Exp 1

```
import numpy as np
x=np.array([[1,2,3],[4,5,6]])
print("x:\n{}".format(x))

from scipy import sparse
eye=np.eye(4)
print("Numpy array :\n{}".format(eye))

sparse_matrix=sparse.csr_matrix(eye)
print("\n Scipy sparse CSR matrix:\n{}".format(sparse_matrix))

%matplotlib inline
import matplotlib.pyplot as plt
x=np.linspace(-10,10,100)
y=np.sin(x)
plt.plot(x,y,marker="x")
plt.show()

import pandas as pd
data = {'Name':["John","Anna","Peter","Linda"],
      'Location':["New York","Paris","Berlin","London"],
      'Age':[24,13,53,33]
}
data_pandas=pd.DataFrame(data)
display(data_pandas)

display(data_pandas[data_pandas.Age>30])
```

Exp 2

```
import numpy as nm
import matplotlib.pyplot as mpt
import pandas as pd

data_set= pd.read_csv("C:\\Users\\Downloads\\Dataset.csv")

print(data_set)

X = data_set.iloc[:, :-1]  # All columns except the last one
y = data_set.iloc[:, -1]   # The last column

print("Independent Variables (X):")
print(X)
print("\nDependent Variable (y):")
print(y)

import numpy as np
from sklearn.impute import SimpleImputer
# Separate numeric and non-numeric data
numeric_columns = ['Age', 'Salary']
categorical_columns = ['Country']
# Impute numeric columns with mean
numeric_imputer = SimpleImputer(strategy='mean')
data_set[numeric_columns] = numeric_imputer.fit_transform(data_set[numeric_columns])
# Impute categorical columns with most frequent value
categorical_imputer = SimpleImputer(strategy='most_frequent')
data_set[categorical_columns] = categorical_imputer.fit_transform(data_set[categorical_columns])
print(data_set)

from sklearn.model_selection import train_test_split
```

```
# Assume 'X' is the matrix of features (independent variables)
# and 'y' is the target vector (dependent variable)
X = data_set[['Country', 'Age', 'Salary']]
y = data_set['Purchased']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training Features:", X_train)
print("Testing Features:", X_test)
print("Training Labels:", y_train)
print("Testing Labels:", y_test)
```

Exp 3 – linear regression

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

data_set= pd.read_csv("C:\\Users\\Downloads\\Salary_Data.csv")

print(data_set)
x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values
print (x)
print (y)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_state=0)

from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)

from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)

y_pred= regressor.predict(x_test)

plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Train set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
```

```
plt.show()


plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

#regression coefficients:
# regression coefficients
print('Coefficients: ', regressor.coef_)
# regression intercept
regressor.intercept_
#Predicting value of unseen data
regressor.predict(np.array([15]).reshape((-1, 1)))
```

Exp 3 – Mutiple Regression

```
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline

data = pd.read_csv("C:\\Users\\omkar\\Downloads\\Advertising.csv")
data.head()

sns.pairplot(data, x_vars=['TV','Radio','Newspaper'], y_vars='Sales', height=7, aspect=0.7, kind='reg')
feature_cols = ['TV', 'Radio', 'Newspaper']

X = data[feature_cols]
X.head()
y = data['Sales']
y.head()
print(type(y))
print(y.shape)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)

from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
print(linreg.intercept_)
```

```
print(linreg.coef_)

list(zip(feature_cols, linreg.coef_))
y_pred = linreg.predict(X_test)

from sklearn import metrics
print(metrics.mean_absolute_error(y_test, y_pred))

from sklearn import metrics
print(metrics.mean_squared_error(y_test, y_pred))

import numpy as np
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Exp 4 – Logistic Regression

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

data_set= pd.read_csv("C:\\Users\\Downloads\\suv_data.csv")

x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

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)

plt.scatter(x_train[:,0], x_train[:,1], c=y_train, cmap='coolwarm')
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("Training Data Scatter Plot")
plt.show()

from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)

from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)

from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

```
y_pred= classifier.predict(x_test)

from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
print(cm)

import matplotlib.pyplot as plt
import seaborn as sns

# Create a heatmap for the confusion matrix
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, cmap="Blues", fmt='d')

plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

import numpy as np
import matplotlib.pyplot as plt
# Generate a grid of points
x_min, x_max = x_train[:, 0].min() - 1, x_train[:, 0].max() + 1
y_min, y_max = x_train[:, 1].min() - 1, x_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                     np.arange(y_min, y_max, 0.01))
# Predict for each point in the grid
Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# Plot the decision boundary
plt.contourf(xx, yy, Z, alpha=0.3, cmap='coolwarm')
```

```
plt.scatter(x_train[:, 0], x_train[:, 1], c=y_train, cmap='coolwarm', edgecolors='k')
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("Logistic Regression Decision Boundary")
plt.show()
```

Exp 5 – Decision Tree

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

df= pd.read_csv("C:\\Users\\Desktop\\ML practical 5\\diabetes_csv.csv")

df

x= df.iloc[:, :-1].values
y= df.iloc[:, 8].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split (x, y, test_size=0.3)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

from sklearn import tree
classifier = tree.DecisionTreeClassifier()
classifier.fit(x_train,y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Exp 5 – KNN

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

df= pd.read_csv("C:\\User\\Desktop\\ML practical 5\\diabetes_csv.csv")

df

x= df.iloc[:, :-1].values
y= df.iloc[:, 8].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split (x, y, test_size=0.3)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Exp 5 – Naïve Bayes

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

df= pd.read_csv("C:\\Users\\Desktop\\ML practical 5\\diabetes_csv.csv")

df

x= df.iloc[:, :-1].values
y= df.iloc[:, 8].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split (x, y, test_size=0.3)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train,y_train)

y_pred = classifier.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Exp 5 – SVM

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

```
import pandas as pd

df= pd.read_csv("C:\\Users\\Desktop\\ML practical 5\\diabetes_csv.csv")

df

x= df.iloc[:, :-1].values
y= df.iloc[:, 8].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split (x, y, test_size=0.3)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

from sklearn import svm
classifier = svm.SVC()
classifier.fit(x_train,y_train)

y_pred = classifier.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Exp 6 – Hierarchical Clustering

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data_set= pd.read_csv("C:\\Users\\Desktop\\Academic\\TY2\\Machine Learning\\practical 6\\Mall_Customers.csv")


x= data_set.iloc[:, [3,4]].values


import scipy.cluster.hierarchy as shc

dendro = shc.dendrogram(shc.linkage(x, method="ward"))

mtp.title("Dendrogrma Plot")

mtp.ylabel("Euclidean Distances")

mtp.xlabel("Customers")

mtp.show()


from sklearn.cluster import AgglomerativeClustering

hc= AgglomerativeClustering(n_clusters=5, metric="euclidean", linkage='ward')

y_pred= hc.fit_predict(x)

mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')

mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s = 100, c = 'green', label = 'Cluster 2')

mtp.scatter(x[y_pred== 2, 0], x[y_pred == 2, 1], s = 100, c = 'red', label = 'Cluster 3')

mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

mtp.title('Clusters of customers')

mtp.xlabel('Annual Income (k$)')

mtp.ylabel('Spending Score (1-100)')

mtp.legend()

mtp.show()
```

Exp 6 – KNN Clustering

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data_set= pd.read_csv("C:\\Users\\Desktop\\Academic\\TY2\\Machine Learning\\practical 6\\Mall_Customers.csv")

x= data_set.iloc[:, [3,4]].values

x

from sklearn.cluster import KMeans

wcss_list= []  #Initializing the list for the values of WCSS


#Using for loop for iterations from 1 to 10.

for i in range(1, 11):

   kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)

   kmeans.fit(x)

   wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list)

mtp.title('The Elobw Method Graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss_list')

mtp.show()


kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)

y_predict= kmeans.fit_predict(x)

y_predict


mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cluster

mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster

mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third cluster
```

```
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cluster

mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster

mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')

mtp.title('Clusters of customers')

mtp.xlabel('Annual Income (k$)')

mtp.ylabel('Spending Score (1-100)')

mtp.legend()

mtp.show()
```

Exp 7 – Random Forest

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

data_set= pd.read_csv("C:\\Users\\Downloads\\ML practical 7\\suv_data.csv")

x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
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)

from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)

from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)

y_pred= classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
print(cm)
```