I AVALIAÇÃO PARCIAL - 25.0 PTS

March 23, 2023

0.1 I PROVA PARCIAL - 25.0

0.2 INTELIGENCIA ARTIFICIAL - PROF. FISCHER STEFAN

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[3]:
```
df = pd.read_csv('Salary.csv')
```

[4]:
```
df.head()
```

[4]:

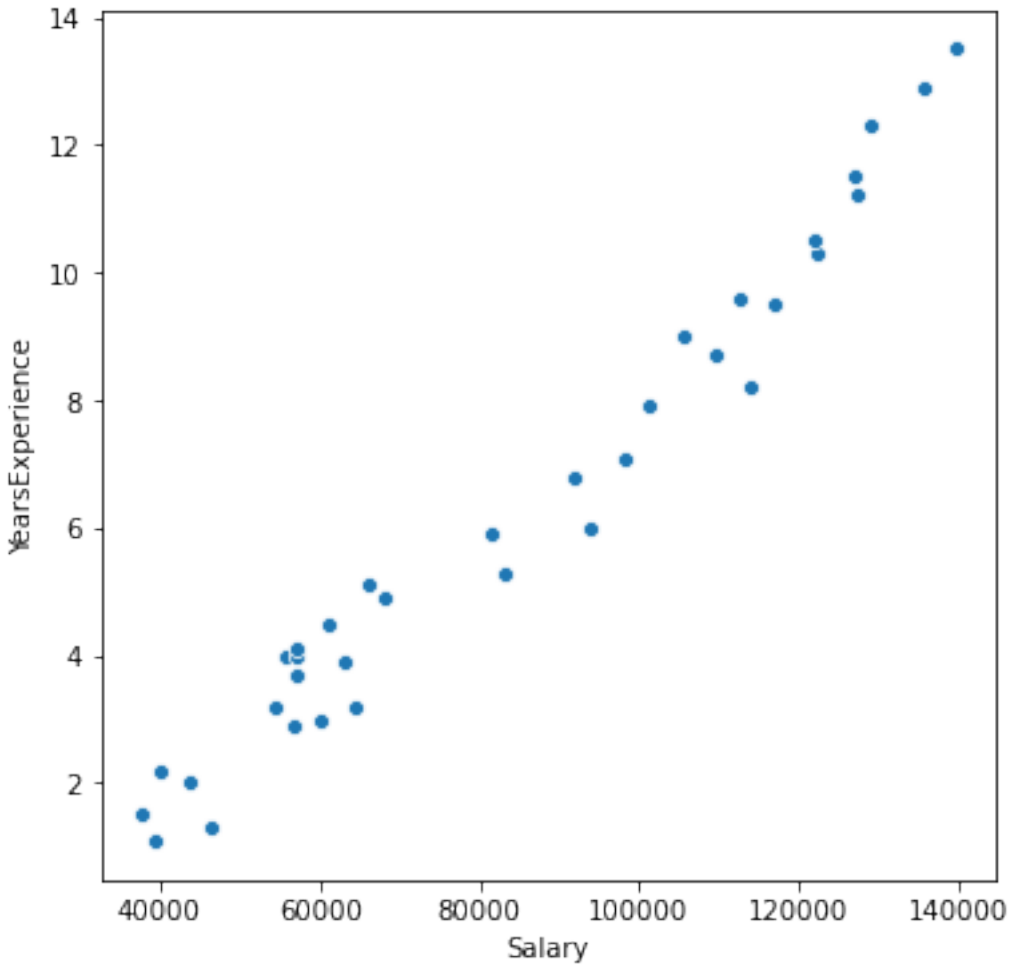
	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

QUESTÃO 1 - Reproduza o gráfico abaixo e explique a correlação dos dados #### entre salário e anos de experiência

[26]:
```

```

[26]:
```
<AxesSubplot:xlabel='Salary', ylabel='YearsExperience'>
```



[ ]:

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  35 non-null     float64
 1   Salary           35 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 688.0 bytes
```

[6]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
```

[7]:
```
from sklearn.model_selection import train_test_split
```

[8]:
```
X_data = df.iloc[:,:-1].values
y_data = df.iloc[:,-1].values
```

Questão 2 - Treine a base de dados e gere as métricas de acerto

usando Regressão Linear, replicando resultado abaixo.

[11]:
```
from sklearn.linear_model import LinearRegression
```

[12]:
```

```

[12]:
```
LinearRegression()
```

[17]:
```
y_pred
```

[17]:
```
array([ 80885.0981995 ,  56748.8141313 ,  88930.52622223, 148824.26816923,
       128263.72988891, 113066.81029042, 108597.12805557, 113960.74673739,
       138097.03080559,  41551.89453281, 120218.30186618])
```

[20]:
```

```

[21]:

```
R2_Score: 0.9775754814071884

Mean Absolute Error(MAE) : 3585.079612952305

Mean Squared Error(MSE): 21303651.024398852
```

1 Logistic Regression

[23]:

```
#X represents the size of a tumor in centimeters.
X = np.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69,␣
 ↪5.88]).reshape(-1,1)

#Note: X has to be reshaped into a column from a row for the␣
 ↪LogisticRegression() function to work.
#y represents whether or not the tumor is cancerous (0 for "No", 1 for "Yes").
y = np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

[24]:

```
from sklearn import linear_model
```

Questão 3 - Treine a base de dados para que ela decida se dado um input, #### consigamos premeditar se representa cancerígeno ou não.

[26]:

[26]:

```
LogisticRegression()
```

[27]:

```
#predict if tumor is cancerous where the size is 3.46mm:
predicted = LogReg.predict(np.array([3.46]).reshape(-1,1))
```

[28]:

```
predicted
```

[28]:

```
array([0])
```

Questão 4 - Imprima a matriz de confusão e o relatório de classificação

para verificar o grau de acerto. Explique a matriz de confusão e o grau

,probabilístico, de acerto.

[ ]:

[22]:

```
[[132  17]
 [ 11 140]]
```

```
              precision    recall  f1-score   support

           0       0.92      0.89      0.90       149
           1       0.89      0.93      0.91       151

    accuracy                           0.91       300
   macro avg       0.91      0.91      0.91       300
weighted avg       0.91      0.91      0.91       300
```

2 K Nearest Neighbors with Python

[29]:
```
df = pd.read_csv("Classified Data",index_col=0)
```

[30]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

[31]:
```
scaler.fit(df.drop('TARGET CLASS',axis=1))
```

[31]:
```
StandardScaler()
```

[32]:
```
scaled_features = scaler.transform(df.drop('TARGET CLASS',axis=1))
df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_feat.head()
```

[32]:

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	\
0	-0.123542	0.185907	-0.913431	0.319629	-1.033637	-2.308375	-0.798951	
1	-1.084836	-0.430348	-1.025313	0.625388	-0.444847	-1.152706	-1.129797	
2	-0.788702	0.339318	0.301511	0.755873	2.031693	-0.870156	2.599818	
3	0.982841	1.060193	-0.621399	0.625299	0.452820	-0.267220	1.750208	
4	1.139275	-0.640392	-0.709819	-0.057175	0.822886	-0.936773	0.596782	

	PJF	HQE	NXJ
0	-1.482368	-0.949719	-0.643314
1	-0.202240	-1.828051	0.636759
2	0.285707	-0.682494	-0.377850
3	1.066491	1.241325	-1.026987
4	-1.472352	1.040772	0.276510

2.1 Train Test Split

[34]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(scaled_features,df['TARGET␣
↪CLASS'],
                                                    test_size=0.30)
```

[35]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn = KNeighborsClassifier(n_neighbors=1)
```

[36]:
```
knn.fit(X_train,y_train)
```

[36]:
```
KNeighborsClassifier(n_neighbors=1)
```

[37]:
```
pred = knn.predict(X_test)
pred
```

[37]:
```
array([0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1])
```

Questão 5 -

a) Imprimir o resultado abaixo

[42]:
```

```

```
[[138   9]
 [ 11 142]]
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       147
           1       0.94      0.93      0.93       153

    accuracy                           0.93       300
   macro avg       0.93      0.93      0.93       300
weighted avg       0.93      0.93      0.93       300
```

b) Explique o que o código abaixo imprime, o motivo de se imprimir o gráfico (o que está sendo analisado) e a importância do parâmetro K e seu

significado

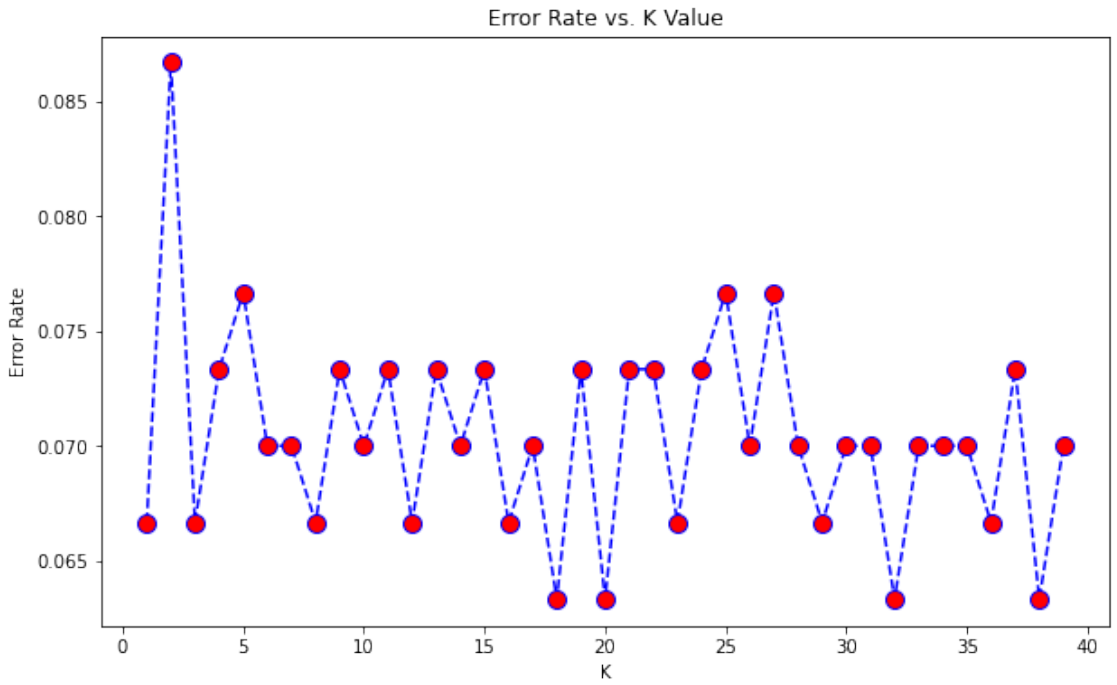
[40]:
```
error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

[41]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

[41]:
```
Text(0, 0.5, 'Error Rate')
```



[ ]:
```

```