LINEAR REGRESSION

```
import pandas as pd
from pandas import read_csv
import io
df1 = read_csv("/content/Salary_dataset.csv")

arrx = list(df1['YearsExperience'])
arry = list(df1['Salary'])
print(arrx)
print(arry)

xbar = sum(arrx)/len(arrx)
ybar = sum(arry)/len(arry)

p = [x - xbar for x in arrx]
q = [y - ybar for y in arry]

psq = [x*x for x in p]

pq = []
for a in range(0,len(arrx)):
  pq.append(p[a]*q[a])

M = sum(pq)/sum(psq)

C = ybar - (M*xbar)

df1 = pd.DataFrame()
df1["X"] = arrx
df1["Y"] = arry
df1["P"] = p
df1["Q"] = q
df1["PSQ"] = psq
df1["PQ"] = pq

print(df1)
print("\n Xbar =",xbar," and Ybar = ",ybar)
print("\n Slope(m)=",M)

x = float(input("\n Enter X to predict Y:"))
y = (M*x)+C

print("Predicting Y when X is ",x," = ",y)
```

POLYNOMIAL REGRESSION

```
import numpy as np
import matplotlib.pyplot as plt
x =
np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,2
8,29,30])
y =
np.array([100,120,150,141,153,175,125,123,125,135,134,136,121,99,120,121,140,152,16
3,172,136,182,121,162,142,153,162,172,153,100])
plt.scatter(x, y)
a = np.poly1d(np.polyfit(x,y,3))
b = np.linspace(1,30)
plt.plot(b, a(b))
plt.show()
```

LOGISTIC REGRESSION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression as lr
from sklearn import metrics

df= pd.read_csv("/content/diabetes.csv")
df.head()

features_cols
=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Diabetes
PedigreeFunction']
x= df[features_cols]
y= df.Outcome

x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.57)
Lreg=lr()
Lreg.fit(x_train,y_train)
yprediction=Lreg.predict(x_test)
train_acc = Lreg.score(x_test,y_test)
print("the accuracy for Testing Set is {}".format(train_acc*100))
df2= pd.DataFrame(x_test)
df2["diabetes"]= yprediction;
print("Predicted dataset: \n",df2)
```

HAC

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

data = list(zip(x, y))
```

```
linkage_data = linkage(data, method='ward', metric='euclidean')
dendrogram(linkage_data)

plt.show()
```

DBSCAN

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

import numpy as np
from scipy.spatial import distance_matrix
x = np.array([3, 4, 5, 7, 8, 5, 8, 9, 3, 3, 2, 5])
y = np.array([7, 6, 5, 5, 4, 1, 3, 3, 7, 5, 4, 9])

data = np.array(list(zip(x,y)))
data

plt.scatter(x,y)
plt.title("Points in the from of Graph")
plt.xlabel('x')
plt.ylabel('y')
plt.show()

dbscan = DBSCAN(eps=1.9,min_samples=4).fit(data)
labels = dbscan.labels_

plt.scatter(data[:,0],data[:,1],c=labels,cmap='rainbow')
plt.title("DBSCAN")
plt.xlabel("x")
plt.ylabel("y")
plt.show()

data = np.column_stack((x, y))
Distance_Matrix = distance_matrix(data, data)
print(Distance_Matrix)
```

DECISION TREE

```
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import io

df=pd.read_csv('/content/drug200.csv')
df

d = {'HIGH': 0, 'NORMAL': 1}
```

```
df['Cholesterol'] = df['Cholesterol'].map(d)
d = {'HIGH': 0, 'LOW': 1, 'NORMAL': 2}
df['BP'] = df['BP'].map(d)
d = {'F': 0, 'M': 1}
df['Sex'] = df['Sex'].map(d)

features = ['Age','Sex','BP','Cholesterol','Na_to_K']
X = df[features]
y = df['Drug']

dtree = DecisionTreeClassifier().fit(X, y)

tree.plot_tree(dtree, feature_names=features)
```

RANDOM FOREST

```
import pandas as pd
from sklearn import datasets
iris = datasets.load_iris()
print(iris.target_names)
print(iris.feature_names)

print(iris.data[0:5])
print(iris.target)
data = pd.DataFrame({
'sepal length':iris.data[:,0],
'sepal width':iris.data[:,1],
'petal length':iris.data[:,2],
'petal width':iris.data[:,3],
'species':iris.target
})
data.head

from sklearn.model_selection import train_test_split
x=data[['sepal length','sepal width','petal length','petal width']]
y=data['species']

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)

from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)

from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))

import pandas as pd
feature_imp = pd.Series(clf.feature_importances_,index=iris.feature_names)
feature_imp
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('features')
plt.title("Visualising Important Features")
plt.legend()
plt.show()

from sklearn.model_selection import train_test_split
x=data[['sepal length','petal length','petal width']]
y=data['species']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))
```