

AI PRACTICAL ASSIGNMENT

1. Find the correlation matrix.

Code :-

```
import numpy as np

temp=list(map(int, input("Temp : ").split()))

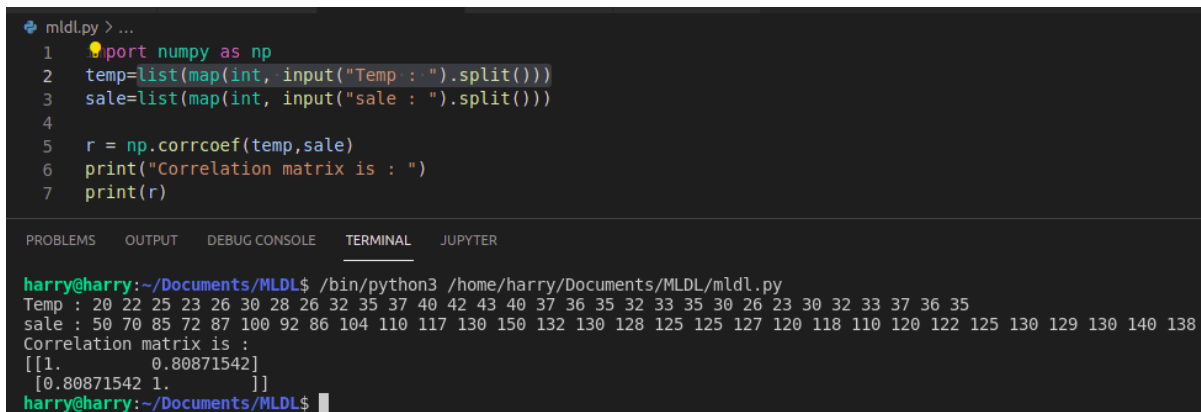
sale=list(map(int, input("sale : ").split()))


r = np.corrcoef(temp,sale)

print("Correlation matrix is : ")

print(r)
```

Output :



```
mldl.py > ...
1  import numpy as np
2  temp=list(map(int, input("Temp : ").split()))
3  sale=list(map(int, input("sale : ").split()))
4
5  r = np.corrcoef(temp,sale)
6  print("Correlation matrix is : ")
7  print(r)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

harry@harry:~/Documents/MLDL$ /bin/python3 /home/harry/Documents/MLDL/mldl.py
Temp : 20 22 25 23 26 30 28 26 32 35 37 40 42 43 40 37 36 35 32 33 35 30 26 23 30 32 33 37 36 35
sale : 50 70 85 72 87 100 92 86 104 110 117 130 150 132 130 128 125 125 127 120 118 110 120 122 125 130 129 130 140 138
Correlation matrix is :
[[1.          0.80871542]
 [0.80871542  1.        ]]
harry@harry:~/Documents/MLDL$
```

**** Find the correlation matrix using Data frame.**

Code:

```
import pandas as pd

data = {'temp':
[20,22,25,23,26,30,28,26,32,35,37,40,42,43,40,37,36,35,32,33,35,30,26,23,30,32,33,37,36,35
],
      'sales':
[50,70,85,72,87,100,92,86,104,110,117,130,150,132,130,128,125,125,127,120,118,110,120,1
22,125,130,129,130,140,138]
      }

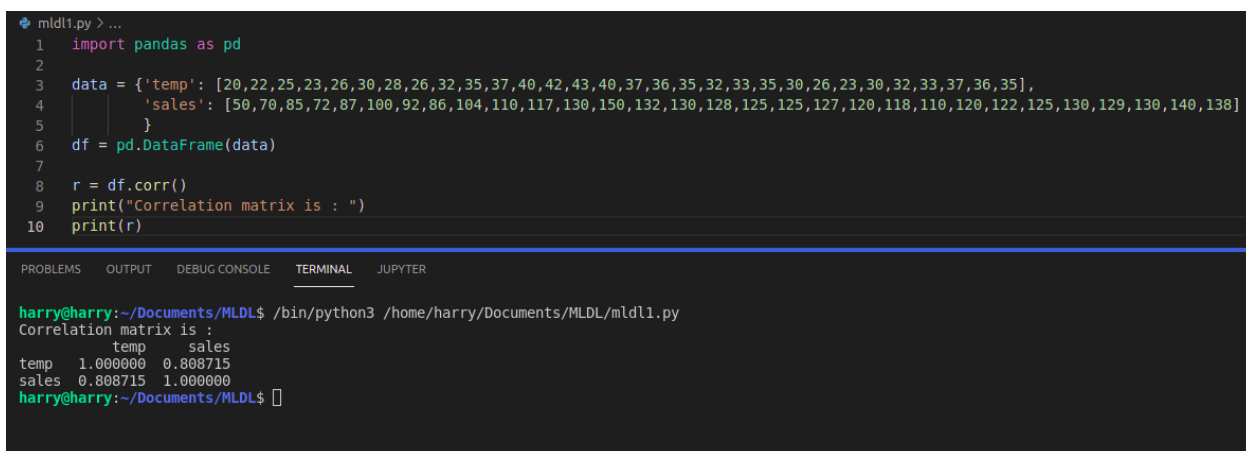
df = pd.DataFrame(data)

r = df.corr()

print("Correlation matrix is : ")

print(r)
```

Output :



```
mld1.py > ...
1  import pandas as pd
2
3  data = {'temp': [20,22,25,23,26,30,28,26,32,35,37,40,42,43,40,37,36,35,32,33,35,30,26,23,30,32,33,37,36,35],
4          'sales': [50,70,85,72,87,100,92,86,104,110,117,130,150,132,130,128,125,125,127,120,118,110,120,122,125,130,129,130,140,138]
5          }
6  df = pd.DataFrame(data)
7
8  r = df.corr()
9  print("Correlation matrix is : ")
10 print(r)
```

harry@harry:~/Documents/MLDL\$ /bin/python3 /home/harry/Documents/MLDL/mld1.py

```
Correlation matrix is :
      temp    sales
temp  1.000000  0.808715
sales  0.808715  1.000000
```

harry@harry:~/Documents/MLDL\$

****Find Correlation matrix using Formula .**

Code:

```
import math

def correlationCoefficient(X, Y, n) :

    sum_X = 0

    sum_Y = 0

    sum_XY = 0

    squareSum_X = 0

    squareSum_Y = 0

    i = 0

    while i < n :

        # sum of elements of array X.

        sum_X = sum_X + X[i]

        # sum of elements of array Y.

        sum_Y = sum_Y + Y[i]

        # sum of X[i] * Y[i].

        sum_XY = sum_XY + X[i] * Y[i]

        # sum of square of array elements.

        squareSum_X = squareSum_X + X[i] * X[i]

        squareSum_Y = squareSum_Y + Y[i] * Y[i]

        i += 1

    # use formula for calculating correlation coefficient.

    corr = (float)(n * sum_XY - sum_X * sum_Y)/(float)(math.sqrt((n * squareSum_X -
sum_X * sum_X)* (n * squareSum_Y - sum_Y * sum_Y)))

    return corr
```

```

# function arguments

X = list(map(int, input("Temp : ").split()))

Y = list(map(int, input("Sale : ").split()))

# Find the size of array.

n = len(X)

#Function call .

print("Correlation Coefficient is : '{}'".format(correlationCoefficient(X, Y, n)))

```

Output :

The screenshot shows a Jupyter Notebook with a Python script defining a function to calculate the correlation coefficient. The script is as follows:

```

1  import math
2
3  def correlationCoefficient(X, Y, n) :
4      sum_X = 0
5      sum_Y = 0
6      sum_XY = 0
7      squareSum_X = 0
8      squareSum_Y = 0
9
10     i = 0
11     while i < n :
12         # sum of elements of array X.
13         sum_X = sum_X + X[i]
14
15         # sum of elements of array Y.
16         sum_Y = sum_Y + Y[i]
17
18         # sum of X[i] * Y[i].
19         sum_XY = sum_XY + X[i] * Y[i]
20
21         # sum of square of array elements.
22         squareSum_X = squareSum_X + X[i] * X[i]
23         squareSum_Y = squareSum_Y + Y[i] * Y[i]
24
25     i += 1
26

```

The terminal output shows the execution of the script, where the user inputs two arrays of data (Temp and Sale) and the program outputs the calculated correlation coefficient.

```

harry@harry:~/Documents/MLDL$ /bin/python3 /home/harry/Documents/MLDL/mldl2.py
Temp : 20 22 25 23 26 30 28 26 32 35 37 40 42 43 40 37 36 35 32 33 35 30 26 23 30 32 33 37 36 35
Sale : 50 70 85 72 87 100 92 86 104 110 117 130 150 132 130 128 125 125 127 120 118 110 120 122 125 130 129 130 140 138
Correlation Coefficient is : 0.808715
harry@harry:~/Documents/MLDL$

```

02. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.

Code :-

#First, start with importing necessary Python packages –

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

#Next, download the iris dataset from its weblink as follows –

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

#Next, we need to assign column names to the dataset as follows –

```
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

#Now, we need to read dataset to pandas dataframe as follows –

```
dataset = pd.read_csv(path, names = headernames)
dataset.head()
```



	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

#Data Preprocessing will be done with the help of following script lines.

```
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, 4].values
X
Y
```

#Next, we will divide the data into train and test split. The following code will split the dataset into 70% training data and 30% of testing data –

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30)
```

#Next, train the model with the help of RandomForestClassifier class of sklearn as follows –

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(X_train, y_train)
RandomForestClassifier(n_estimators=50)
```

#At last, we need to make prediction. It can be done with the help of following script –

```
y_pred = classifier.predict(X_test)
```

#Next, print the results as follows –

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

Output :

```
... Confusion Matrix:
[[18  0  0]
 [ 0 13  1]
 [ 0  0 13]]
Classification Report:
              precision    recall  f1-score   support

   Iris-setosa       1.00      1.00      1.00        18
  Iris-versicolor    1.00      0.93      0.96        14
   Iris-virginica     0.93      1.00      0.96        13

   accuracy                   0.98        45
  macro avg              0.98      0.98      0.98        45
 weighted avg              0.98      0.98      0.98        45

Accuracy: 0.9777777777777777
```

03. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

Code :-

```
#First, start with importing necessary Python packages –
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#Next, download the iris dataset from its weblink as follows –
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

#Next, we need to assign column names to the dataset as follows –
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

#Now, we need to read dataset to pandas dataframe as follows –
dataset = pd.read_csv(path, names = headernames)
dataset.head()

#Data Preprocessing will be done with the help of following script lines.
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, 4].values
X
Y

#Next, we will divide the data into train and test split. The following code will split the dataset into
70% training data and 30% of testing data –
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30)

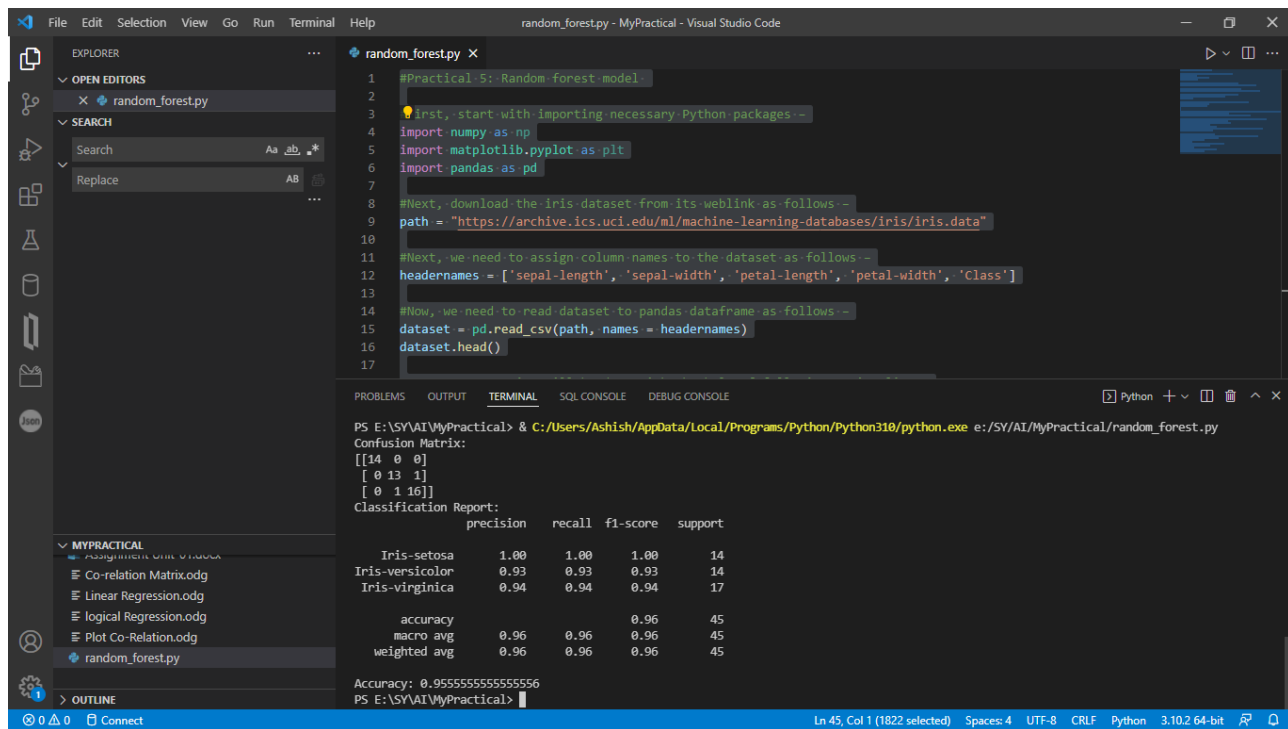
#Next, train the model with the help of RandomForestClassifier class of sklearn as follows –
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(X_train, y_train)
RandomForestClassifier(n_estimators=50)

#At last, we need to make prediction. It can be done with the help of following script –
y_pred = classifier.predict(X_test)

#Next, print the results as follows –
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:")
print(result1)
result2 = accuracy_score(y_test, y_pred)
```

```
print("Accuracy:",result2)
```

Output :-



The screenshot shows a Visual Studio Code editor with a Python file named `random_forest.py`. The code is a Jupyter-style notebook with comments and code cells. The code imports `numpy`, `matplotlib.pyplot`, and `pandas`. It downloads the Iris dataset from a web link, assigns column names, and reads the dataset into a pandas DataFrame. The output in the terminal shows the confusion matrix and a classification report.

```
#Practical 5: Random forest model
#First, start with importing necessary Python packages -
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#Next, download the iris dataset from its weblink as follows -
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

#Next, we need to assign column names to the dataset as follows -
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

#Now, we need to read dataset to pandas dataframe as follows -
dataset = pd.read_csv(path, names = headernames)
dataset.head()
```

PS E:\SV\AI\MyPractical> & C:/Users/Ashish/AppData/Local/Programs/Python/Python310/python.exe e:/SV/AI/MyPractical/random_forest.py

Confusion Matrix:

```
[[14  0  0]
 [ 0 13  1]
 [ 0  1 16]]
```

Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	14
Iris-versicolor	0.93	0.93	0.93	14
Iris-virginica	0.94	0.94	0.94	17
accuracy			0.96	45
macro avg	0.96	0.96	0.96	45
weighted avg	0.96	0.96	0.96	45

Accuracy: 0.9555555555555556

PS E:\SV\AI\MyPractical>

04. Apply linear regression Model techniques to predict the data on any dataset.

Code :-

- **Simple Linear Regression.**

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm

data = pd.read_csv('1.01 Simple linear regression.csv')
data

data.describe()

y = data ['GPA']

x1 = data ['SAT']

plt.scatter(x1,y)

plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)

plt.show()

x = sm.add_constant(x1)

results = sm.OLS(y,x).fit()

results.summary()

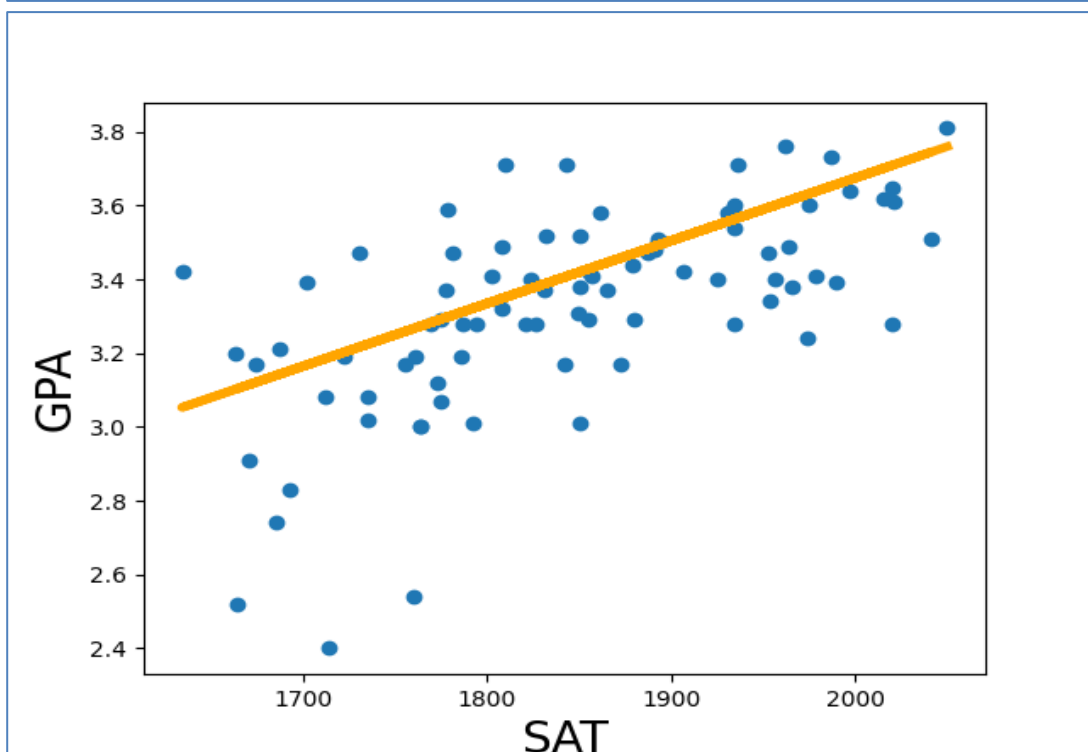
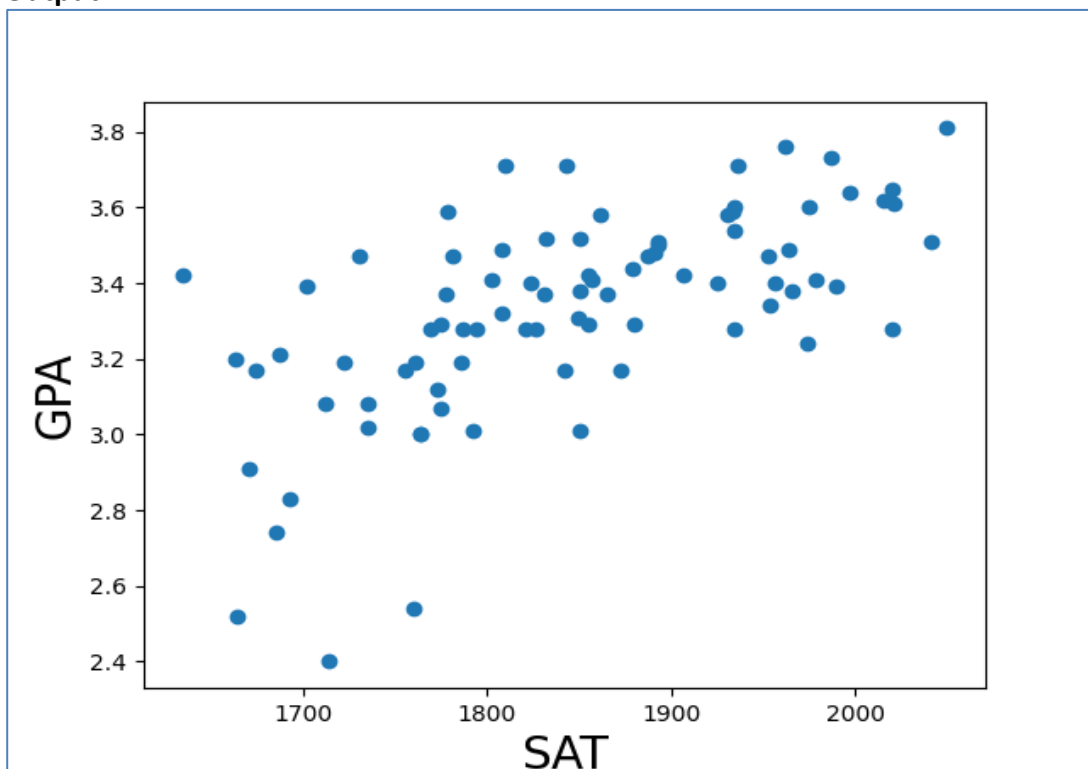
# Create a scatter plot
plt.scatter(x1,y)

# Define the regression equation, so we can plot it later
yhat = 0.0017*x1 + 0.275

# Plot the regression line against the independent variable (SAT)
fig = plt.plot(x1,yhat, lw=4, c='orange', label ='regression line')
```

```
# Label the axes
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)
plt.show()
```

Output:



05. Apply logical regression Model techniques to predict the data on any dataset.

Code :-

Logistic Regression :

Create a logistic regression based on the bank data provided.

The data is based on the marketing campaign efforts of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Note that the first column of the dataset is the index.

Import the relevant libraries

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

# this part not be needed after the latests updates of the library
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

Load the 'Example_bank_data.csv' dataset

```
from google.colab import files
uploaded = files.upload()
raw_data = pd.read_csv('Example_bank_data.csv')
raw_data

# Label the axes
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)
plt.show()
```

Output:

```
...
```

	Unnamed: 0	duration	y
0	0	117	no
1	1	274	yes
2	2	167	no
3	3	686	yes
4	4	157	no
...
513	513	204	no
514	514	806	yes
515	515	290	no
516	516	473	yes
517	517	142	no

518 rows × 3 columns

We want to know whether the bank marketing strategy was successful, so we need to transform the outcome variable into 0s and 1s in order to perform a logistic regression.

We make sure to create a copy of the data before we start altering it Note that we don't change the original data we loaded.

```
data = raw_data.copy()
```

Removes the index column that came with the data

```
data = data.drop(['Unnamed: 0'], axis = 1)
```

We use the map function to change any 'yes' values to 1 and 'no' values to 0.

```
data['y'] = data['y'].map({'yes':1, 'no':0})
```

```
data
```

Check the descriptive statistics

```
data.describe()
```

```
...
```

	duration	y
count	518.000000	518.000000
mean	382.177606	0.500000
std	344.295990	0.500483
min	9.000000	0.000000
25%	155.000000	0.000000
50%	266.500000	0.500000
75%	482.750000	1.000000
max	2653.000000	1.000000

Declare the dependent and independent variables

```
y = data['y']
x1 = data['duration']
```

Simple Logistic Regression

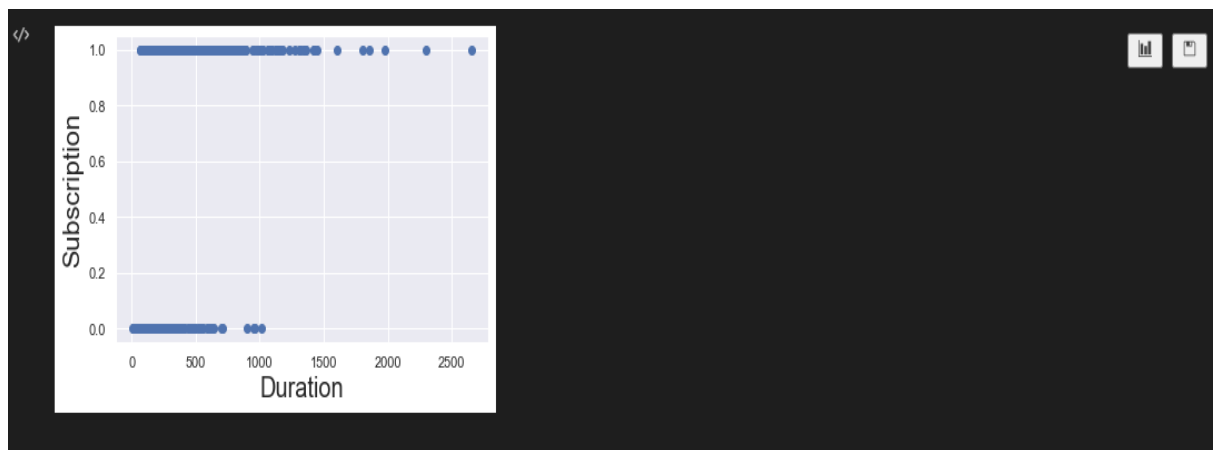
```
x = sm.add_constant(x1)
reg_log = sm.Logit(y,x)
results_log = reg_log.fit()
```

```
# Get the regression summary
results_log.summary()
```

```
# Create a scatter plot of x1 (Duration, no constant) and y (Subscribed)
plt.scatter(x1,y,color = 'C0')
```

```
# Don't forget to label your axes!
plt.xlabel('Duration', fontsize = 20)
plt.ylabel('Subscription', fontsize = 20)
plt.show()
```

```
np.set_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})
```



```

#np.set_printoptions(formatter=None)
results_log.predict()

np.array(data['y'])
results_log.pred_table()

cm_df = pd.DataFrame(results_log.pred_table())
cm_df.columns = ['Predicted 0', 'Predicted 1']
cm_df = cm_df.rename(index={0: 'Actual 0', 1: 'Actual 1'})
cm_df

cm = np.array(cm_df)
accuracy_train = (cm[0,0]+cm[1,1])/cm.sum()
accuracy_train

```

```

#np.set_printoptions(formatter=None)
results_log.predict()

np.array(data['y'])
results_log.pred_table()

cm_df = pd.DataFrame(results_log.pred_table())
cm_df.columns = ['Predicted 0', 'Predicted 1']
cm_df = cm_df.rename(index={0: 'Actual 0', 1: 'Actual 1'})
cm_df

cm = np.array(cm_df)
accuracy_train = (cm[0,0]+cm[1,1])/cm.sum()
accuracy_train

```

[7] ✓ 0.4s Python

... 0.693050193050193