stock_deferred' e 'director_fees' possui uma quantidade relevante de campos sem valores.

Outliers:

Escolhi os features ‘salary’ e ‘bonus’ para visualização dos dados num gráfico disperso. Surpreendentemente foi encontrado outlier. Quando verifiquei manualmente os dados, o valor era referente ao registro ‘TOTAL’. Além desse ponto encontrei um outro registro que é ‘THE TRAVEL AGENCY IN THE PARK’ na pesquisa. Como não havia nenhum relacionamento com os funcionários, removi os dados referente a esses dois.



Antes de remover os outliers



Depois de remover os outliers

Mesmo após a retirada do valor ‘TOTAL’ há alguns dados mais dispersos no gráfico, não foi removido pois me ajudarão a encontrar as pessoas de interesses (POI) posteriormente.

2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.

Apliquei diferentes algoritmos como GaussianNB, DecisionTreeClassifier, SVC e RandomForestClassifier em todo o conjunto de features. Os resultados estão mostrados abaixo:

Classifier	Accuracy	Precision	Recall
GaussianNB	0.39	0.8	0.13
DecisionTreeClassifier	0.83	0.2	0.25
SVC	0.88	0	0
RandomForestClassifier	0.90	0.2	1.0

Como podemos verificar, exceto o classificador GaussianNB possuem boa accuracy. Porém a precision e recall estão zeradas.

Então criei dois novos features:

- 'fraction_from_this_person_to_poi'
- 'fraction_from_poi_to_this_person'

Fórmula para a criação dos dois:

- 'fraction_from_this_person_to_poi' = 'from_this_person_to_poi' / 'from_messages'
- 'fraction_from_poi_to_this_person' = 'from_poi_to_this_person' / 'to_messages'

Lógica dessa fórmula é que pessoas de interesse (POI) provavelmente envie ou receba mais e-mails das outras pessoas de interesse (POI).

Segue abaixo desempenho dos algoritmos novamente, agora com novo feature.

Classifier	Accuracy	Precision	Recall
GaussianNB	0.41	0.8	0.14
DecisionTreeClassifier	0.83	0.2	0.25
SVC	0.88	0	0
RandomForestClassifier	0.90	0.2	1.0

Após inclusão dos 2 features, taxa de accuracy aumentou para o classificador GaussianNB. Então decidi classificar meus features de acordo com as pontuações de F1 usando GaussianNB. Escolhi o GaussianNB porque a taxa de accuracy foi bem maior após incluir os 2 novos features criados em comparação com as outras. Segue os resultados:

feature	precision	recall	F1
poi	1.00	1.00	1.00
total_stock_value	0.64	0.28	0.39
exercised_stock_options	0.51	0.28	0.37
bonus	0.53	0.22	0.31
deferred_income	0.51	0.22	0.31
long_term_incentive	0.56	0.17	0.26
restricted_stock_deferred	0.15	1.00	0.26
director_fees	0.15	1.00	0.26
fraction_from_this_person_to_poi	0.27	0.15	0.19
salary	0.52	0.11	0.19
restricted_stock	0.52	0.11	0.19
total_payments	0.44	0.11	0.18
loan_advances	0.50	0.05	0.09
other	0.25	0.05	0.09
from_this_person_to_poi	0.18	0.06	0.08
shared_receipt_with_poi	0.17	0.05	0.08
deferral_payments	0.00	0.00	0.00
expenses	0.00	0.00	0.00
to_messages	0.00	0.00	0.00
from_poi_to_this_person	0.00	0.00	0.00
from_messages	0.00	0.00	0.00
fraction_from_poi_to_this_person	0.00	0.00	0.00

Retirando as features que tiveram valor zero em precision, recall e F1, ficamos com:

- poi
- total_stock_value
- exercised_stock_options
- bonus
- deferred_income
- restricted_stock_deferred
- director_fees
- long_term_incentive
- fraction_from_this_person_to_poi
- salary
- restricted_stock
- shared_receipt_with_poi
- loan_advances
- total_payments
- other

Embora as features ‘restricted_stock_deferred’ e ‘director_fees’ estejam na lista acima, elas possuem muitos registros com valores faltantes. Sendo assim removi elas da minha lista de feature.

Ficando assim minha lista de features:

- poi
- total_stock_value
- exercised_stock_options
- bonus
- deferred_income
- restricted_stock_deferred
- director_fees
- long_term_incentive
- salary
- restricted_stock
- shared_receipt_with_poi
- loan_advances
- total_payments
- other

Feature 'fraction_from_this_person_to_poi' que criei novo não está nesta minha lista, para verificar posteriormente se tem algum efeito no desempenho do classificador final.

Segue resultado da execução da lista informado acima nos classificadores:

Classifier	Accuracy	Precision	Recall
GaussianNB	0.86	0.4	0.4
DecisionTreeClassifier	0.72	0	0
SVC	0.88	0	0
RandomForestClassifier	0.86	0.2	0.33

Verifiquei que a taxa de accuracy melhorou para GaussianNB, SVC se manteve e DecisionTreeClassifier e RandomForestClassifier cairam.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

Escolhi o classificador GaussianNB. Realizei testes com DecisionTreeClassifier, SVC e RandomForestClassifier.

4. What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).

Embora a taxa de accuracy esteja alta, precision e recall estão com valor igual a zero na maioria dos casos. Isso significa que accuracy, precision, recall ou algum outro atributo não esteja customizado corretamente nesse conjunto de dados específico que temos. Para melhorar a taxa de precision e recall em todos os classificadores, precisamos ajustar os parâmetros passados pela função de cada classificador para nos dar melhores resultados possíveis.

Experimentei diferentes algoritmos com diferentes combinações na passagem de parâmetros das funções. Assim achei interessante a utilização da funcionalidade de GridsearchCV e Pipeline.

GaussianNB

Utilizei o SelectKBest no pipeline para selecionar o melhor valor para o K, Principal Component Analysis(PCA) para reduzir a dimensionalidade dos features e minmaxscaler para a escala com o proposito de popular o classificador GaussianNB. GridSearchCV foi utilizado para testar diferentes valores de 'K' no SelectKBest e 'n_components' no PCA.

- k = [9,10,11]
- n_components = [6,7,8]

Melhor resultado foi gerado quando k = 10 e n_components = 7, f_classif foi usado como 'score_func' e ficou como:

Training time: 0.274229049683
Predicting time: 0.00026798248291
Naive Bayes accuracy: 0.860465116279
f1 score: 0.4
precision score: 0.4
recall score: 0.4

DecisionTreeClassifier

Nesse classificador utilizei diferentes parâmetros como:

- min_samples_split = [2,3,4]
- criterion = ['gini', 'entropy']

Depois de popular com esses parâmetros no GridSearchCV, encontrei a melhor combinação que é criterion = 'gini' e min_samples_split = 4. Na qual resultou com os seguintes valores:

Training time: 0.135
Predicting time: 0.0
Decision Tree accuracy: 0.720930232558
f1 score: 0.0
precision score: 0.0
recall score: 0.0

SVC

Nesse caso adicionei 'minmaxscaler' no pipeline e testei diferentes valores no kernel, C e ganma:

- svc__kernel = ['linear','rbf']
- svc__C = [0.1,1,10,100,1000]
- svc__gamma = [1e-3,1e-4,1e-1,1,10]

Depois de popular com esses parâmetros no GridSearchCV, encontrei a melhor combinação que é kernel = 'linear', ganma = 0.001 e C = 1000. Na qual resultou com os seguintes valores:

Training time: 0.802779912949
Predicting time: 0.000152111053467
SVC accuracy score: 0.860465116279
f1 score: 0.25
precision score: 0.2
recall score: 0.333333333333

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation é uma forma de provar o desempenho do algoritmo de machine learning, ou seja, testar o quão bem o seu modelo foi treinado. Um erro de validação clássico é testar seu algoritmo sobre os mesmos dados em que foi treinado. Sem separar o conjunto de dados de treino com o conjunto de dados de testes, isso é a dificuldade de determinar o quão bom o seu algoritmo generaliza os novos dados.

Nas minhas validações separei os meus conjuntos de dados de treino e teste usando a função 'train_test_split' do módulo 'sklearn.cross_validation', onde 70% de dados eram para o treino e 30% para fins de teste.

Ao testar diferentes parâmetros nos estimadores, havia um risco de o conjunto de teste ter um melhor desempenho porque os parâmetros poderiam ser ajustados até que funcionasse de forma otimizada. Desta forma, os testes feitos em cima do conjunto de dados de teste poderiam 'viciar' o modelo e as métricas de avaliação já não desempenhasse de forma correta para dados mais genéricos. Para resolver esse problema, há uma outra parte do conjunto de dados que pode ser estendido como um conjunto de dados de validação. Prosseguindo o treinamento no conjunto de dados de treino, após decidir qual a avaliação será executada no conjunto de validação e o experimento estiver bem-sucedido, a avaliação final pode ser feita no conjunto de teste.

Validei minha análise chamando a função 'test_classifier' do arquivo 'tester.py' que faz validação cruzada com 'StratifiedShuffleSplit'. Como o conjunto de dados da Enron é pequeno, esse tipo de validação cruzado é útil pois cria múltiplos conjuntos de dados de um único conjunto na qual obtêm resultados mais precisos.

6. Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance.

Métrica de avaliação utilizada para identificar POI são precision e recall:

- precision = number of true positive / (number of true positive + number of false positive)

  É a proporção de classificações positivas corretamente previstas em relação ao total de observações previstas positivas. No meu caso é 0.5, significa que daqueles que classifiquei como pessoa de interesse, 50% eram efetivamente POI.

- recall = number of true positive / (number of true positive + number of false negative)

  É a proporção de classificações positivas corretamente prevista para todas as observações na classe. No meu caso é 0.4, significa que do total de pessoas de interesse que eram efetivamente, conseguimos classificar 40%.