FT 14
Curso: UFCD 10793
UFCD/Módulo/Temática: UFCD 10793
Ação: 10793_02/N
Formador/a: Sandra Liliana Meira de Oliveira
Nome do Formando/a:

Nesta atividade vamos efetuar a limpeza e o tratamento de dados dos passageiros do Titanic. Este afundou na sua viagem inaugural em 1912, após colidir com um iceberg. O acidente matou 1502 dis 2224 passageiros.

Uma das razões pelas quais o naufrágio provocou tal perda de vidas foi o facto de não haver botes salva-vidas suficientes para os passageiros e tripulantes. Embora houvesse algum elemento de sorte envolvido em sobreviver ao naufrágio, alguns grupos de pessoas eram mais propensos a sobreviver do que outros, como mulheres, crianças e a classe alta.

Estes dados podem, por exemplo, ser utilizados na análise de que tipo de pessoas poderiam sobreviver, aplicando as ferramentas de machine learning para prever quais os passageiros que sobreviveram à tragédia.

As variáveis presentes neste dataset são:

```
VARIABLE DESCRIPTIONS:
survival        Survival
                (0 = No; 1 = Yes)
pclass          Passenger Class
                (1 = 1st; 2 = 2nd; 3 = 3rd)
name            Name
sex             Sex
age             Age
sibsp           Number of Siblings/Spouses Aboard
parch           Number of Parents/Children Aboard
ticket          Ticket Number
fare            Passenger Fare
cabin           Cabin
embarked        Port of Embarkation
                (C = Cherbourg; Q = Queenstown; S = Southampton)

SPECIAL NOTES:
Pclass is a proxy for socio-economic status (SES)
 1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

Age is in Years; Fractional if Age less than One (1)
```

1

```
 If the Age is Estimated, it is in the form xx.5

With respect to the family relation variables (i.e. sibsp and parch)
some relations were ignored.  The following are the definitions used
for sibsp and parch.

Sibling:  Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard
Titanic
Spouse:   Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances
Ignored)
Parent:   Mother or Father of Passenger Aboard Titanic
Child:    Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic

Other family relatives excluded from this study include cousins,
nephews/nieces, aunts/uncles, and in-laws.  Some children travelled
only with a nanny, therefore parch=0 for them.  As well, some
travelled with very close friends or neighbors in a village, however,
the definitions do not support such relations.
```

1. Cria uma pasta com o nome FT14 no repositorio github (que está clonada localmente).
2. Abre a pasta no Visual Studio Code.
3. Adiciona à pasta o ficheiro limpezanulos.py
4. **No ficheiro anterior edita, analisa e corre o seguinte código – podes em alternativa usar um jupyter notebook.**

```python
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd
import numpy as np
import pylab as plt

#read the data
df_train = pd.read_csv('https://www.rootsystems.pt/train.csv')

print(df_train.head())
print(df_train.tail())

#view the data types of each column
print(df_train.dtypes)

#Type 'object' is a string for pandas, which poses problems with machine learning algorithms.
# #If we want to use these as features, we'll need to convert these to number representations.
#Get some basic information on the DataFrame:

print(df_train.info())

#we can see that Age, Cabin, and Embarked are missing values. Cabin has too many missing values,
#  whereas we might be able to infer values for Age and Embarked
#Now we are going to Generate various descriptive statistics on the DataFrame:
print(df_train.describe())

#vamos agora remover os nulos

## Lista as colunas e marca as que possuem algum valor nulo
print(df_train.isnull().any())
print()

# Na informação básica do dataframe, podemos ver o número de valores não nulos de cada coluna.
print(df_train.info())

#Como observado acima, temos 891 registros ao todo. As colunas Age, Cabin e Embarked são as que possuem valores nulos.
#O caso da coluna Cabin é o mais crítico, pois ele possui apenas 22% de dados preenchidos
# (provavelmente apenas pessoas da primeira classe) e essa informação pode não ser muito útil. Nesse caso,
```

```python
# vamos simplesmente remover a coluna de nossos dados.
#Obs: essa é uma opção apenas para exercitar as possibilidades de forma didática.
#  Vários algoritmos podem se beneficiar da informação da cabine, mesmo com poucos exemplos.
## Cria um novo dataframe (df_train2) sem a coluna Cabin

df_train2 = df_train.drop('Cabin', axis=1)
df_train2.info()

#Agora vamos analisar a coluna idade. Temos duas opções:

#Trocar nulo por um valor (média ou mediana)
#Ignorar
#A segunda opção limita o uso de alguns algoritmos, mas a primeira também tem os seus problemas.
# Vamos analisar a primeira opção e para isso vamos calcular a média e a mediana das idades
# e ver o histograma dos dados originais.

media_idade = df_train2['Age'].mean()
mediana_idade = df_train2['Age'].median()

print(media_idade)
print(mediana_idade)

df_train2['Age'].hist()
plt.title('Idade')
plt.show()

#Como a média e mediana são muito próximos, a diferença será mínima. Então, vamos trocar os nulos pela média e depois gerar novo histograma de idades.

## preenche os nulos com a média das idades
df_train2['Age'].fillna(media_idade, inplace=True) ## O inplace altera no próprio dataframe em vez de termos de criar outro
df_train2['Age'].hist()
plt.title('Idade')
plt.show()
```

```python
#Como podemos ver, o formato do histograma modifica-se completamente, mas essa será nossa opção para essa coluna.
# Por último vamos analisar a coluna Embarked, que tem poucos valores nulos. Vamos gerar um histograma:
# histograma só é possível com valores numéricos, mas Embarked é categórico
print(df_train2.groupby('Embarked')["PassengerId"].count()) #group by porto de embarque - número de passageiros por porto de embarque


#Como podemos ver nos dados resultantes , a  maioria dos passageiros embarcaram no porto de Southampton (S),
# então vamos, nesse caso, atribuir os valores em falta para o valor mais comum:

df_train2['Embarked'].fillna('S', inplace=True)
print(df_train2.info())

#Agora nossos dados não contém nulos. Vamos salvá-los para recuperar posteriormente.
df_train2.to_csv('train_no_nulls.csv', index=False)
```

5. Associa o ficheiro outliers.py à pasta anterior.
6. No ficheiro anterior edita, analisa e corre o seguinte código.

```python
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd
import numpy as np
import pylab as plt
# Vamos agora carregar nossos dados já sem os nulos
df_train = pd.read_csv('train_no_nulls.csv')

#vamos agora exercitar a remoção de outliers. Vale ressaltar que os dados originais foram modificados
# para incluir ocorrências incomuns em alguns atributos.
#O primeiro passo para identificar outliers é ver uma breve descrição dos dados numéricos.

print(df_train.describe())

#As colunas que nos interessam são `Age`, `SibSP`, `Parch` e `Fare`, pois as demais são categóricas ou apenas o ID do passageiro.
# Observando as 4 colunas, podemos ver algumas coisas incomuns:
#* A menor tarifa é negativa (o que é provavelmente um erro) e a maior é um número 10x maior que o Terceiro Quartil (75% percentil)
#  Há alguém com idade 133 anos. Hoje já parece improvável . Deve tratar-se também de um outlier (ou erro).
# A discussão sobre outliers é subjetiva. Sem o conhecimento do domínio do problema (como fizemos com a idade), é muito difícil dizer o que é um outlier.
# Mas para fins didáticos, vamos tratar aqui as anomalias como erros nos dados e vamos removê-los trocando pelo valor médio.
#Primeiro, vamos mostrar os 5 maiores e 5 menores idades.

print(df_train.sort_values('Age', ascending=False).head(5)['Age'])
print(df_train.sort_values('Age', ascending=True).head(5)['Age'])

#No caso da coluna idade, podemos notar que apenas o valor 133 parece fora da normalidade.
# Vamos trocá-lo pelo valor médio das idades, que já calculamos na limpezaNulos.py.

media_idade = df_train['Age'].mean()
df_train.loc[df_train['Age']==133, 'Age'] = media_idade ## Aqui substituimos os valores de idade iguais a 133 pela média

#Agora vamos fazer o mesmo para o campo tarifa
print(df_train.sort_values('Fare', ascending=False).head(5)['Fare'])
print(df_train.sort_values('Fare', ascending=True).head(5)['Fare'])

#Aqui confirmamos o valor negativo e um valor 10x do bilhete mais caro para o segundo mais caro.
# Vamos tratá-los como erros e atualizá-los com o valor da mediana da tarifa.
#  Usaremos a mediana em vez da média, pois ela é menos sensível aos outliers.
```


PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU

```
mediana_tarifa = df_train['Fare'].median()
df_train.loc[df_train['Fare']>5000, 'Fare'] = mediana_tarifa
df_train.loc[df_train['Fare']<0, 'Fare'] = mediana_tarifa

#vamos conferir a remoção de outliers

print(df_train.sort_values('Fare', ascending=False).head(5)['Fare'])
print(df_train.sort_values('Fare', ascending=True).head(5)['Fare'])

#Vamos agora guardar os nossos dados
df_train.to_csv('train_no_nulls_no_outliers.csv', index=False)
```

7. Associa o seguinte ficheiro à pasta featureEngineering.py
8. No ficheiro anterior edita, analisa e corre o seguinte código.

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd
import numpy as np
import pylab as plt

# Vamos agora exercitar algumas tarefas comuns na atividade de *Feature Engineering*,
# que consiste em remodelar atributos para obter melhor(?) desempenho nos algoritmos.
#  Entre as tarefas faremos conversão de categorias usando OHE (*one-hot-encoding*) e *Feature Hashing*,
#  e também a produção de novas colunas calculadas com base noutras.
# Não há uma "receita " para essas atividades. É sempre uma questão de experimentar  e ver quais modelos são melhores que outros.

##Recarregando os dados da parte anterior...
df_train = pd.read_csv('train_no_nulls_no_outliers.csv')
df_train.head(2)

# A técnica de One-hot-encoding é utilizada para converter dados que são categóricos em numéricos de forma
#  a não influenciar de forma negativa alguns algoritmos.
#  Converter cada valor possível para a coluna em números de 1 a N implicaria  haver uma relação fixa de
#  distância geométrica entre os dados, o que normalmente não ocorre.
# A conversão funciona de forma  simples: para cada categoria numa determinada coluna, é criada uma nova coluna onde o valor será 1
# quando a linha tiver aquele valor para a categoria ou 0 caso contrário.
# O código abaixo usa o método get_dummies da biblioteca pandas para criar 3 novas colunas para cada
# um dos 3 valores possíveis para a coluna `Embarked`.
novas_colunas = pd.get_dummies(df_train['Embarked'])
df_train2 = pd.concat([df_train,novas_colunas], axis=1) # axis = 1 concatena colunas. axis = 0 concatena linhas
print(df_train2.head(3))

#Agora podemos remover a coluna original dos dados
df_train2.drop('Embarked', axis=1, inplace=True)
```