PROJETO: **IDENTIFICAR FRAUDE NO EMAIL DA ENRON**

Este projeto pertence ao Nanodegree Fundamentos de Data Science II da Udacity

SOBRE

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 visa criar um modelo preditivo para determinar se um funcionário é ou não um funcionário de interesse (POI). Um POI é um funcionário que participou do escândalo da empresa Enron.

Utilizar algoritmos de Machine Learning é muito útil na criação desse modelo pois disponibilizamos de um conjunto de dados cuja relação entre as variáveis e aquilo que desejamos encontrar não é clara para nós, humanos.

TRATAMENTO DE DADOS

- Investigou-se a presença de funcionários cujos nomes possuiam menos de 2 palavras ou mais de 4 palavras. Encontrou-se os seguintes nomes: TOTAL e THE TRAVEL AGENCY IN THE PARK. Deletou-se ambos as linhas de dados referentes a esses nomes, pois não representavam funcionários;

- Deletou-se as linhas de dados onde mais de 85% dos dados, ignorando a *feature* poi, eram nulos;

- Verificou-se a quantidade de dados negativos em cada coluna. Para as *features* deferred_income e restricted_stock_deferred, espera-se dados negativos. Entretanto, encontrou-se um valor negativo nas *features* restricted_stock, deferral_payments, total_stock_value. Então, analisou-se os funcionários que apresentavam esses valores negativos e, ao comparar com o arquivo enron61702insiderpay.pdf, notou-se que os funcionários BHATNAGAR SANJAY e BELFER ROBERT possuiam seus dados trocados em *features* erradas. Corrigiu-se seus valores errados;

- Trocou-se todos os valores NaN por zeros para que toda *feature* possuísse apenas um tipo de dado. Assim, evita-se que o classificador encontre uma relação entre valores não informados e a classe desejada;

- Trocou-se todos os valores negativos por valores absolutos, pois em etapas posteriores, a função *SelectKBest* será aplicada com o parâmetro *chi²* para selecionar as *features* que serão usadas para treinar o classificador. O parâmetro *chi²* pontua a relação de cada *feature* não-negativa com a classe POI. Apesar da função *MinMaxScaler*, aplicada também em estapas posteriores, reescalar os valores entre 0 e 1, preferi tornar os valores negativos em valores absolutos desde essa etapa para caso eu não desejasse aplicar o reescalador e mesmo assim desejasse aplicar o selecionador *KBest* com o parâmetro *chi²*

CRIAÇÃO DE NOVAS FEATURES

- Criou-se um total de 7 novas *features* de proporção. *Features* de proporção podem ser mais úteis que valores absolutos uma vez que, por exemplo, um funcionário pode receber um salário semelhante ao da maioria, entretanto, ao pegar a proporção de seu salário pelo total de pagamentos, verifica-se que o salário, na verdade, é discrepante em relação à proporção da maioria.

Quadro 1 – Novas *features* de proporção com seus respectivos numeradores e denominadores.

NOVA FEATURE	NUMERADOR	DENOMINADOR
bonus_ratio	bonus	total_payments
salary_ratio	salary	total_payments
expenses_ratio	expenses	total_payments
from_poi_to_this_person_ratio	from_poi_to_this_person	from_messages
from_this_person_to_poi_ratio	from_this_person_to_poi	to_messages
exercised_stock_options_ratio	exercised_stock_options	total_stock_value

restricted_stock_ratio	restricted_stock	total_stock_value

- Criou-se um total de 10 novas *features* quadráticas. O propósito foi criar uma polarização dos valores dessas *features*, uma vez que, dessa forma, a distância entre os pontos aumentaria.

Quadro 2 – Novas *features* quadráticas

NOVA FEATURE	FEATURE USADA
squared_salary	salary
squared_bonus	bonus
squared_expenses	expenses
squared_salary_ratio	salary_ratio
squared_bonus_ratio	bonus_ratio
squared_expenses_ratio	expenses_ratio
squared_from_poi_to_this_person_ratio	from_poi_to_this_person_ratio
squared_from_this_person_to_poi_ratio	from_this_person_to_poi_ratio
squared_exercised_stock_options_ratio	exercised_stock_options_ratio
squared_restricted_stock_ratio	restricted_stock_ratio

APLICAÇÃO DE CLASSIFICADORES "NÃO-TUNADOS"

1• Classificadores escolhidos: *Naive Bayes, Support Vector Machines, K Nearest Neighbors, Decision Tree*.

2• Dividiu-se o conjunto de dados da seguinte forma:

- 70% como conjunto para o treinamento;
- 30% como conjunto para teste.

3• Aplicou-se o reescalador *MaxMinScaler* em todas as *features*;

4• Nesta etapa, não se fez a seleção de *features*, a ideia foi criar novas listas de *features* em ordem de importância, para cada classificador que possuisse um dos atributos: `feature_importances_` ou `coef_`;

5• Treinou-se os classificadores com o conjunto para o treinamento

6• Aplicou-se a função *test_classifier* do arquivo tester.py em cada um dos 4 classificadores a fim de avaliar as métricas *Accuracy*, *Precision*, *Recall* e F1:

Quadro 3 – Métricas avaliadas, suas descrições e fórmulas.

Accuracy	Representa a proporção de itens preditos corretamente.	$\frac{VP + VN}{Total\ de\ Predições}$
Precision	Representa a proporção de itens preditos verdadeiros que são, de fato, verdadeiros.	$\frac{VP}{VP + FP}$
Recall	Representa a proporção de itens preditos verdadeiros recuperados do conjunto de dados.	$\frac{VP}{VP + FN}$
F1	Representa a média harmônica das métricas Precision e Recall.	$\frac{2 * VP}{2 * VP + FP + FN}$

Quadro 4 – Resultados obtidos na execução de cada classificador.

Classificadores	Métricas			
	Accuracy	Precision	Recall	F1
NB	0,85729	0,50138	0,18200	0,26706
SVM	0,31693	0,10274	0,48900	0,16981
KNN	0,86121	0,53857	0,19900	0,29062
DT	0,78786	0,25431	0,25100	0,25264

NB: Naive Bayes, **SVM**: Support Vector Machines, **KNN**: K Nearest Neighbors, **DT**: Decision Tree.

7• Escolheu-se o classificador baseado na métrica F1. Quando mais alta essa pontuação é, menos falsos positivos e falsos negativos o classificador obteve no teste. Portanto, o classificador escolhido foi o *K Nearest Neighbors* que obteve não apenas a métrica F1 superior, mas também a maior *Accuracy* e *Precision*.

"TUNANDO" O CLASSIFICADOR ESCOLHIDO

"Tunar" um classificador significa identificar e selecionar os melhores parâmetros de um classificador a fim de melhorar a sua performance e as suas métricas.

1• Neste modelo, empregou-se a validação cruzada. Uma abordagem que utiliza todo o conjunto de dados para treinar e testar o modelo. Esse método reparte o conjunto de dados em k repartições e, então, aplica o modelo k vezes, utilizando sempre uma repartição diferente como conjunto de teste e as outas k-1 repartições como conjunto de treino;

2• Aplicou-se o mesmo reescalador *MinMaxScaler* em todas as *features*. É necessário aplicar esse reescalador, visto que, o algoritmo em questão lida diretamente com as distâncias entre pontos, portanto, distâncias absolutas podem levar um algoritmo à uma aprendizagem errônea.

3• Aplicou-se o selecionador de *features KBest*;

4• Aplicou-se o *GridSearchCV* sobre o classificador *K Nearest Neighbors*, realizando um *StratifiedKFold* (validação cruzada onde a proporção de instâncias positivas do conjunto de dados original é mantida em todas as repartições) com 3 repartições (*folds*) e buscou os melhores parâmetros que melhorassem a métrica *Recall* do classificador;

5• Executou-se um *loop* nos passos 3 e 4 para diferentes quantidades de *features* selecionadas (k = [3, 4, ..., 36]). Verificou-se que as métricas se repetiam para todos os valores de k, com exceção de k=3 e k=7. Ao final, escolheu-se k = 4, uma vez que esse era o menor valor de k para o maior valor de F1. Além disso, ambas as métricas Precision e Recall foram superiores a 0.3.

Quadro 5 – Valores das métricas obtidas na aplicação do classificador para diferentes quantidades de *features* escolhidas (k).

Nº de *features* (k)	Accuracy	Precision	Recall	F1
3	0,85371	0,48240	0,32900	0,39120
4	0,87207	0,59812	0,31850	0,41566
5, 6	0,87207	0,59812	0,31850	0,41566
7	0,86121	0,53857	0,19900	0,29062
8-36	0,87207	0,59812	0,31850	0,41566

Quadro 6 – Features escolhidas pelo SelectKBest. K = 4.

Features	Score
loan_advances	6,5268
squared_bonus_ratio	6,3257
exercised_stock_options	6,0992
deferred_income	5,5203

Verifica-se a importância da criação de novas *features* ao fazer-se a comparação das métricas de um mesmo classificador (com os mesmos parâmetros e mesmo número k de *features* escolhidas) nas seguintes condições:

1• Utilizando as *features* originais + as *features* criadas;

2• Utilizando apenas as *features* originais.

Para a condição 1, as métricas são as do Quadro 5 quando k = 4. Entretanto, para a condição 2, as métricas são as seguintes:

Quadro 7 – Valores das métricas obtidas.

Nº de *features* (k)	Accuracy	Precision	Recall	F1
4	0,85007	0,42710	0,14500	0,21650

Nota-se a diminuição de todas as métricas:

• **Accuracy:** Redução de 0,022;

• **Precision:** Redução de 0,17102;

• **Recall:** Redução de 0,1735 (Mais de 50%);

• **F1:** Redução de 0,19916.

VALIDAÇÃO DO CLASSIFICADOR

Validação é um grupo de processos que nos permite ter o controle da performance do modelo em conjuntos de dados diferentes daquele que foi usado para o seu treino. Ao dividirmos o conjunto de dados em conjuntos de treino e teste, podemos avaliar, com as métricas desejadas: a performance real do modelo treinado e a presença de *overfitting* – evento que acontece quando o modelo se ajusta muito bem ao conjunto de dados de treinamento, entretanto é ineficaz para prever novos resultados.

O *GridSearchCV* retornou os melhores parâmetros para o classificador *K Nearest Neighbors*.

A performance desse modelo – e, também, dos modelos não tunados - foi avaliada da seguinte maneira:

1• Embaralhou-se o conjunto de dados e separou-se 10% como conjunto de teste e 90% como conjunto de treinamento. A proporção das classes em ambos os conjuntos se mantiveram.

2• Treinou-se o modelo com o conjunto de treinamento e realizou-se uma predição com o conjunto de teste. Armazenou-se a quantidade de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN).

3• Repetiu-se as etapas 1 e 1 outras 999 vezes. Somando-se sempre a quantidade de VP, VN, FP, FN.

4• Por fim, utilizou-se as fórmulas do Quadro 4 para a obtenção dos valores das métricas.

Apesar das métricas estarem acima do parâmetro exigido, estima-se que essas métricas sejam superestimadas, uma vez que o grupo usado para decidir os melhores parâmetros, foi usado no treinamento do modelo. Portanto, o classificador deverá ter uma performance abaixo do esperado em dados não vistos.