Udacity Nanodegree Fundamentos de Data Science 2
Projeto 4 (Introdução a Machine Learning)

Autor: Diego Serra

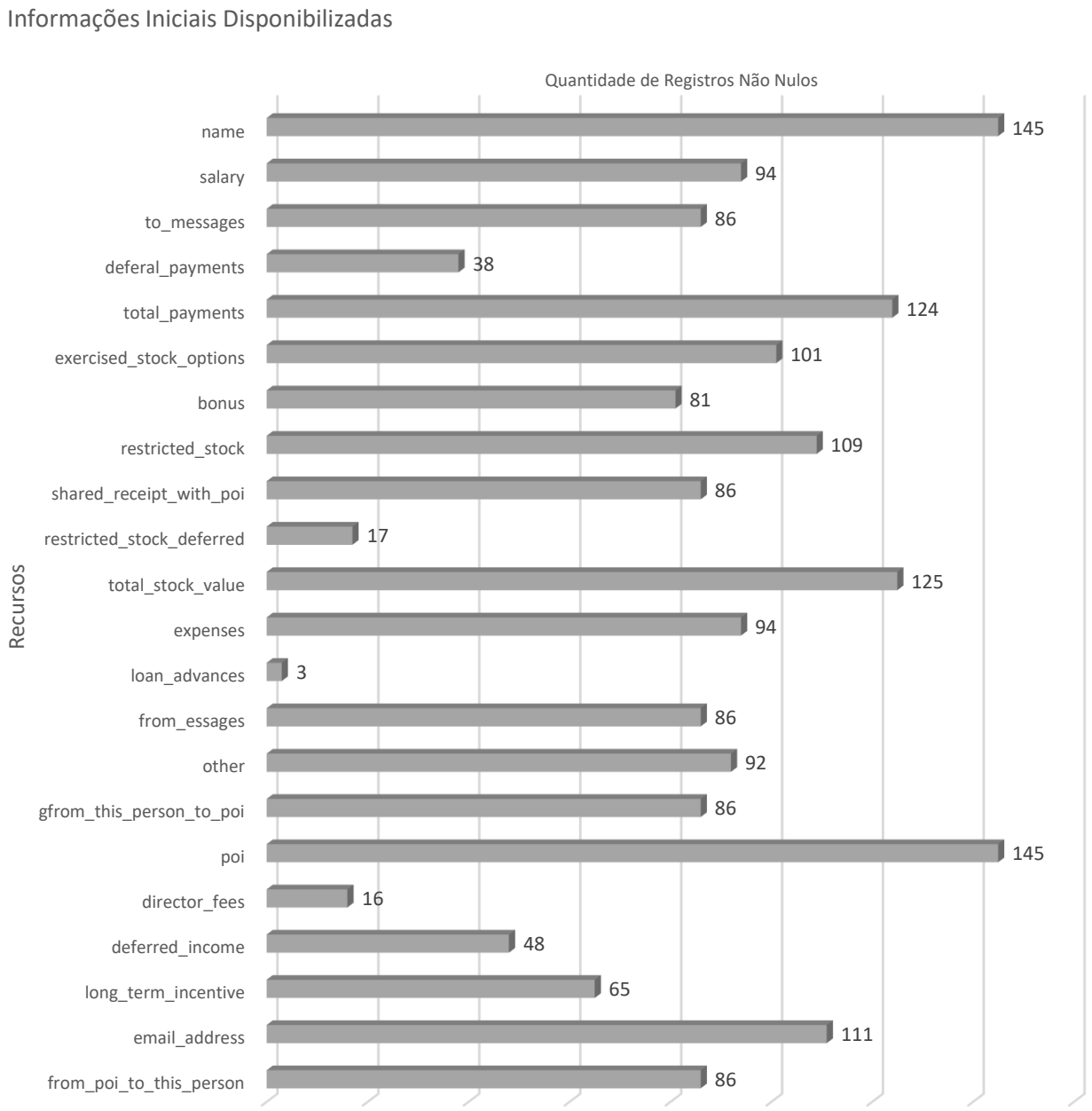
1. **Resuma para nós o objetivo deste projeto e como o aprendizado de máquina é útil na tentativa de realizá-lo. Como parte de sua resposta, dê um panorama sobre o conjunto de dados e como ele pode ser usado para responder à pergunta do projeto. Houve algum valor discrepante nos dados quando você os obteve e como você lidou com eles? [rubrica relevante de rubricas: "exploração de dados", "investigação outlier"]**

1.1 – Panorama do Conjunto de dados.

O projeto tem por objetivo usar técnicas de Aprendizado de Máquina (Machine Learning) para treinar algoritmos com dados reais da empresa americana Enron Corporation foi uma companhia de energia americana, localizada em Houston, Texas. Empregava cerca de 21 000 pessoas, tendo sido uma das empresas líderes no mundo em distribuição de energia (eletricidade, gás natural) e comunicações, mas decretou falência. Seu faturamento atingia 101 bilhões de dólares em 2000, pouco antes do escândalo financeiro que ocasionou sua falência. Em posse de dados disponibilizados pela Udacity (dados públicos), iremos realizar as técnicas de aprendizado de máquina e temos o intuito de encontrar um algoritmo que possa prever o envolvimento de possíveis suspeitos tanto neste caso quanto em casos similares em outras empresas.

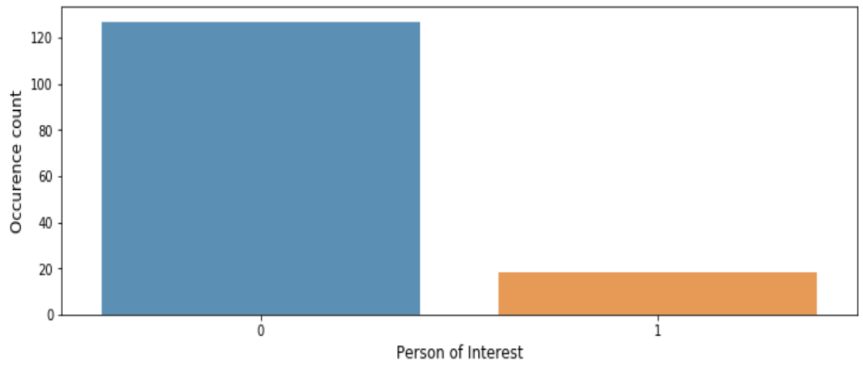
Usando o conjunto de dados disponibilizado, com informações financeiras (salário, bônus, valores em ações, pagamentos realizados, quantidade de e-mails enviados e recebidos, etc), vamos treinar nosso algoritmo de aprendizado de máquina para detectar possibilidades de fraude. Neste caso, conhecemos as pessoas de interesse (POIs) em nosso conjunto de dados e por isso poderemos usar algoritmos de aprendizagem supervisionados na construção de um identificador de POIs e separa-los dos não POIs.

No conjunto de dados foram identificados 146 registros de funcionários, e passamos a trabalhar inicialmente com 21 variáveis:



No conjunto de dados que apenas 18 registros foram marcados como POIs:

```
O conjunto de dados contêm: 18 POIs.
```



1.2 – Panorama do Conjunto de dados.

No decorrer do trabalho, foram descartadas as colunas “loan_advances”, “diretor_fees” devido o grande número de registros ausentes e a coluna “email_address”, por não apresentar relevância para a pesquisa.

Foram removidos os outliers sendo o primeiro o totalizador do dicionário (TOTAL) que consta na planilha inicial e no CSV inicial, na sequencia o registro de “LOCKHART EUGENE E” que não está classificado como POI e não possuía atributos, mesmo em pesquisa na internet, não aparece como um dos envolvidos.

2. **Quais recursos você usou no seu identificador de POI e que processo de seleção você usou para selecioná-los? Você teve que fazer algum escalonamento? Por que ou por que não? Como parte da tarefa, você deve tentar projetar seu próprio recurso que não vem pronto no conjunto de dados - explique qual recurso você tentou fazer e a lógica por trás dele. (Você não precisa necessariamente usá-lo na análise final, apenas o engenheie e teste.) Na sua etapa de seleção de recursos, se você usou um algoritmo como uma árvore de decisão, forneça também as importâncias de recursos dos recursos que você usa, e se você usou uma função de seleção de recursos automatizada como o SelectKBest, por favor, informe as pontuações dos recursos e as razões para a sua escolha de valores de parâmetros. [rubricas relevantes: “criar novos recursos”, “selecionar recursos de maneira inteligente”, “dimensionar recursos corretamente”]**

No início da análise dos dados, foram incluídas as colunas: total_be que representa o somatório de “bonus” e “exercised_stock_options”, assim como a coluna “total_millions” que representa o somatório de “total_pyments” e “total_stock_value”. Ambas as variáveis foram criadas para facilitar a busca por informações financeiras e facilitar o rastreio de montantes elevados. Por seguinte, conforme exemplos aplicados em sala, foram criadas duas variáveis que nos retornam a proporção de mensagens recebidas e enviadas pelos POIs, respectivamente “fraction_to_poi” e “fraction_from_poi”.

Em seguida, foram comparados o desempenho dos algoritmos, mantendo apenas os recursos originais, (já excluídos os mencionados acima) e compramos o desempenho dos algoritmos em análise e chegamos aos seguintes resultados:

Resultados sem a Inclusão de novos recursos			
Algoritmo	**Accuracy**	**Precision**	**Recall**
GaussianNB	0.5	0.19230769230769232	0.8333333333333334
DecisionTree	1.0	1.0	1.0
Kneighbors	0.8636363636363636	0.5	0.16666666666666666

Incluindo os recursos os novos recursos podemos identificar uma melhora significativa , em uma simulação considerando todos os recursos, em especial para o resultado do algoritmo Kneighbors.

Resultados com a Inclusão de novos recursos			
Algoritmo	**Accuracy**	**Precision**	**Recall**
GaussianNB	0.6590909090909091	0.263157894736842	0.8333333333333334
DecisionTree	0.16666666666666666	0.1111111111111111	0.7045454545454546
Kneighbors	0.8863636363636364	0.6666666666666666	0.3333333333333333

Em um segundo passo foi realizada uma verificação individual das variáveis(colunas), foram selecionadas quatro listas, que podem fornecer informações sobre o comportamento de um POI, e por consequência ser interessante para os classificadores, considerando as características que mostraram uma maior diferença global entre as estatísticas POI e não-POI foram escolhidas. Uma média relevante observada foi a de POIs possuem participação mais positiva em “exercised_stock_options”.

A quantidade de total de emails não se apresentou relevante para identificar POIs e não POIs devido a proximidade de seus envios e recebimentos. Para isso foram criadas as variáveis “fraction_to_poi” e “fraction_from_poi” que é o percentual de mensagens enviadas e recebidas de um POI, para buscar as afinidades dos suspeitos nas fraudes.

Foi feita a opção de 4 listas de teste com apenas 3 variáveis, para evitar distorções provocados por outras variáveis, sendo assim a “fraction_to_poi”, demonstrou ser uma boa escolha, visto que, fez parte do em conjunto de variáveis escolhidas: ['poi', 'bonus', 'exercised_stock_options'], que apresentaram um resultado positivo com o algoritmo Kneighbors conforme abaixo:

KNeighbors:

'Recall': 0.6,

'Precision': 0.75,

'Accuracy': 0.9210526315789473.

Com o objetivo de melhorar a eficiência do algoritmo escolhido, foi utilizado o GridSearchCV, testado usando uma validação cruzada com 5 folds (cv=5). Em seguida, solicitamos os melhores parâmetros com "clf_params.best_estimator_", e o aplicamos. O resultado da sugestão do GridSearchCV foi o mesmo apresentado acima, e por isso decidido manter as características escolhidas:

```
clf_params.best_estimator_
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=1, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
clf = KNeighborsClassifier(algorithm='auto', leaf_size=1, metric='minkowski',
                           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                           weights='uniform')
clf.fit(features_train,labels_train)
pred = clf.predict(features_test)

print {'Accuracy': accuracy_score(labels_test,pred),'Precision': precision_score(labels_test,pred),
       'Recall': recall_score(labels_test,pred)}
```

```
{'Recall': 0.6, 'Precision': 0.75, 'Accuracy': 0.9210526315789473}
```

3. **Qual algoritmo você acabou usando? Que outro (s) você tentou? Como o desempenho do modelo diferiu entre os algoritmos? [rubrica de rubrica relevante: "escolha um algoritmo"]**

O classificador selecionado foi o KNeighbors e as variáveis finais, com base na análise e teste de recurso, são:

1. 'bonus'
2. 'exercised_stock_options
3. 'fraction_to_poi'

Mesmo sem ajuste, com as configurações padrão, o classificador KNeighbors, apresentou um melhor resultado se comparado a Árvore de decisão.

Foram testados 3 algoritmos: GaussianNB, DecisionTreeClassifier, KNeighborsClassifier e os resultados para as variáveis escolhidas foram:

	Accuracy	Precision	Recall
GaussianNB	0.7368421052631579	0.14285714285714285	0.2
DecisionTree	0.7105263157894737	0.2	0.4
KNeighbors	0.9210526315789473	0.75	0.6

As variáveis escolhidas, apresentaram um bom desempenho com esse classificador Kbeigbors. E as variáveis foram reduzidas para apenas três com o objetivo de minimizar as distorções que podem acontecer com listas mais longas de variáveis.

4. **O que significa ajustar os parâmetros de um algoritmo e o que pode acontecer se você não fizer isso bem? Como você ajustou os parâmetros do seu algoritmo particular? Quais parâmetros você ajustou? (Alguns algoritmos não possuem parâmetros que você precisa ajustar - se este for o caso daquele que você escolheu, identifique e explique brevemente como você teria feito isso para o modelo que não foi a sua escolha final ou um modelo diferente que faz utilizar o ajuste de parâmetros, por exemplo, um classificador da árvore de decisão). [rubricas relevantes: "discutir o ajuste dos parâmetros", "ajustar o algoritmo"]**

Ajuste de parâmetro significa encontrar uma combinação de parâmetros que resultará na melhor performance do classificado de acordo com as métricas que estamos avaliando. Foram realizados testes com o parâmetro padrão de cada de cada classificador, e os resultados ficaram abaixo do classificador escolhido.

Os ajustes foram feitos manualmente, como o Gaussian Naive Bayes não possui parâmetros para serem ajustados. Então, testamos o dois restantes DecisionTreeClassifier e KNeighbors Classifier.

5. **O que é validação e qual é um erro clássico que você pode cometer se errar? Como você validou sua análise? [rubricas relevantes: "discutir validação", "estratégia de validação"]**

Validação se traduz na separação de parte do conjunto de dados para treinamento do classificador, uma vez que, usar apenas os mesmos dados podem gerar resultados excessivos. Foi utilizado o StratifiedShuffleSplit onde foi configurado para teste(test_size) o valor de 0.30 e o gerador de números aleatórios (random_state) o valor de 42, para dividir nossos dados em dados de treinamento e teste. Assim pode-se garantir que nossas classes sejam alocadas pela mesma proporção definida para treinamento/teste e que cada ponto de dados da classe seja selecionado aleatoriamente.

o StratifiedShuffleSplit é um validador-cruzado que testa diversas combinações de dados mantendo um percentual siilar dePOI's então deve gerar métricas consistentes.

Foi entendido que devido ao nosso pequeno conjunto de dados, definir as iterações para 1000 nos dará resultados mais confiáveis no final, pois teremos treinado e testado em quase todos os nossos pontos de dados, com a desvantagem do tempo de execução.

Após o teste de validação ao resultado para accuracy foi de : 0.9210526315789473, o que é satisfatório.

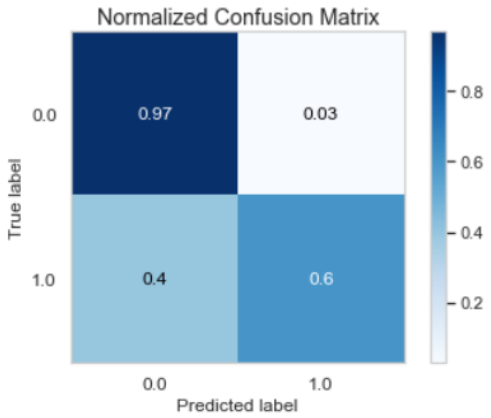
6. **Dê pelo menos duas métricas de avaliação e seu desempenho médio para cada uma delas. Explique uma interpretação de suas métricas que diz algo que seja compreensível ao homem sobre o desempenho do seu algoritmo. [rubrica de rubrica relevante: "uso de métricas de avaliação"]**

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

- **accuracy de 85-95%** - Este índice nos informa os acertos positivos dividido pelo número total de predições, ou seja, quão perto chemamos de classificar um POI.
- **precison de 65-75%** - Indica qe com uma confiança relevante, que quando identificado um POI, é muito provável que o seja e não apenas um alarme falso (falsos positivos);
- **recall de 35-60%** - O quanto o algoritimo é capaz de identificar um POI toda vez que ele aparece em casos de teste. Se for muito baixo indica muitos falsos negativos, onde os POIs não seriam punidos corretamente.

Com os resultados apresentados podemos verificar que, de modo geral, o classificador tem ótima performance para identificar a quantidade de pessoas de interesse no conjunto de dados, para identificar com mais segurança quais os funcionários são realmente um POI. Apesar de termos identificado um sobreajuste (overfitting) no KNeighbors Classifier, se mantermos sua configuração no padrão 5, manteremos resultados confiáveis.

Observando a matriz de confusão, conseguimos identificar 97% dos não POIs, para reduzir a margem de acusações ou desconfianças indevidas, e 60% dos POIs.



Referências:

pprint: Utilizada para melhor visualizar o dicionário. fonte: (https://docs.python.org/3/library/pprint.html)

Humphrey Gene E:

(https://books.google.com.br/books?id=LgxtDwAAQBAJ&pg=PT421&lpg=PT421&dq=HUMPHREY+GENE+E+enron&source=bl&ots=v5SeYD5bBi&sig=ACfU3U23G8B8IhRhZASZkxdcXSA6FbVFkQ&hl=ptBR&sa=X&ved=2ahUKEwjwgrGb1JjhAhVUA9QKHXisCh8Q6AEwBnoECAcQAQ#v=onepage&q=HUMPHREY%20GENE%20E%2 0enron&f=false)

Lay Kenneth L: https://pt.wikipedia.org/wiki/Kenneth_Lay

Skilling Jeffrey K: https://en.wikipedia.org/wiki/Jeffrey_Skilling

KNeighborsClassifier:https://paulovasconcellos.com.br/como-criar-seu-primeiro-aplicativo-de-machine-learning-7b6af291ba11
KNeighborsClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Site: http://minerandodados.com.br/index.php/2018/01/17/gridsearch-algoritmo-machine-learning/