Udacity Nanodegree Fundamentos de Data Science 2

Projeto 4 (Machine Learning)

Autor: Márcio Souza de Oliveira

Questionário de avaliação dos resultados do projeto

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? [relevant rubric items: "data exploration", "outlier investigation"]

Resposta: O objetivo desse projeto é usar técnicas de Machine Learning para treinar algoritmos (classificadores) com dados reais de fraudes que ocorreram na empresa americana ENRON para podermos prever o envolvimento de possíveis suspeitos em casos similares em outras empresas.

O conjunto de dados possui informações financeiras de cada funcionário da ENRON como salário, bônus, valores em ações, pagamentos realizados e recebidos, quantidades de e-mails enviados e recebidos, conteúdo dos e-mails, etc. Além disso, temos a informação extra dos funcionários que foram classificados como "POI" (*person of interest*), ou seja, daqueles que realmente foram acusados de estarem envolvidos com as fraudes, que é uma informação relevante para treinar os algoritmos.

No total temos 146 registros de funcionários com 21 atributos cada onde todos foram todos utilizados, com exceção do atributo *"email_address"* que era o único não numérico e que não era relevante para os testes. No projeto também foram disponibilizados os conteúdos de diversos e-mails que talvez pudessem ser relevantes para criar novos atributos, mas não foram utilizados. Desse conjunto percebi que apenas 18 registros foram marcados como "POI" e temos bastantes atributos com dados disponíveis, como é o caso dos "*loan_advances"* que só aparece em 3 registros ou *"restricted_stock_deffered"* que só aparece em 17 registros

Três registros foram removidos: o primeiro era um totalizador (*TOTAL*) criado na planilha original e foi importado incorretamente como um dado, portanto destacava-se como *"outlier"*, o segundo (*LOCKHART EUGENE E*) não possuía dados em nenhum atributo e o último (*THE TRAVEL AGENCY IN THE PARK*) não parecia ser um funcionário.

Por fim, também reparei que 2 registros foram importados incorretamente da planilha original (os dados estavam deslocados de suas colunas corretas) e fiz o acerto manual dos mesmos (*BELFER ROBERT* e *BHATNAGAR SANJAY*).

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. [relevant rubric items: "create new features", "intelligently select features", "properly scale features"]

Resposta: Os atributos escolhidos foram selecionados usando o algoritmo *"SelectKBest"*. Para chegar na quantidade de atributos ideais, usei uma técnica de força bruta que começava com apenas 1 atributo para ser selecionado pelo *"SelectKBest"* e em seguida testava os classificadores com apenas esse atributo, em seguida utilizei 2 atributos selecionados pelo *"SelectKBest"* e repeti os testes. E isso foi se repetindo até chegar no total dos 21 atributos possíveis (incluindo os 2 novos, explicados mais abaixo) para serem testados com os classificadores. Após análise de todos

resultados, o número ideal de atributos para o melhor classificador se resumiu a 11 com os seguintes scores do *"SelectKBest"*:

- Valor total em ações (22.51)
- Opções de ações exercidas (22.35)
- Bônus (20.79)
- Salário (18.29)
- Receita diferida (11.42)
- Incentivo de longo prazo (9.92)
- Pagamentos totais (9.28)
- Ações restritas (8.83)
- Recibos compartilhados com POI (8.59)
- Adiantamentos de empréstimos (7.18)
- % dos e-mails totais destinados para um POI (16.41)

Não houve necessidade de escalonamento dos atributos uma vez que essa técnica não afeta os algoritmos selecionados para os testes.

Dois atributos foram computados e adicionados ao conjunto original:

- % dos e-mails totais destinados para um POI
- % dos e-mails totais recebidos de um POI

Como e-mails são um meio comum de comunicação em empresas, a quantidade total poderia não ser relevante para identificar um suspeito, então optei por trabalhar apenas com o percentual das mensagens que foram trocadas com um POI para tentar identificar uma possível afinidade do suspeito nas fraudes. Para funcionários que já estavam marcados como POI não computei esses dados para não enviesar o classificador.

O % dos e-mails totais destinados para um POI acabou se mostrando como uma boa escolha ficando em décimo primeiro lugar dentre os atributos selecionados para o melhor classificador, como listado acima. Curiosamente, o atributo de adiantamentos de empréstimos (*"loan advances"*) que só foi informado em 3 registro, como já comentado na resposta da pergunta 1, ficou em décimo lugar.

OBS: Mais detalhes sobre os relatórios gerados com os testes dos classificadores estão nos apêndices desse documento.

3) What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

Resposta: O classificador selecionado foi o *"Decision Tree"* com os seguintes parâmetros ajustados:

- *criterion='entropy'*
- *min_samples_split=15*

Também foram testados *"Gaussian Naive Bayes"* e *"Logistic Regression"*.

Quando usei um dos atributos novos, o melhor resultado obtido foram com 11 no total selecionados pelo *"SelectKBest"*:

Classificador	Acurácia (*"Accuracy"*)	Precisão (*"Precision"*)	Abrangência (*"Recall"*)
Gaussian Naive Bayes (padrão)	0.8223	0.3258	0.3115
DecisionTree (ajustado)	**0.9299**	**0.7881**	**0.6490**
LogisticRegression (ajustado)	0.5996	0.1865	0.5960

Fazendo a mesma análise usando apenas os atributos originais, o melhor resultado obtido foi com apenas 5 atributos selecionados pelo *"SelectKBest"* e o Gaussian Naive Bayes mostrou melhor performance:

Classificador	Acurácia (*"Accuracy"*)	Precisão (*"Precision"*)	Abrangência (*"Recall"*)
Gaussian Naive Bayes (padrão)	**0.8470**	**0.4541**	**0.3510**
DecisionTree (ajustado)	0.8201	0.3401	0.2755
LogisticRegression (ajustado)	0.6516	0.2093	0.5180

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). [relevant rubric items: "discuss parameter tuning", "tune the algorithm"]

Resposta: Ajustar os parâmetros de um classificador significa encontrar uma combinação de parâmetros que resultará na melhor performance do classificador de acordo com as métricas que estamos avaliando. Quando realizei testes iniciais com os parâmetros padrão de cada classificador, os resultados para algumas métricas, como precisão e abrangência, ficaram bem baixos, sequer atingindo o valor mínimo de 0.3 exigidos nesse projeto para essas mesmas métricas. Portanto é algo muito importante fazer o ajuste dos parâmetros.

Os ajustes foram feitos tanto manualmente quanto automaticamente. A parte manual é a escolha dos atributos dos dados e a quantidade dos mesmos que serão utilizados para o ajuste dos parâmetros de cada classificador. Isso também é essencial para algoritmos como o Gaussian Naive Bayes que não possui parâmetros para serem ajustados, então selecionar bons atributos é essencial para atingir o melhor desempenho com ele.

Como citado na resposta da pergunta 1, utilizei um método de força-bruta, onde eu realizei testes com todas as quantidades possíveis de atributos ranqueadas pelo *"SelectKBest"* e, para cada quantidade de atributos selecionados, foi feito o ajuste automático dos parâmetros de cada classificador (pelo menos dos que possuem parâmetros) usando o *"GridSearchCV"* que resultou em excelentes combinações de parâmetros.

Para o classificador *"Decision Tree"* foram usadas as seguintes combinações de valores dos seguintes parâmetros:

- ***criterion*** *['gini', 'entropy'] – função para medir a qualidade de uma divisão, onde "gini" corresponde à impureza Gini e "entropy" corresponde ao ganho de informação.*
- ***min_samples_split*** *[10,15,20,25] – a quantidade mínima de amostras requeridas para dividir um nó interno*

Já para o classificador *"Logistic Regression"* foram usadas as seguintes combinações de valores dos seguinte parâmetros:

- ***C*** *[0.05, 0.5, 1, 10, 10**2, 10**3, 10**5, 10**10, 10**15] – Controla o limite de troca entre uma fronteira de decisão suave e outra que classifica todos os pontos de treinamento corretamente.*
- ***tol*** *[10**-1, 10**-2, 10**-4, 10**-5, 10**-6, 10**-10, 10**-15] – tolerância para o critério de interrupção*

5) What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric items: "discuss validation", "validation strategy"]

Resposta: Validação significa separar parte do seu conjunto de dados para treinamento do classificador e outra para avaliar a performance do classificador. O grande desafio é fazer uma boa divisão para que o classificador não fique ou muito enviesado ("bias") – ignora os dados independente do treinamento – ou com muita variância ("variance") – fica extremamente perceptivo aos dados replicando apenas o que já viu.

A validação foi realizada usando o *"train_test_split"* do *sklearn* onde 30% dos dados foram reservados para testes e 70% para treinamento. Após a obtenção dos melhores parâmetros do classificador com o *"GridSearchCV"* e a quantidade de atributos selecionadas com o *"SelectKBest"*, os resultados foram avaliados com validação-cruzada usando o algoritmo "*StratifiedShuffleSplit"* do *sklearn* com 1000 "folds", da mesma forma realizada no método *"test_classifier"* do arquivo *"tester.py"* disponibilizado no projeto.

Acredito que essa escolha foi apropriada porque o "*StratifiedShuffleSplit"* é um validador-cruzado que testa diversas combinações de dados mantendo um percentual similar de POI's do conjunto completo em cada teste, então deve gerar métricas mais consistentes, e aproveitando para já avaliar quais classificadores iriam atingir as métricas mínimas requeridas no projeto (0.3 para precisão e abrangência) usando esse mesmo algoritmos e quantidade de "folds".

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. [relevant rubric item: "usage of evaluation metrics"]

Resposta: As métricas escolhidas foram precisão ("precision"), abrangência ("recall") e acurácia ("accuracy"). Em média, o classificador final atinge:

- Acurácia: 0.92
- Precisão: 0.75
- Abrangência: 0.64

O que significa de modo geral que o classificador tem ótima performance para identificar a quantidade POI's no conjunto, mas com maior foco em afirmar com confiança quais os funcionários que são de fato um POI, mesmo com o efeito colateral de talvez deixar de registrar alguns POI's legítimos (falsos negativos). Se fosse o inverso (abrangência maior do que a precisão), então o foco seria de tentar identificar os POI's no conjunto o máximo possível, mesmo que alguns inocentes fossem eventualmente classificados como POI (falsos positivos).

Talvez para um caso de investigação de fraudes, essa segunda opção fosse mais interessante, pois é melhor encontrar os culpados primeiro e depois tentar "limpar o nome" dos inocentes do que ter a possibilidade de alguns acusados não serem identificados. Infelizmente os resultados que deram boas abrangências não tiveram boa performance nas outras métricas, então optei pelo resultado atual que estava mais balanceado.

Apêndice:

Em anexo ao projeto estou encaminhando relatórios de 4 execuções para escolha dos melhores atributos e classificadores:

- **classifier_evaluation_NO_FILTERED_NEW_FEATURES.txt** – Lista o resultado de todos os classificadores para cada quantidade possível de atributos selecionada pelo *"SelectKBest"* (incluindo os 2 novos atributos computados)
- **classifier_evaluation_NO_FILTERED_OLD_FEATURES.txt** – Mesma coisa que o anterior, mas considerando apenas os atributos originais do conjunto de dados
- **classifier_evaluation_FILTERED_NEW_FEATURES.txt** – Mesma coisa que o primeiro, mas lista apenas os classificadores que obtiveram métricas de *"precision"* e *"recall"* com um valor mínimo de 0.3
- **classifier_evaluation_ FILTERED_OLD_FEATURES.txt** – Mesma coisa que o anterior, mas considerando apenas os atributos originais do conjunto de dados

OBS: O programa "poi_id.py" permite que os relatórios acima sejam gerados com base em escolhas durante a sua execução:

- Usar ou não os novos atributos computados
- Definir o limite mínimo (*threshold*) de *“precision”* e *“recall”* que classificadores devem atingir para serem listados no relatório
- Definir a faixa de valor que deverá ser utilizada para o processo de força-bruta de seleção de atributos como o “SelectKBest”