Machine Learning on ENRON dataset

Jader Martins Camboim de Sá

29 de Maio de 2018

1 Introdução

Este projeto tem por objetivo investigar a base de dados da antiga empresa ENRON em busca de "Pessoas de Interesse", chamadas no dataset de "poi", que são aquelas que de alguma forma se envolveram com o escandalo de corrupção da empresa. Para isso foram dadas a classe '0' para pessoas inocentes e '1' para as que tiveram algum envolvimento, essas são chamadas de classes alvo. Além disso atributos financeiros e de emails foram fornecidos sobre cada pessoa. São no total 143 pessoas tendo cada uma delas 20 atributos, alguns não preenchidos. A investigação será feita por meio de machine learning em que o modelo preverá, dentro de uma certa metrica de erro, quais pessoas são inocentes e quais estão envolvidas no esquema de corrupção.

2 Estatísticas da Base

A base possui 143 registros com 20 atributos, dos 143 registros apenas 18 são pessoas de interesse e o restante, 125, não tiveram envolvimento. Na tabela a seguir estão presentes a quantidade de registros para cada atributo que contém dados missing.

Atributo	Qt NaN
bonus	62
deferral_payments	105
deferred_income	95
director_fees	127
exercised_stock_options	42
expenses	49
from_messages	57
from_poi_to_this_person	57
from_this_person_to_poi	57
loan_advances	140

Atributo	Qt NaN
long_term_incentive	78
other	52
poi	0
restricted_stock	34
restricted_stock_deferred	126
salary	49
shared_receipt_with_poi	57
to_messages	57
total_payments	20
total_stock_value	18

Alguns dos atributos tem diversos dados faltando, por isso possivelmente esses não serão usados. Também é importante entender a distribuição dos dados, como mostrado na figura 1, os valores para dados financeiros estão bem dispersos.

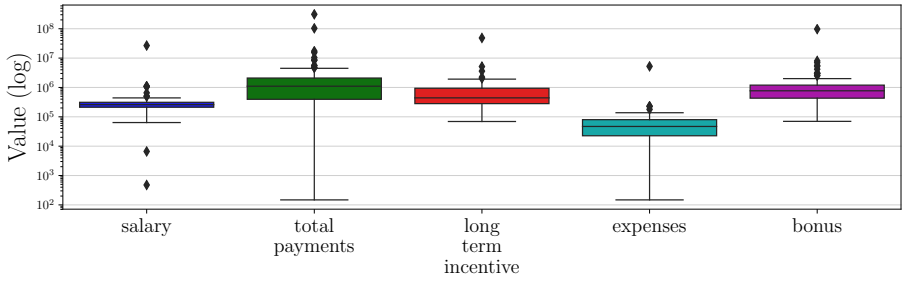

Figure 1: Financial Atributes

Dados muito maiores que 3 desvios padroes serão dropados e substituidos. Para as caracteristicas de emails a primeira vista não há nada interessante entre as razões de email de/para pois. Porem como visto no curso manteremos esses dados pois apresentam informações importantes.

Haviam registros inconsistentes que foram removidos ('TOTAL', 'THE TRAVEL AGENCY IN THE PARK', 'LOCKHART EUGENE E') esses estavam fora do contexto tabular.

3 Modelagem dos Dados

Muitos atributos não contem informações relevantes sobre o objetivo da predição por isso adiciona-los apenas adicionará ruído ao modelo atrapalhando a predição, como o objetivo é generalizar, utilizei do algoritmo SelectKBest para me escolher

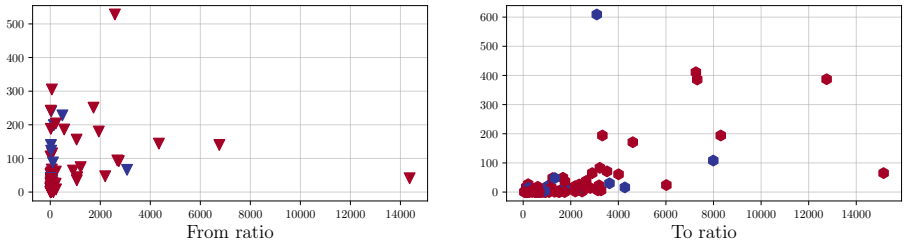

Figure 2: Email Atributes

os atributos visando que não ficasse enviesado aos meus dados de teste e que cada modelo pudesse trabalhar com uma quantidade diferente de dados com o GridSearchCV.

Também foi necessário fazer um *rescaling* dos dados pois alguns modelos assumem ou se beneficiam de uma distribuição normal, por isso dependendo do atributo foram aplicadas funções raiz, log ou normalização.

Table 2: Sem rescalonamento de variaveis

class	precision	recall	f1-score	support
0.0	0.89	0.87	0.88	38
1.0	0.17	0.20	0.18	5
avg/total	0.81	0.79	0.80	43

Table 3: Com rescalonamento de variaveis

class	precision	recall	f1-score	support
0.0	0.91	0.84	0.88	38
1.0	0.25	0.40	0.31	5
avg/total	0.84	0.79	0.81	43

Houve um ganho significativo com o rescaling. Por fim foram criados alguns atributos como 'total_stock_value'/'salary' para capturar alguma anormalidade no padrão de investimento dado conhecimento externo de um "poi", outro atributo criado foi o percentual de mensagens enviadas ou recebidas para/de um poi. Tais atributos como mostra na tabela a seguir não tiveram qualquer ganho para a regressão linear sem tuning de parametros, por isso foram ignorados pelo SKB.

Table 4: Com atributos criados

class	precision	recall	f1-score	support
0.0	0.91	0.82	0.86	38
1.0	0.22	0.40	0.29	5
avg/total	0.83	0.77	0.79	43

Table 5: Sem atributos criados

class	precision	recall	f1-score	support

class	precision	recall	f1-score	support
avg/total	0.84	0.79	0.81	43

4 Metodologia

Alguns algoritmos escolhidos conseguiram prever apenas os "não-poi", tendo em vista que dos 143 registros apenas 18 são "poi", por isso dei preferência aqueles que permitiam balancear o peso das classes, para minha surpresa a regressão logistica (junto ao GradientBoost) teve bons resultados sendo escolhida para o fine tuning final. Na etapa final (tuning) me atentei a fazer uma validação razoavel, de forma que não fossem tantos "folds" podendo demorar muito e nem poucos de forma que o modelo possa sofrer de overfitting, consultando diversos posts do forum e atraves de testes empiricos cheguei em um modelo balanceado de validação cruzada. No passo seguinte busquei fazer o tuning de parametros dos meus dados, esse passo é importante para que o modelo consiga capturar mais ou menos nuances nos dados, sendo que se realizado de forma correta pode gerar resultados muito melhores, capturando as peculiaridades do fenomeno. Dado que a regressão logistica tem poucos parametros (comparado a modelos de arvore por exemplo) decidi explorar uma vasta gama de valores no começo porém conforme ia modificando os dados vi que não seria necessário reduzindo meu conjunto de parametros a um menor.

Durante a fase de tuning de parametros é possivel que nosso modelo se ajuste muito bem para os casos de treino e teste, porém não queremos apenas isso, queremos que pra novos casos a serem avaliados nosso modelo também desempenhe bem, com base em alguma metrica de erro, surge então a validação, que é a analise da capacidade de generalização do algoritmo, ela irá avaliar se nosso modelo não sofreu de overfitting, tendo boa performance apenas para os dados amostrados, mas sim se consegue a partir de novos dados manter um bom desempenho. Aqui usando o GridSearchCV realizamos a busca por parametros que mais generalizam. Coloquei a principio como métrica o f1-score, porém vi que o mesmo não tinha bons resultados e voltei ao default. Aqui o SelectKBest, através do GridSearchCV escolheu todas as features menos a criada 'total_stock_value'/'salary'.

5 Conclusão

Obtive os seguintes resultados para a avaliação do GradientBoostClassifier e do LogisticRegression utilizando o GridSearchCV para otimizar os parametros:

Table 6: GradientBoostClassifier

class	precision	recall	f1-score	support
0.0	0.90	0.97	0.94	38
1.0	0.50	0.20	0.29	5
avg/total	0.86	0.88	0.86	43

Table 7: LogisticRegression

class	precision	recall	f1-score	support
0.0	0.97	0.84	0.90	38
1.0	0.40	0.80	0.53	5
avg/total	0.90	0.84	0.86	43

A revocação (recall) é a fração dos registros positivos que foram classificados corretamente, já a precisão é a fração de classificados corretamente como positivos sobre todos classificados como positivos, o F1-score combina a media harmonica destas duas metricas para dar uma visão abrangente do desempenho.

Dada uma amostra desbalanceada, maioria de registros de uma unica classe, é importante que usemos esta métrica, pois se chutassemos que todo valor é da classe mais presente acertariamos na maioria dos casos, tendo uma boa acurácia, (acertos sobre total de chutes), porém como nos dados aqui avaliados é mais importante conhecer os registros mais raros, o F1-score garante uma boa avaliação do modelo mesmo em casos desbalanceados.

Como mostrado acima a regressão logistica obteve um score melhor para o recall ela foi exportada e por fim avaliada usando o script teste.py, este faz uma validação cruzada usando 1000 folds, que gerou os seguintes resultados, cumprindo com o objetivo deste projeto:

Metrica	Resultado
Accuracy	0.79767
Precision	0.33863
Recall	0.54300
F1	0.41713