DATA ANALYTICS REPORT

By,

N.Anandhi (15z306)

Exercise:2 **Web scraping**

Code:

```
import request

Required_page = requests.get("http://psgim.ac.in/2017/01/full-time-faculty/")

from bs4 import BeautifulSoup

soup = BeautifulSoup(page.content, 'html.parser')

for id in list(soup.select('div a')):

   s=id.text;

   if(len(s)==0):

      continue;

   if((id.text[0]=='D')and(id.text[1]=='R')):

        print(id.text)
```

output:

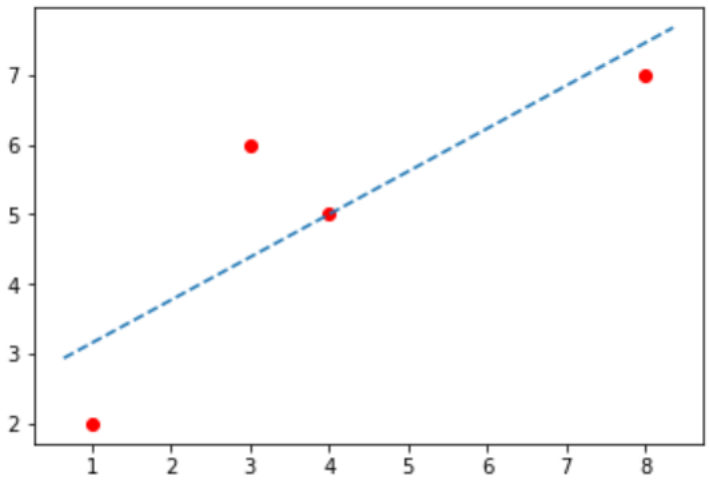
```
DR ARUL RAJAN K
DR Balasudarsun N L
DR DEEPA R
DR JAGAJEEVAN R
DR JOSHUA SELVAKUMAR J
DR KAVITHA D
DR KRISHNAVENI MUTHIAH R
DR MANSURALI A
DR RAMAN H
DR SATHISH M
DR SATHYANARAYANAN R S
DR SEKKIZHAR J
DR SUJATHA R
DR SUDHARANI RAVINDRAN D
DR SWAMYNATHAN R
DR THILAGAM V
DR UMA MAHESWARI B
DR UMESH CHANDRASEKHAR
DR VIVEK N
```

Exercise:3 Linear Regression

Code:

```
import numpy as np
x=[1,2,3,4]
y=[2,5,6,7]
for var in range(len(x)):
   x[var]=int(x[var])
   y[var]=int(y[var])

x_mean=float(np.mean(x))
y_mean=float(np.mean(y))
n=len(x)
s1=int(0); s2=int(0);
for var in range(len(x)):
   s1=s1+(x[var]*y[var])
   s2=s2+(x[var]*x[var])

slope=float((s1-(n*x_mean*y_mean))/(s2-(n*x_mean*x_mean)))
intercept=y_mean-x_mean*slope
y_vals=float((slope*x_mean)+intercept)

import matplotlib.pyplot as plt
plt.plot(x,y,'ro')

axes = plt.gca()
x_vals = np.array(axes.get_xlim())
y_vals = intercept + slope * x_vals
g1 = plt.plot(x_vals, y_vals, '--')
g2 = g1.add_subplot()
g2.show()
```

output:



Exercise :4 **Naïve bayes**

Code:

```
# Gaussian Naive Bayes
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
# load the iris datasets
dataset = datasets.load_iris()
from sklearn.model_selection import train_test_split
x,testX,y,testY=train_test_split(dataset.data,dataset.target,test_size=0.20)
# fit a Naive Bayes model to the data
model = GaussianNB()
model.fit(x, y)
print(model)
# make predictions
expected = testY
predicted = model.predict(testX)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
accuracy_score(expected,predicted)
GaussianNB(priors=None)
```

Output:

```
GaussianNB(priors=None)
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        14
          1       1.00      0.91      0.95        11
          2       0.83      1.00      0.91         5

avg / total       0.97      0.97      0.97        30

[[14  0  0]
 [ 0 10  1]
 [ 0  0  5]]

GaussianNB(priors=None)
```

Exercise:5 Neural networks

Code:

```
import pandas as pd
df1 = pd.read_csv("C:/Users/umarm/Downloads/movie_metadata.csv")
df1=df1[['num_critic_for_reviews','director_facebook_likes','actor_1_facebook
_likes','movie_facebook_likes','num_user_for_reviews','imdb_score']]
df1=df1.dropna()
df1 = df1[df1.movie_facebook_likes != 0]
df1 = df1[df1.director_facebook_likes != 0.0]
df1 = df1[df1.num_user_for_reviews != 0.0]
df1 = df1[df1.num_critic_for_reviews != 0.0]
df1 = df1[df1.actor_1_facebook_likes != 0.0]
df1 = df1[df1.imdb_score != 0.0]
x=df1[['num_critic_for_reviews','director_facebook_likes','actor_1_facebook_li
kes','movie_facebook_likes','num_user_for_reviews']]
y=df1[['imdb_score']]
y=y.round()
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(5, 4, 3), max_iter=1000,
alpha=0.00001, solver='sgd', verbose=10 )
mlp.fit(x_train, y_train.values.ravel())

y_pred = mlp.predict(x_test)
acc(y_test,y_pred)
```

output:

```
Iteration 1, loss = 3.86200346
Iteration 2, loss = 1.98077153
Iteration 3, loss = 1.96985569
Iteration 4, loss = 1.96071731
Iteration 5, loss = 1.95059066
Iteration 6, loss = 1.94188187
Iteration 7, loss = 1.93387695
Iteration 8, loss = 1.92657383
Iteration 9, loss = 1.91994831
Iteration 10, loss = 1.91332284
Iteration 11, loss = 1.90707189
Iteration 12, loss = 1.90076628
Iteration 13, loss = 1.89489692
Iteration 14, loss = 1.88902808
Iteration 15, loss = 1.88345539
Iteration 16, loss = 1.87790399
Iteration 17, loss = 1.87253344
Iteration 18, loss = 1.86723131
Iteration 19, loss = 1.86212169
```

Exercise-6 **Agglomerative clustering**

#Single Agglomerative clustering performed on the iris data set

Code:

```
import os
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display
from sklearn import datasets
iris = datasets.load_iris()
feat = iris.feature_names
X = iris.data[:, :2]  # we only take the first two features. We could
                      # avoid this ugly slicing by using a two-dim dataset
y = iris.target
y_name = ['Setosa', 'Versicolour', 'Virginica']
from sklearn.cluster import AgglomerativeClustering
clustering = AgglomerativeClustering(linkage="ward", n_clusters=3)
clustering.fit(X);



# MinMax scale the data so that it fits nicely onto the 0.0->1.0 axes of the plot.
from sklearn import preprocessing
X_plot = preprocessing.MinMaxScaler().fit_transform(X)
```

```
Z = linkage(X, 'ward')
colours = 'rbg'
for i in range(X.shape[0]):
   plt.text(X_plot[i, 0], X_plot[i, 1], str(clustering.labels_[i]),
            color=colours[y[i]],
            fontdict={'weight': 'bold', 'size': 9}
       )

plt.xticks([])
plt.yticks([])
plt.axis('off')
plt.show()
figure = plt.figure(figsize=(7.5, 5))
dendrogram(
   linkage_matrix,
   truncate_mode='lastp',  # show only the last p merged clusters
   p=24,  # show only the last p merged clusters
   leaf_rotation=90.,
   leaf_font_size=12.,
   show_contracted=True,  # to get a distribution impression in truncated
branches
)
plt.title('Hierarchical Clustering Dendrogram (Ward, aggrogated)')
plt.xlabel('sample index or (cluster size)')
plt.ylabel('distance')
plt.show()
```

output: