



Parle Tilak Vidyalaya Associations

SATHAYE COLLEGE (Autonomous)

Vile-Parle (East), Mumbai – 400 057.

Practical Journal

Big Data Analytics

Submitted by

Mazhar Iqbal Solkar

Seat No : 32

M.Sc. [I.T.]-Information Technology Part I
2021 – 2022



Parle Tilak Vidyalaya Association's
SATHAYE COLLEGE (Autonomous)

Vile-Parle (East), Mumbai – 400 057.

CERTIFICATE

This is to certify that **Mazhar Iqbal Solkar**
Seat No **32** has successfully completed all the practicals
in the subject of **Big Data Analytics** for M.Sc.I.T. Part-I
SEM – II as prescribed by University of Mumbai for the
year 2021-2022.

Coordinator
M.Sc. [I.T.]

Professor in Charge

External Examiner

Date:

Date

Date

INDEX

Sr.No	Topic	Date	Sign
1	Implementing k-means classification Technique.	08-Feb-2022	
2	Implementing Linear Regression.	15-Feb-2022	
3	Implementing Logistic Regression.	28-Feb-2022	
4	Implement an application that stores data in MongoDB and manipulate it using python.	21-Mar-2022	
5	SVM classification Technique.	06-Apr-2022	
6	Tree classification Technique.	30-Apr-2022	

Practical No 1

Aim: Implementing k-means classification Technique.

Description :-

The algorithm will categorize the items into k groups of similarity. To calculate that similarity, we will use the euclidean distance as measurement.

The algorithm works as follows:

First, we initialize k points, called means, randomly. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far. We repeat the process for a given number of iterations and at the end, we have our clusters.

Methods :-

1. `numpy.random.randint(low, high=None, size=None)` :-

Return random integers from low (inclusive) to high (exclusive).

2. `matplotlib.pyplot.figure(figsize=(x,y))` :-

Create a new figure, or activate an existing figure.

3. `matplotlib.pyplot.scatter(x, y, color='k')` :-

With Pyplot, you can use the `scatter()` function to draw a scatter plot.

4. `matplotlib.pyplot.xlim(*args, **kwargs)` :-

The `xlim()` function in pyplot module of matplotlib library is used to get or set the x-limits of the current axes.

5. `matplotlib.pyplot.ylim(*args, **kwargs)` :-

The `ylim()` function in pyplot module of matplotlib library is used to get or set the y-limits of the current axes.

6. `matplotlib.pyplot.show()` :-

This method is used to display the graph.

7. `df.head()` :-

This method is used to obtain size of the dataset.

Program :-

```
## Initialisation

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

df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24]
})

np.random.seed(200)
k = 3
# centroids[i] = [x, y]
centroids = {
    i+1: [np.random.randint(0, 80), np.random.randint(0, 80)]
    for i in range(k)
}

fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color='k')
colmap = {1: 'r', 2: 'g', 3: 'b'}
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.title("Initialisation step __By Mazhar Solkar")
plt.show()

## Assignment Stage

def assignment(df, centroids):
    for i in centroids.keys():
        # sqrt((x1 - x2)^2 - (y1 - y2)^2)
        df['distance_from_{}'.format(i)] = (
            np.sqrt((df['x'] - centroids[i][0]) ** 2 + (df['y'] - centroids[i][1]) ** 2)
        )
    centroid_distance_cols = ['distance_from_{}'.format(i) for i in centroids.keys()]
    df['closest'] = df.loc[:, centroid_distance_cols].idxmin(axis=1)
    df['closest'] = df['closest'].map(lambda x: int(x.lstrip('distance_from_')))
    df['color'] = df['closest'].map(lambda x: colmap[x])
    return df

df = assignment(df, centroids)
print(df.head())

fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.title("Assignment Stage __By Mazhar Solkar")
plt.show()
```

```
## Update Stage

import copy

old_centroids = copy.deepcopy(centroids)

def update(k):
    for i in centroids.keys():
        centroids[i][0] = np.mean(df[df['closest'] == i]['x'])
        centroids[i][1] = np.mean(df[df['closest'] == i]['y'])
    return k

centroids = update(centroids)

fig = plt.figure(figsize=(5, 5))
ax = plt.axes()
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
for i in old_centroids.keys():
    old_x = old_centroids[i][0]
    old_y = old_centroids[i][1]
    dx = (centroids[i][0] - old_centroids[i][0]) * 0.75
    dy = (centroids[i][1] - old_centroids[i][1]) * 0.75
    ax.arrow(old_x, old_y, dx, dy, head_width=2, head_length=3, fc=colmap[i], ec=colmap[i])
plt.title("Update Stage __By Mazhar Solkar")
plt.show()

## Repeat Assignment Stage

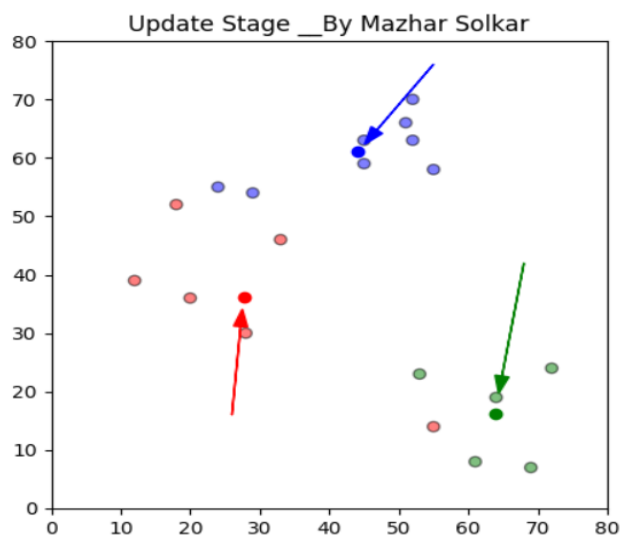
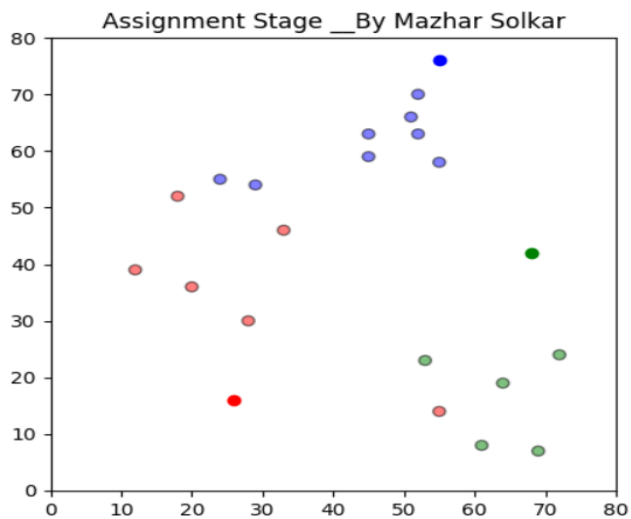
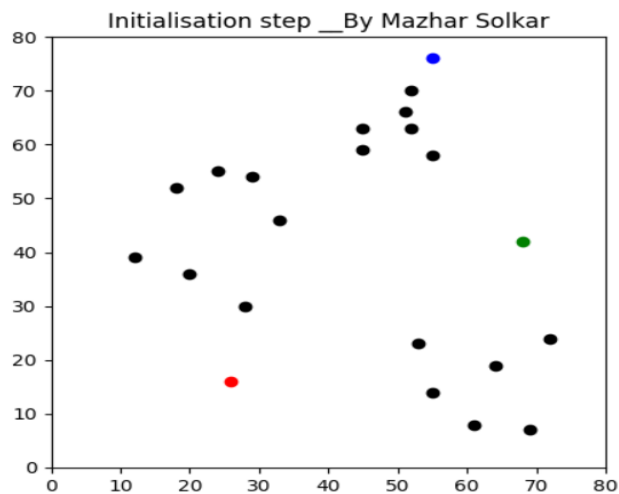
df = assignment(df, centroids)

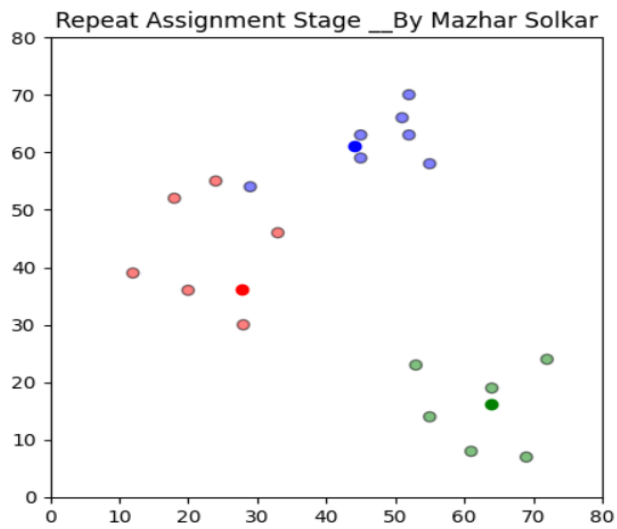
# Plot results
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.title("Repeat Assignment Stage __By Mazhar Solkar")
plt.show()

# Continue until all assigned categories don't change any more
while True:
    closest_centroids = df['closest'].copy(deep=True)
    centroids = update(centroids)
    df = assignment(df, centroids)
    if closest_centroids.equals(df['closest']):
        break

fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.title("Final Stage __By Mazhar Solkar")
plt.show()
```

Output :-





Practical No 2

Aim: Implementing Linear Regression

Description: -

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price**, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

Methods :-

1. `pd.read_csv(inputfilename) :-`

This method is used to read the csv files.

2. `dataframe.iloc[:,[colno_1,colno_3]] :-`

This method is used to fetch specific row of specific columns.

3. `train_test_split(x,y,test_size=0.25,random_state=0) :-`

This method is used to split dataframe into training and testing dataset.

4. `StandardScaler() :-`

This method is used for feature scaling.

5. `SVC(kernel='linear', random_state=0) :-`

This method is used for linear support vector classifier.

6. `metrics.accuracy_score(y_test,y_pred) :-`

This method is used to check the accuracy score.

7. `model.coef_ :-`

`model.coef_` is used to obtain coefficient value and

8. model.intercept_ :-

model.intercept_ is used to obtain intercept value.

9. model.score(waist,weight) :-

This method is used to check the accuracy of the model.

10. model.predict(Waist_new) :-

This method is used to predict the value based on trained dataset.

11. data.corr() :-

This method is used to obtain correlation.

12. lm.fit(waist, weight) :-

fit() is used to train model.

Program :-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model

data=pd.read_csv("C:/0_MSc_IT_Notes/Big Data Analytics/practicals/linear_regression_practical/weightwaist.csv")
print(data)
data.plot(kind='scatter',x='waist_cm',y='weight_kg')
plt.title('__By Mazhar Solkar')
plt.show()

print('\nCorrelation')
print(data.corr())

# Defining dependent and independent variables
waist=pd.DataFrame(data['waist_cm'])
weight=pd.DataFrame(data['weight_kg'])

print('\nwaist')
print(waist)
print('\nweight')
print(weight)

#implementing linear regression
lm =linear_model.LinearRegression()
model = lm.fit(waist,weight)

print('\nCoefficient')
print(model.coef_)
print('\nintercept')
print(model.intercept_)
print('\nscore')
print(model.score(waist,weight))

Waist_new = np.array([97])
Waist_new = Waist_new.reshape(-1,1)
Weight_predict = model.predict(Waist_new)
print('\nWeight_predict')
print(Weight_predict)

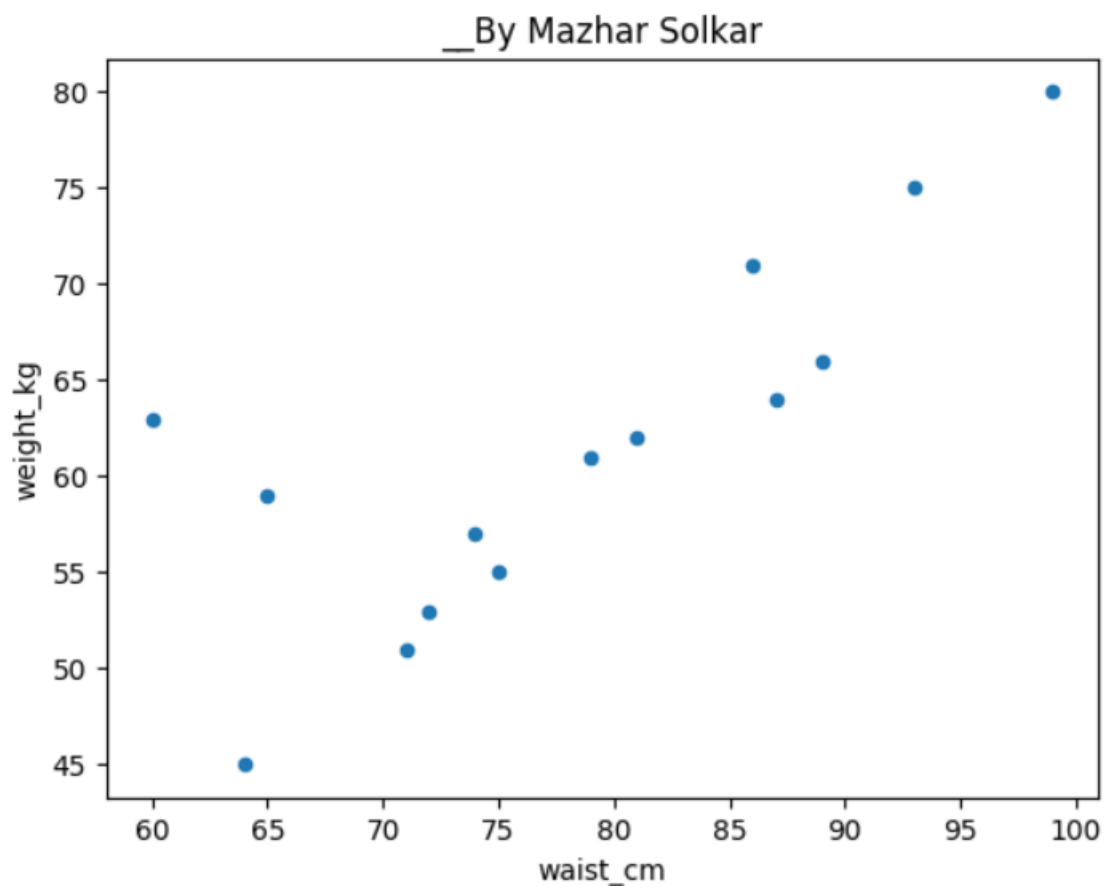
X=([67,78,94])
X=pd.DataFrame(X)
Y=model.predict(X)
Y=pd.DataFrame(Y)
df = pd.concat([X,Y], axis=1, keys=['Waist_new','Weight_predicted'])
print('\ndf')
print(df)

data.plot(kind='scatter',x='waist_cm',y='weight_kg')
plt.plot(waist,model.predict(waist),color='red', linewidth=2)
plt.scatter(Waist_new,Weight_predict, color='black')
plt.title('__By Mazhar Solkar')
plt.show()
print('__By Mazhar Solkar')
```

Output :-

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Mazhar@DESKTOP-0PG7LTN MINGW64 /c:/0_MSc_IT_Notes/Big Data Analytics/practicals (main)
$ python -u "c:\0_MSc_IT_Notes\Big Data Analytics\practicals\linear_regression_practical\linear_regression.py"
waist_cm weight_kg
0      71      51
1      89      66
2      64      45
3      74      57
4      87      64
5      93      75
6      79      61
7      81      62
8      75      55
9      72      53
10     65      59
11     60      63
12     99      80
13     86      71
□
```



```
Correlation
waist_cm  waist_cm  weight_kg
waist_cm  1.000000  0.798577
weight_kg 0.798577  1.000000
```

```
waist
waist_cm
0      71
1      89
2      64
3      74
4      87
5      93
6      79
7      81
8      75
9      72
10     65
11     60
12     99
13     86
```

```
weight
weight_kg
0      51
1      66
2      45
3      57
4      64
5      75
6      61
7      62
8      55
9      53
10     59
11     63
12     80
13     71
```

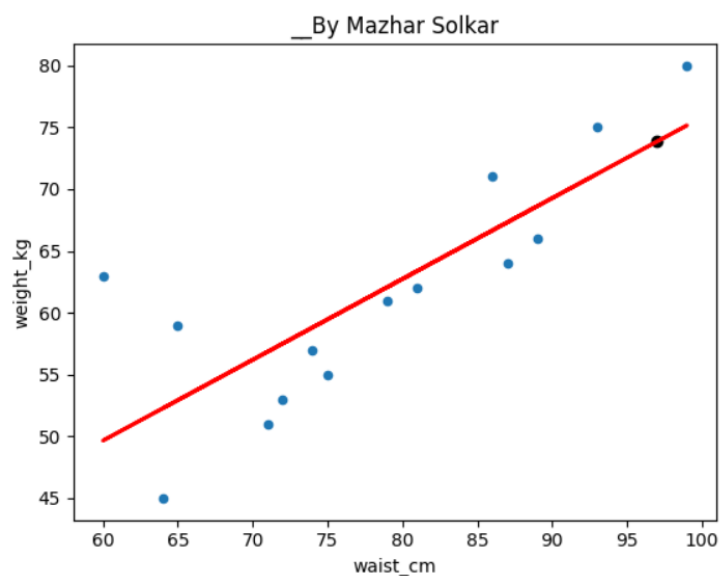
```
Coefficient
[[0.65405294]]
```

```
intercept
[10.41514467]
```

```
score
0.6377256319321334
```

```
Weight_predict
[[73.85828032]]
```

```
df
Waist_new Weight_predicted
0          0
0         67      54.236692
1         78      61.431274
2         94      71.896121
__By Mazhar Solkar
```



Practical No 3

Aim : Implementing Logistic Regression.

Description :-

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Methods :-

LogisticRegression() :-

This method is used to implement logistic regression.

Program :-

```
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
import pandas as pd

# Generate and dataset for Logistic Regression
x, y = make_classification(
    n_samples=100,
    n_features=1,
    n_classes=2,
    n_clusters_per_class=1,
    flip_y=0.03,
    n_informative=1,
    n_redundant=0,
    n_repeated=0
)

# Create a scatter plot
plt.scatter(x, y, c=y, cmap='rainbow')
plt.title('Scatter Plot of Logistic Regression')
plt.title('__By Mazhar Solkar')
plt.show()

# Split the dataset into training and test dataset
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1)

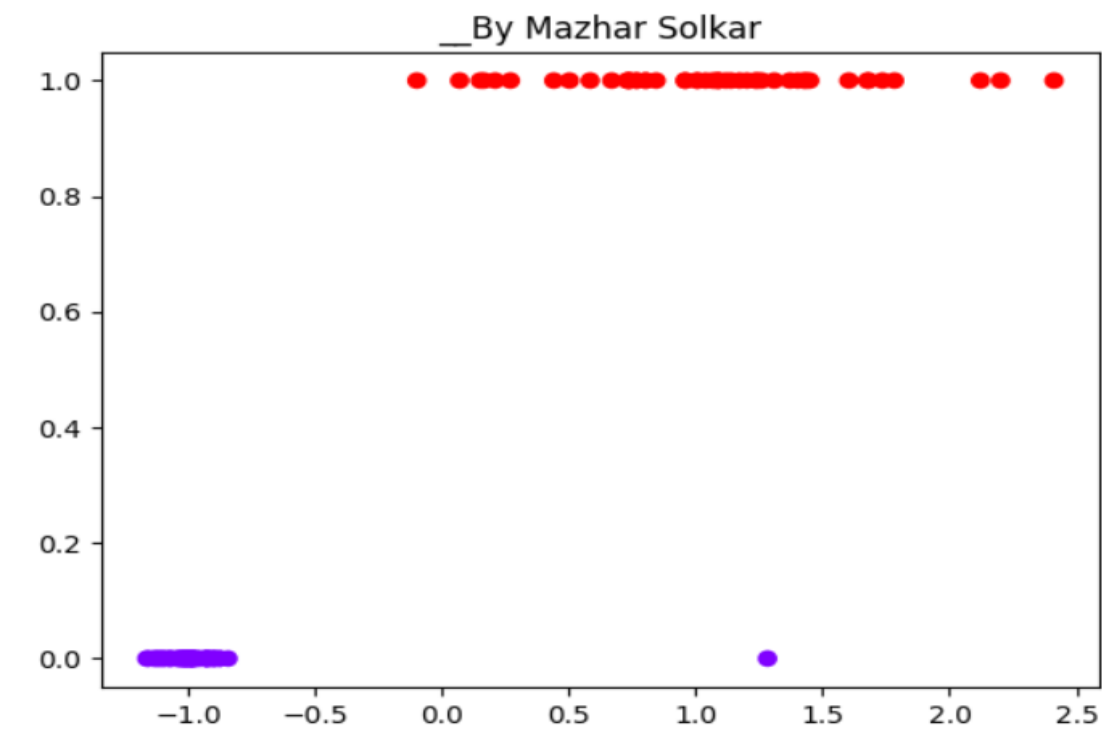
# Create a Logistic Regression Object, perform Logistic Regression
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)

# Show to Coefficient and Intercept
print("\n Coefficient_")
print(log_reg.coef_)
print("\n Intercept")
print(log_reg.intercept_)

# Perform prediction using the test dataset
y_pred = log_reg.predict(x_test)

# Show the Confusion Matrix
from sklearn.metrics import confusion_matrix
print("\n Confusion Matrix")
print(confusion_matrix(y_test, y_pred))
print('__By Mazhar Solkar')
```

Output :-



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Mazhar@DESKTOP-0PG7LTN MINGW64 /c/0_MSc_IT_Notes/Big Data Analytics/practicals (main)
$ python -u "c:\0_MSc_IT_Notes\Big Data Analytics\practicals\logistic_regression\logistic_regression.py"

Coefficient_
[[2.77004992]]

Intercept
[-0.08136218]

Confusion Matrix
[[11  0]
 [ 1 13]]
__By Mazhar Solkar
```


Practical No 4

Aim: Implement an application that stores big data in MongoDB and manipulate it using python.

Description :-

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data.

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NOSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

Methods :-

1. `MongoClient('localhost:27017')` :-

This method is used to get at which port monodb is running.

2. `client.get_database('database_name')` :-

This method is used to access the database.

3. `db.records_name` :-

This method is used to access the collection of database.

4. `records.count_documents({})` :-

This method is used to count the number of records in the collection.

5. `list(records.find())` :-

This method is used to print all the records in collection.

6. `records.update_one({"$set":{"key","value"}}) :-`

This method is used to update one record in collection.

7. `records.insert_one({"eno":6,"name":"Raj","location":"India"}) :-`

This method is used to insert one record in collection.

8. `records.delete_one({"name":"Raj"}) :-`

This method is used to delete one record from collection.

Program :-

```
from pymongo import MongoClient
client = MongoClient('localhost:27017')
db = client.get_database('sample')
records = db.employee

print("\n##### Count of Records #####")
print(records.count_documents({}))

print("\n##### list of records #####")
print(list(records.find()))







print("\n##### one record updated #####")
myquery = { "eno" : 4}
newvalues = { "$set":{"ename":"Laxman"}}
records.update_one(myquery,newvalues)
for v in records.find():
|   print(v)

print("\n##### one record inserted #####")
myq1={"eno":6,"name":"Raj","location":"India"}
x=records.insert_one(myq1)
for v in records.find():
|   print(v)

print("\n##### one record deleted #####")
y=records.delete_one({"name":"Raj"})
for v in records.find():
|   print(v)
print("\n__By Mazhar Solkar")
```

Output :-

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

 Code     

```
Mazhar@DESKTOP-0PG7LTN MINGW64 /c:/0_MSc_IT_Notes/Big Data Analytics/practicals (main)
$ python -u "c:\0_MSc_IT_Notes\Big Data Analytics\practicals\mongo_practical\mongo.py"

##### Count of Records #####
5

##### list of records #####
[{'_id': ObjectId('6258447d8ce536ffffa4f97d'), 'eno': 1.0, 'ename': 'amar', 'deptno': 10.0}, {'_id': ObjectId('625844aa8ce536ffffa4f97e'), 'eno': 2.0, 'ename': 'arjun', 'deptno': 10.0}, {'_id': ObjectId('625844d18ce536ffffa4f97f'), 'eno': 3.0, 'ename': 'shruti', 'deptno': 10.0}, {'_id': ObjectId('625845218ce536ffffa4f980'), 'eno': 4.0, 'ename': 'Laxman', 'deptno': 20.0}, {'_id': ObjectId('6258454a8ce536ffffa4f981'), 'eno': 5.0, 'ename': 'sham', 'deptno': 20.0}]

##### one record updated #####
{'_id': ObjectId('6258447d8ce536ffffa4f97d'), 'eno': 1.0, 'ename': 'amar', 'deptno': 10.0}
{'_id': ObjectId('625844aa8ce536ffffa4f97e'), 'eno': 2.0, 'ename': 'arjun', 'deptno': 10.0}
{'_id': ObjectId('625844d18ce536ffffa4f97f'), 'eno': 3.0, 'ename': 'shruti', 'deptno': 10.0}
{'_id': ObjectId('625845218ce536ffffa4f980'), 'eno': 4.0, 'ename': 'Laxman', 'deptno': 20.0}
{'_id': ObjectId('6258454a8ce536ffffa4f981'), 'eno': 5.0, 'ename': 'sham', 'deptno': 20.0}

##### one record inserted #####
{'_id': ObjectId('6258447d8ce536ffffa4f97d'), 'eno': 1.0, 'ename': 'amar', 'deptno': 10.0}
{'_id': ObjectId('625844aa8ce536ffffa4f97e'), 'eno': 2.0, 'ename': 'arjun', 'deptno': 10.0}
{'_id': ObjectId('625844d18ce536ffffa4f97f'), 'eno': 3.0, 'ename': 'shruti', 'deptno': 10.0}
{'_id': ObjectId('625845218ce536ffffa4f980'), 'eno': 4.0, 'ename': 'Laxman', 'deptno': 20.0}
{'_id': ObjectId('6258454a8ce536ffffa4f981'), 'eno': 5.0, 'ename': 'sham', 'deptno': 20.0}
{'_id': ObjectId('62714195bb6ce05dcad2df50'), 'eno': 6, 'name': 'Raj', 'location': 'India'}

##### one record deleted #####
{'_id': ObjectId('6258447d8ce536ffffa4f97d'), 'eno': 1.0, 'ename': 'amar', 'deptno': 10.0}
{'_id': ObjectId('625844aa8ce536ffffa4f97e'), 'eno': 2.0, 'ename': 'arjun', 'deptno': 10.0}
{'_id': ObjectId('625844d18ce536ffffa4f97f'), 'eno': 3.0, 'ename': 'shruti', 'deptno': 10.0}
{'_id': ObjectId('625845218ce536ffffa4f980'), 'eno': 4.0, 'ename': 'Laxman', 'deptno': 20.0}
{'_id': ObjectId('6258454a8ce536ffffa4f981'), 'eno': 5.0, 'ename': 'sham', 'deptno': 20.0}

__By Mazhar Solkar
```

Practical No 5

Aim: Implement SVM classification Technique.

Description :-

SVM is a famous supervised machine learning algorithm used for classification as well as regression algorithms. However, mostly it is preferred for classification algorithms. It basically separates different target classes in a hyperplane in n-dimensional or multidimensional space.

The main motive of the SVM is to create the best decision boundary that can separate two or more classes (with maximum margin) so that we can correctly put new data points in the correct class. Because it chooses extreme vectors or support vectors to create the hyperplane, that's why it is named so.

Methods :-

1. StandardScaler() :-

It is used for feature scaling.

2. SVC(kernel='linear', random_state=0) :-

This method is used for implementing SVM.

3. metrics.accuracy_score(y_test,y_pred) :-

This method is used to check the accuracy score of the model.

Program :-

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4
5  # inputfilename = 'C:\0_MSc_IT_Notes\Big Data Analytics\practicals\svm_practical\social.csv'
6  #make sure to give common slash(forward slash / in path)
7
8  inputfilename = 'C:/0_MSc_IT_Notes/Big Data Analytics/practicals/svm_practical/social.csv'
9  df = pd.read_csv(inputfilename)
10 print(df)
11
12 x = df.iloc[:,[2,3]]
13 y = df.iloc[:,4]
14
15 #Splitting the dataset into the training set and test set
16 from sklearn.model_selection import train_test_split
17 x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=0)
18
19 print("Training data:",x_train)
20 print('*****')
21 print("Testing data:",x_test)
22
23 # Feature scaling
24 from sklearn.preprocessing import StandardScaler
25 ss = StandardScaler()
26 x_train_scaled = ss.fit_transform(x_train)
27 x_test_scaled = ss.fit_transform(x_test)
28
29 #train classifier
30 from sklearn.svm import SVC
31 classifier = SVC(kernel='linear', random_state=0)
32 classifier.fit(x_train_scaled,y_train)
33
34 #predicting the test set results
35 y_pred = classifier.predict(x_test_scaled)
36 print(y_pred)
37 from sklearn import metrics
38 print("accuracy score with linear kernel")
39 print(metrics.accuracy_score(y_test,y_pred))
40 print("__By Mazhar Solkar")
```

Output :-

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Mazhar@DESKTOP-0PG7LTN MINGW64 /c:/0_MSc_IT_Notes/Big Data Analytics/practicals (main)
$ python -u "c:\0_MSc_IT_Notes\Big Data Analytics\practicals\svm_practical\svm_college.py"

  User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510   Male   19             19000           0
1    15810944   Male   35             20000           0
2    15668575  Female   26             43000           0
3    15603246  Female   27             57000           0
4    15804002   Male   19             76000           0
..      ...      ...      ...      ...
395  15691863  Female   46             41000           1
396  15706071   Male   51             23000           1
397  15654296  Female   50             20000           1
398  15755018   Male   36             33000           0
399  15594041  Female   49             36000           1

[400 rows x 5 columns]
Training data:      Age  EstimatedSalary
250    44             39000
63     32            120000
312    38             50000
159    32            135000
283    52             21000
..      ...      ...
323    48             30000
192    29             43000
117    36             52000
47     27             54000
172    26            118000

[300 rows x 2 columns]
*****
Testing data:      Age  EstimatedSalary
132    30             87000
309    38             50000
341    35             75000
196    30             79000
246    35             50000
..      ...      ...
146    27             96000
135    23             63000
390    48             33000
264    48             90000
364    42            104000

[100 rows x 2 columns]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 0
 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1]
accuracy score with linear kernel
0.88
__By Mazhar Solkar
```

Practical No 6

Aim: Implement Decision Tree classification Technique.

Description :-

Decision Tree is a supervised learning method used in data mining for classification and regression methods. It is a tree that helps us in decision-making purposes. The decision tree creates classification or regression models as a tree structure. It separates a data set into smaller subsets, and at the same time, the decision tree is steadily developed. The final tree is a tree with the decision nodes and leaf nodes. A decision node has at least two branches. The leaf nodes show a classification or decision. We can't accomplish more split on leaf nodes- The uppermost decision node in a tree that relates to the best predictor called the root node. Decision trees can deal with both categorical and numerical data.

Methods :-

1. MinMaxScaler() :-

This method is used for feature scaling.

2. DecisionTreeClassifier() :-

This method is used to implement decision tree

Program :-

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

#read dataset
df=pd.read_csv('C:/0_MSc_IT_Notes/Big Data Analytics/practicals/decision_tree_practical/social.csv')
print(df)

#choose independent(input) and dependent(ouput) variables
x = df.iloc[:,[2,3]]    #x=df[['Age','EstimatedSalary']]
y = df.iloc[:,4]        #y=df['Purchased']

#split dataset into x_train x_test y_train y_test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)
print(x_train.shape,y_train.shape,x_test.shape,y_test.shape)

# feature scaling
from sklearn.preprocessing import MinMaxScaler
ss = MinMaxScaler()
ss.fit(x_train)
x_train_scaled = ss.transform(x_train)
ss.fit(x_test)
x_test_scaled = ss.transform(x_test)

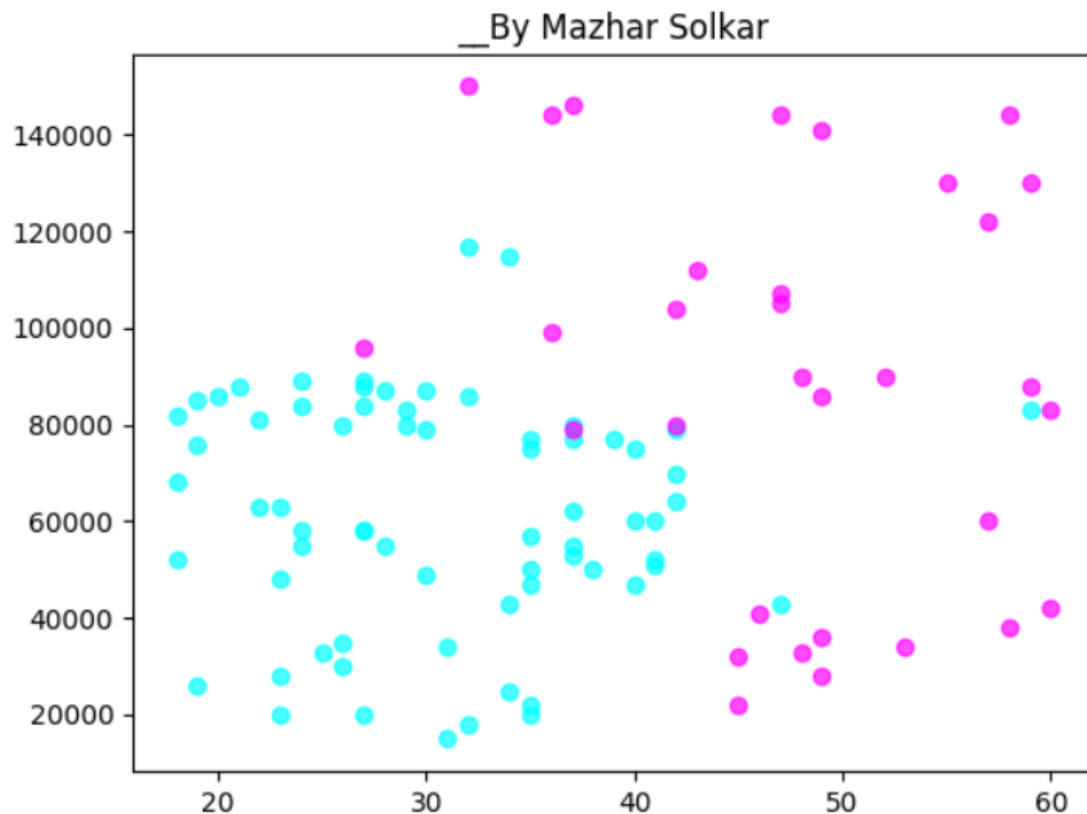
#implement decision tree
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()
classifier.fit(x_train_scaled,y_train)
y_predict = classifier.predict(x_test_scaled)

#accuracy score
print("\naccuracy score")
print(classifier.score(x_test_scaled,y_test))

#plot the graph
plt.scatter(x_test[y_test==0]['Age'],x_test[y_test==0]['EstimatedSalary'],c='cyan',alpha=0.7)
#plotting the scatter plot, c is color alpha is for transparency y_test==0 indicates product not purchased
plt.scatter(x_test[y_test==1]['Age'],x_test[y_test==1]['EstimatedSalary'],c='magenta',alpha=0.7)
plt.title("__By Mazhar Solkar")
plt.show()
print("\n__By Mazhar Solkar")
```

Output :-



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Mazhar@DESKTOP-0PG7LTN MINGW64 /c:/_MSc_IT_Notes/Big Data Analytics/practicals (main)
$ python -u "c:/_MSc_IT_Notes/Big Data Analytics/practicals/decision_tree_practical/decision_tree_college.py"
  User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510   Male   19             19000         0
1    15810944   Male   35             20000         0
2    15668575  Female   26             43000         0
3    15603246  Female   27             57000         0
4    15804002   Male   19             76000         0
..      ...      ...      ...          ...      ...
395  15691863  Female   46             41000         1
396  15706071   Male   51             23000         1
397  15654296  Female   50             20000         1
398  15755018   Male   36             33000         0
399  15594041  Female   49             36000         1

[400 rows x 5 columns]
(300, 2) (300,) (100, 2) (100,)

accuracy score
0.91

__By Mazhar Solkar
```