Análise e Extração do Conhecimento: Classe Salarial Anual

José Parpot Nº: PG41848, Raimundo Barros Nº: PG42814, Tiago Gonçalves Nº: PG42851, and André Soares Nº: A67654

Universidade do Minho, R. da Universidade, 4710-057 Braga, Portugal

Abstract. Muitas tarefas de classificação binária não têm um número igual de exemplos de cada classe, por exemplo, a distribuição de classes é distorcida ou desequilibrada. Um exemplo popular é o conjunto de dados referente a renda de adultos que envolve a previsão de níveis de renda pessoal acima ou abaixo de $50k (dólares americanos) por ano com base em detalhes pessoais, como relacionamento e nível de educação. Dessa forma, uma análise exploratória e tarefas de pré-processamento são necessários para tratar a base de dados. Além disso, há muito mais casos de renda inferior a $50K do que acima de $50K, embora a diferença não seja grave. Isso significa que as técnicas de classificação podem ser usadas enquanto o desempenho do modelo ainda pode ser relatado usando como referência de avaliação a precisão do resultado do teste do modelo, para inferir a performance deste, como é utilizado na maioria dos problemas de classificação. Sendo assim, diferentes algoritmos preditivos foram implementados utilizando bibliotecas de *Machine Learning* em Python, a buscar a validação e optimização dos modelos.

Keywords: Renda Salarial · Classificação · *Machine Learning.*

1 Introdução

Os modelos preditivos de Machine Learning utilizam dados e algoritmos estatísticos para identificar a probabilidade de acontecimentos futuros a partir de dados históricos. A utilização destes modelos permite resolver problemas difíceis e descobrir novas oportunidades, sendo cada vez mais utilizados em diversas áreas.

Numa primeira fase da resolução do problema tivemos que fazer a análise e tratamento do dataset. Desta forma, é importante visualizar a distribuição do dataset de modo a interpretar a sua relação com a variável a prever (salary-classification).

Ao longo deste processo de tratamento de dados, os diferentes modelos preditivos deverão ser treinados, percebendo através de métricas de avaliação de performance, mais concretamente a métrica de acurácia, se o modelo se encontra validado.

2 Análise Exploratória

O primeiro passo em qualquer projecto de análise de dados é a compreensão do problema. Neste caso específico, o conjunto de dados é constituído por dois (2) ficheiros .csv (*datset* de treino e de teste), o conjunto de dados em questão foi partilhado pelos professores, contudo a fonte destes foram extraído do Repositório de *Machine Learning* da UCI. Esta é considerada a etapa mais importante do projecto, pois ela requer cerca de 70% do tempo total de dedicação. As principais informações e respostas para as questões iniciais podem ser obtidas através da análise exploratória dos dados, além disso, o desempenho dos modelos de *Machine Learning* construídos dependem directamente desta etapa do projecto.

2.1 Análise

A análise exploratória do *dataset* revela algumas informações básicas fundamentais como: tamanho do conjunto de dados; tipos de variáveis; distribuição estatísticas; valores ausentes, entre outras.

É fundamental também identificar a variável alvo *(target)* para a criação dos modelos, as correlações entre as variáveis e gerar os principais gráficos para o melhor entendimento do comportamento dos dados.

Nesta fase inicial, os dois *datasets* (Treino e Teste) foram unidos, em busca de facilitar a manipulação dos dados na fase de pré-processamento e eliminar algum tendência que possa existir, além disso, antes de implementar os modelos de *Machine Learning* pretende-se aplicar de forma aleatória a separação do *dataset* em subconjuntos de treino e teste. Dessa forma, com os *datasets* unidos observa-se que existem 48842 registos (observações) e 15 variáveis, sendo que 1 (uma) variável é a resposta (*target*) cujo os resultados são $\geq 50K, \leq 50K$.

Abaixo pode-se encontrar as informações das variáveis independentes (preditores):

1. *age*: idade da pessoa.
2. *workclass*: Privado, Empregado Autonômo, Governo Federal, Governo Local, Governo Estadual, Sem Pagamento, Nunca Trabalhou.
3. *fnlwgt*: É o número (estimado) de pessoas que cada linha nos dados representa.
4. *education*: Bacharelado, Alguma faculdade, 11º, Colegial, Escola Profissionalizante, Associação Académica, Assoc-voc, 9º, 7º-8º, 12º, Mestrado, 1º-4º, 10º, Doutorado, 5º-6º, Pré-escolar.
5. *education-num*: Número referente as classes do item anterior (educação).
6. *marital-status*: Casado-civil-cônjuge, Divorciado, Nunca-casado, Separado, Viúvo, Casado-cônjuge-ausente, Casado-AF-cônjuge.
7. *occupation*: Suporte técnico, Reparação de artesanato, Outros serviços, Vendas, Gerência Executiva, Profissional Espe-

cialista, Limpeza, Operador/Inspector de Máquinas, Administração, Pesca Agrícola, Movimentação/Transporte, Serviços Domésticos Particular, Serviço de Protecção, Forças Armadas.

8. *relationship*: Esposa, Filho Próprio, Marido, Não na Família, Outro Parente, Solteiro.

9. *race*: Branco, Asiático, Americano-Indio-Eskimo, Outro, Negro.

10. *sex*: Homem, Mulher

11. *capital-gain*: Ganho de Capital

12. *capital-loss*: Perda de Capital

13. *hours-per-week*: Jornada de trabalho semanal.

14. *native-country*: Estados Unidos, Cambodja, Inglaterra, Porto Rico, Canadá, Alemanha, Outlying-US (Guam-USVI-etc), Índia, Japão, Grécia, Sul, China, Cuba, Irão, Honduras, Filipinas, Itália , Polónia, Jamaica, Vietname, México, Portugal, Irlanda, França, República Dominicana, Laos, Equador, Taiwan, Haiti, Colômbia, Hungria, Guatemala, Nicarágua, Escócia, Tailândia, Jugoslávia, El-Salvador, Trindade e Tobago, Peru, Hong , Holanda, Holanda.

Entre as 14 variáveis listadas acima, 6 variáveis são do tipo numérico e 8 são do tipo categórico. A variável resposta também é categórica, que indica a quantidade de pessoas que possuem uma renda anual maior a $50 mil dólares (76%) ou menor ou igual a $50 mil dólares (24%).

2.2 Pré-Processamento

Antes de implementar e testar os algoritmos de *Machine Learning*, é necessário tratar e escolher os dados que irão sustentar o modelo. Nesta etapa, realiza-se os ajustes finais dos dados, estes ajustes envolvem por exemplo, a padronização das variáveis, criação de novas variáveis, limpeza dos dados, separação do *dataset* em treino e teste, modificação do tipo de variáveis, etc.

1. Com base na análise inicial do *dataset* observou-se que existe uma quantidade de campos, cujo o valor é "?" (valor ausente) distribuído ao longo das observações registadas nas variáveis. Portanto, a função *SimpleImputer()* foi aplicada sobre o *dataset* para substituir os valores ausentes pelos valores mais frequentes em cada coluna (variáveis) onde o mesmo se encontra.
   A estratégia para substituir os valores ausentes pelos valores mais frequentes foi aplicada, pois pretende-se utilizar todas as linhas que contêm a informação, e ao usar o valor mais frequente não afetará muito as colunas em um modo geral, além disso, a aplicação de outras estratégias podem utilizar valores *outliers*, por exemplo, a média dos valores.
2. Ao extrair as informações do *dataset* após a transformação dos valores ausentes em mais frequentes, observou-se que o tipo das variáveis estavam como *object* (pode conter vários tipos diferentes, por exemplo, a coluna a pode incluir

inteiros, flutuantes e strings que são rotulados colectivamente como um objecto). Portanto, foi necessário aplicar funções para transformar novamente as variáveis em tipo numérico e categórico.

```
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   age                   48842 non-null  int64
 1   workclass             48842 non-null  category
 2   fnlwgt                48842 non-null  int64
 3   education             48842 non-null  category
 4   education-num         48842 non-null  int64
 5   marital-status        48842 non-null  category
 6   occupation            48842 non-null  category
 7   relationship          48842 non-null  category
 8   race                  48842 non-null  category
 9   sex                   48842 non-null  category
 10  capital-gain          48842 non-null  int64
 11  capital-loss          48842 non-null  int64
 12  hours-per-week        48842 non-null  int64
 13  native-country        48842 non-null  category
 14  salary-classification  48842 non-null  category
dtypes: category(9), int64(6)
```

Fig. 1. Informações do *Database*

3. Um ponto importante é analise de distribuição dos dados, para isso foi utilizada a função *describe* para avaliar as métricas descritivas das variáveis numéricas (Fig. 2).

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

Fig. 2. Medidas Descritivas - Variáveis Numéricas

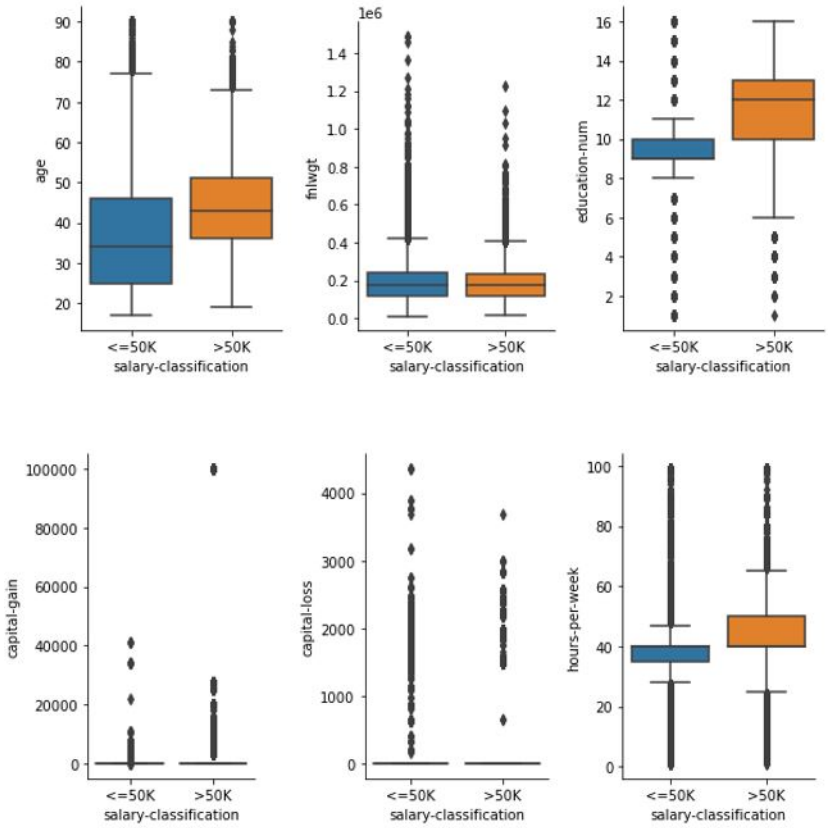

Fig. 3. Gráficos Catplot - Variáveis Numéricas

Gráficos da função *Catplot* (Fig. 2), mostram a relação entre uma variável numérica e uma ou mais variáveis categóricas usando uma das várias representações visuais. Dessa forma, estes gráficos foram gerados para facilitar a interpretação e obter as inferências sobre a relação das variáveis numéricas e a variável resposta, e a dispersão destes dados.

Ao observar a tabela e os gráficos acima, obteve-se as seguintes interpretações:

- **Idade:** Pessoas uma idade média 35 anos tem uma renda anual inferior a $50K, enquanto pessoas com idade média de 43 anos tem uma renda anual igual ou superior a $50K. Pode-se observar ainda que há ainda senhorios que estão ter uma renda anual, contudo são raro o número de casos.
- **Peso Final:** Em termos de estimativa da população, mostra que uma média de 200 mil pessoas recebem tanto abaixo de $50K quanto acima de $50K.

- **Educação:** Há uma concentração maior de pessoas com um nível escolar elevado que possuem salários superiores a $50K, enquanto que pessoas com baixo nível escolar possui salários menor do que $50K.
- **Ganho e Perca de Capital:** No geral observa-se que há uma concentração e proporção nas pessoas que ganham mais que $50K tendem a ter um maior ganho de capital e uma menor perca, enquanto quem ganha menos que $50K acontece o oposto.
- **Horas por Semana:** Fica evidente ao analisar o 2º quartil e o 3º quartil que as pessoas que costumam ter uma renda anual superior a $50K trabalham acima de 40 horas semanais, enquanto quem ganham menos, costumam trabalhar até 40 horas semanais.

  * Os *outliers* foram analisados e tratados em uma outra versão, principalmente nas variáveis *capital-gain* e *capital-loss*, pois estes possuem valores de desvio padrão superiores a média. Contudo tal modificação não fez com que a precisão do modelo melhorasse, portanto, decidiu-se manter os dados originais.

4. Outro ponto relevante na análise exploratória é a frequência de cada classe em suas respetivas variáveis, para isso foi utilizado uma função para realizar essa contagem e verificar quantas pessoas estão agrupadas em determinada classificação e posteriormente foi gerado um gráfico de barras.

   - **Classe de Trabalho:** Observa-se que a maioria das pessoas trabalham no sector privado, enquanto que a minoria nunca trabalhou.
   - **Educação:** A maioria do nível de educação das pessoas nessa base de dados é primeiramente o colegial, depois alguma faculdade e por fim licenciatura.
   - **Status Matrimonial:** A maioria das pessoas da base dados ou são casados no civil ou nunca casaram.
   - **Ocupação:** Um ponto interessante desse gráfico é que a maioria das pessoas são profissionais especialistas, contudo, com base nos resultados dessa amostra, a maioria das pessoas recebem salários abaixo de $50K anual.
   - **Sexo:** Observa-se que a maioria das pessoas que fazem parte dessa amostra são do sexo masculino.
   - **Raça:** A raça predominante é a Branca.
   - **Relacionamento:** A grande parte das pessoas que compõe a base de dados são casos, comprovando também a maioria do sexo ser masculino. Seguidamente, temos as pessoas que não possuem família.
   - **País Nativo:** Essa variável explica que a maioria dos casos analisados são de pessoas nativas dos Estados Unidos.

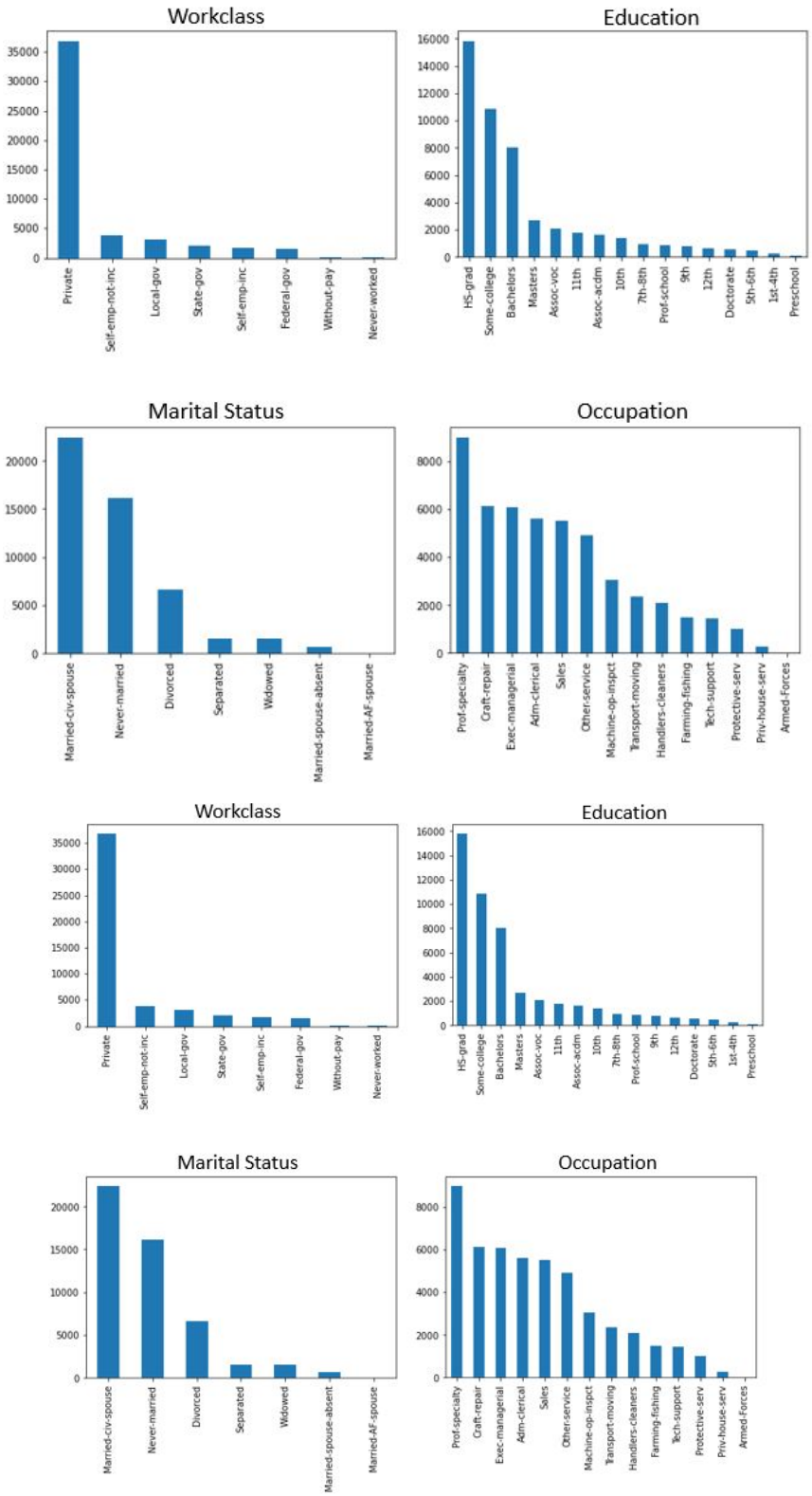

Fig. 4. Gráficos Barplot - Variáveis Categóricas

5. Em Python, não há opção para representar dados categóricos como factores. Os dados estão armazenados em *DataFrame*, podemos converter essas categorias em números usando o método *cat.codes* do pandas, e como as categorias nominais não são ordenadas, por padrão elas serão ordenadas alfabeticamente. Dessa forma, o *dataset* é parametrizado para ser utilizado nos modelos de *Machine Learning*.

3 Avaliação dos Modelos

3.1 Divisão do *Dataset*

Em preparação final para a aplicação dos modelos foi feita a separação do *dataset* em parte de treino e parte de teste, com uma divisão de 70% para o de treino e de 30% para o de teste.

```
#Criar subconjunto de dados de treino e de test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(preditors, target, test_size=0.3, random_state=1)
```

Fig. 5. Divisão do dataset para treino e teste

3.2 Naive Bayes

O primeiro modelo de avaliação testado foi o do Naive Bayes, que é um modelo de avaliação baseado no Teorema de Bayes (mostra a relação entre uma probabilidade condicional e a sua inversa). Ao lidar com dados contínuos, uma suposição típica é que os valores contínuos associados a cada classe são distribuídos de acordo com uma distribuição normal (ou gaussiana).

A sua aplicação foi simples, terminando com uma *accuracy* aproximada de 79.28%.

Naive Bayes

```
#Metodo Naive Bayes
from sklearn.naive_bayes import GaussianNB
NBmodel = GaussianNB()
```

```
#Executar o modelo e aplicar o predict sobre o modelo para uma posterior análise de desempenho do mesmo
NBmodel.fit(X_train, y_train)
NBpredict = NBmodel.predict(X_test)
```

```
#Confusion Matrix para Modelo Naive Bayes
print(classification_report(y_test, NBpredict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.94      0.87     11152
           1       0.64      0.31      0.42      3501

    accuracy                           0.79     14653
   macro avg       0.72      0.63      0.65     14653
weighted avg       0.77      0.79      0.77     14653
```

```
# Armazenar nas variáveis os Scores dos dataset de treino e teste
train_score = NBmodel.score(X_train, y_train)
test_score = NBmodel.score(X_test, y_test) #Accuracy
```

```
#Exibe no console os scores de treino e teste.
print(f'Gaussian Naive Bayes : Training score - {train_score} | Accuracy score - {test_score}')

#Armazena os valores obtidos acima no vetor performance.
performance.append({'Algoritmo':'Gaussian Naive Bayes', 'training_score':train_score, 'accuracy_score':test_score})
```

```
Gaussian Naive Bayes : Training score - 0.792886600953523 | Accuracy score - 0.7928069337337064
```

Fig. 6. Performance do modelo Naive Bayes

3.3 Logistic Regression

A regressão logística é uma técnica estatística que tem como objectivo modelar, a partir de um conjunto de observações, a relação "logística" entre uma variável resposta e uma série de variáveis explicativas numéricas (contínuas, discretas) e/ou categóricas. Na Regressão Logística, a variável resposta é binária:

1 - acontecimento de interesse (sucesso)

0 - acontecimento complementar (insucesso)

Logistic Regression

```
#Metodo Regressao Logistica
from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression()
```

```
logmodel.fit(X_train,y_train)
logpredict = logmodel.predict(X_test)
```

```
print(classification_report(y_test, logpredict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.97      0.88     11152
           1       0.72      0.25      0.38      3501

    accuracy                           0.80     14653
   macro avg       0.76      0.61      0.63     14653
weighted avg       0.79      0.80      0.76     14653
```

```
train_score = logmodel.score(X_train, y_train)
test_score = logmodel.score(X_test, y_test)

print(f'LogisticRegression : Training s core - {train_score} | Accuracy - {test_score}')

performance.append({'Algoritmo':'LogisticRegression', 'training_score':train_score, 'accuracy_score':test_score})
```

```
LogisticRegression : Training s core - 0.7987949340431133 | Accuracy - 0.7982665665733979
```

Fig. 7. Performance do modelo Logistic Regression

O segundo modelo aplicado obteve uma *accuracy* aproximada de 79.83%.

3.4 K-Nearest Neighbors

O terceiro modelo testado foi o de *K-Nearest Neighbors*. Para este modelo foi necessário a criação do gráfico com a média de erro por *k-value*, de modo a determinar qual o *k-value* ideal, usando o método do cotovelo. Depois de saber qual o *k-value* a aplicação do modelo foi simples e terminou com uma *accuracy* aproximada 80.05%.

K-Nearest Neighbors

```
# Metodo KNN
from sklearn.neighbors import KNeighborsClassifier
knn_scores = []
```

```
train_scores = []
test_scores = []
error = []

for n in range(1,40,2):
    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(X_train, y_train)
    pred_n = knn.predict(X_test)
    error.append(np.mean(pred_n != y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1, 40, 2), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

```
Text(0, 0.5, 'Mean Error')
```

Fig. 8. Criação do gráfico com a média do erro para o K-value

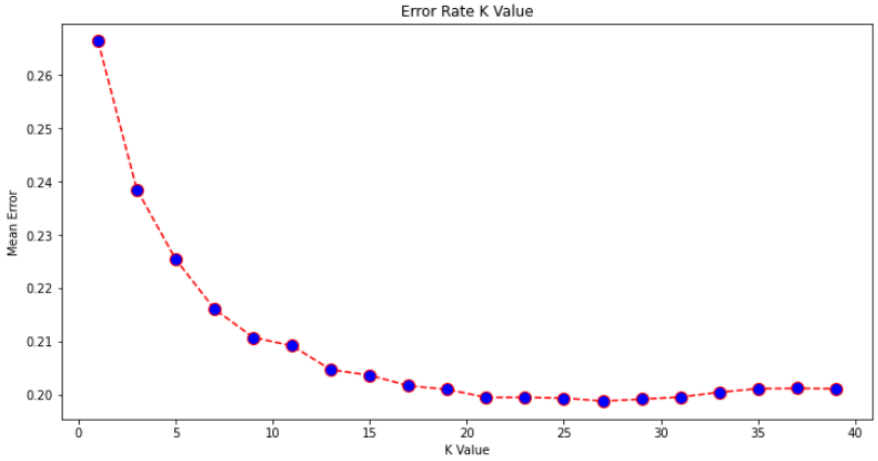

Fig. 9. Gráfico com a média do erro para o K-value

```
knn = KNeighborsClassifier(n_neighbors=21)
```

```
knn.fit(X_train, y_train)
knnpredict = knn.predict(X_test)
```

```
print(classification_report(y_test, knnpredict))
```

```
              precision    recall  f1-score   support

           0       0.80      0.98      0.88     11152
           1       0.80      0.22      0.34      3501

    accuracy                           0.80     14653
   macro avg       0.80      0.60      0.61     14653
weighted avg       0.80      0.80      0.75     14653
```

```
train_score = knn.score(X_train, y_train)
test_score = knn.score(X_test, y_test)

print(f'K Neighbors : Training score - {train_score} | Accuracy - {test_score}')

performance.append({'Algoritmo':'K Neighbors', 'training_score':train_score, 'accuracy_score':train_score})
```

```
K Neighbors : Training score - 0.8061657258182456 | Accuracy - 0.8005186651197707
```

Fig. 10. Performance do modelo KNN

3.5 Decision Tree

Em uma árvore de decisão cada folha da árvore é rotulada com uma classe ou distribuição de probabilidade sobre as classes, o que significa que o conjunto de dados foi classificado pela árvore em uma classe específica ou em uma distribuição de probabilidade específica. Uma árvore é construída dividindo o conjunto de origem, constituindo o nó raiz da árvore, em subconjuntos - que constituem os filhos sucessores. A divisão é baseada em um conjunto de regras de divisão com base em recursos de classificação. Este processo é repetido em cada subconjunto derivado de uma maneira recursiva chamada de particionamento recursivo . Essa repetição é concluída quando o subconjunto em um nó tem todos os mesmos valores da variável de destino ou quando a divisão não agrega mais valor às previsões.

Decision Tree

```
# Metodo Decison Tree (Arvore de Decisao)
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
```

```
dtc = dtc.fit(X_train, y_train)
dtcpredict = dtc.predict(X_test)
```

```
print(classification_report(y_test, dtcpredict))
```

```
              precision    recall  f1-score   support

           0       0.88      0.87      0.87     11152
           1       0.60      0.61      0.61      3501

    accuracy                           0.81     14653
   macro avg       0.74      0.74      0.74     14653
weighted avg       0.81      0.81      0.81     14653
```

```
train_score = dtc.score(X_train,y_train)
test_score = dtc.score(X_test,y_test)

print(f'Random Forests : Training score - {train_score} | Accuracy - {test_score}')

performance.append({'Algoritmo':'Decision Tree', 'training_score':train_score, 'accuracy_score':test_score})
```

```
Random Forests : Training score - 0.9999707508262892 | Accuracy - 0.8102095134102232
```

Fig. 11. Performance do modelo Decision Tree

Este modelo obteve uma *accuracy* aproximada de 81.02%.

3.6 Random Forest

É a principal especialização das árvores de decisão. Ou seja, o *Random Forest* que nada mais é do que uma colecção de árvores de decisão que seja ajusta a vários classificadores de árvore de decisão em várias sub-amostras do conjunto de dados e usa a média para melhorar a precisão da previsão.

Random Forest

```
# Metodo Random Forest (Floresta Aleatoria)
from sklearn.ensemble import RandomForestClassifier
rndTree = RandomForestClassifier()
```

```
rndTree.fit(X_train,y_train)
rndTpredict = rndTree.predict(X_test)
```

```
print(classification_report(y_test, rndTpredict))
```

```
              precision    recall  f1-score   support

           0       0.89      0.93      0.91     11152
           1       0.73      0.62      0.67      3501

    accuracy                           0.85     14653
   macro avg       0.81      0.77      0.79     14653
weighted avg       0.85      0.85      0.85     14653
```

```
train_score = rndTree.score(X_train,y_train)
test_score = rndTree.score(X_test,y_test)

print(f'Random Forests : Training score - {train_score} | Accuracy - {test_score}')

performance.append({'Algoritmo':'Random Forests', 'training_score':train_score, 'accuracy_score':test_score})
```

```
Random Forests : Training score - 0.9999707508262892 | Accuracy - 0.8542278031802362
```

Fig. 12. Performance do modelo Random Forest

O modelo aplicado obteve uma *accuracy* aproximada de 85.42%.

3.7 Comparação das Perfomances

Por fim, depois de realizados os testes e obtida a *accuracy* de cada modelo, decidimos comparar com o propósito de encontrar o modelo que apresente a melhor performance.

	Algoritmo	training_score	accuracy_score
0	Gaussian Naive Bayes	0.792887	0.792807
1	LogisticRegression	0.798795	0.798267
2	K Neighbors	0.806166	0.806166
3	Random Forests	0.999971	0.854228
4	Decision Tree	0.999971	0.810551

Fig. 13. Performance dos modelos testados

De acordo com a tabela contida na figura, podemos concluir que em termos de *accuracy* o melhor modelo é o de *Random Forest* com aproximadamente 85.42%, e como uma vantagem significativa relativamente aos outros modelos. Além disso, ao observar as informações geradas na *Confusion Matrix*, há 4 parâmetros que auxiliou também na tomada de decisão:

– O *precision*, é intuitivamente a capacidade do classificador de não rotular como positiva uma amostra negativa.

- O *recall*, é a capacidade do classificador de encontrar todas as amostras positivas.
- O *f-score*, pode ser interpretada como uma média harmónica ponderada da *precision* e *recall*.
- O *support*, é o número de ocorrências de cada classe para resultados verdados.

Dessa forma, o modelo escolhido foi o *Random Forest*, pois apresentou a melhor *accuracy* dos modelos testados, foi bastante consistente entre a fase de treino e a fase de teste e obteve os melhores resultados gerados na matriz confusão.

4 Conclusão

Ao longo de todo o processo deste projeto fomos ganhando percepção da importância das diversas fases no desenvolvimento de um modelo de classificação. Em tom de conclusão podemos referir que nos diferentes modelos apresentados ao longo deste relatório obteu-se a melhor *accuracy* no modelo de Random Forest com aproximadamente 85.42%, levando uma vantagem significativa em relação aos outros modelos.

O grupo dá-se satisfeito por melhorar o seu conhecimento e ter posto em prática os conhecimentos leccionados nas aulas, sabendo que ainda podíamos ter explorado os parâmetros das funções de *Machine Learning* para tentar ajustar os modelos e obter ainda melhores resultados.

References

1. Adult Data Set, https://archive.ics.uci.edu/ml/datasets/Adult. Último acesso 03 Jan 2021