Practical No 1 :

```
dataframe = {
   'Stock_Id':[1,2,3,4,5,6,7,8,9,10],
   'Stock_Name':['ABC','PQR','XYZ','MNO','TCS','INFY','RELI','HDFC','SBIN','ITC'],
   'Open':[120,50,170,200,250,300,250,400,450,500],
   'Close': [130,60,180,210,260,310,260,410,460,510]
}
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.DataFrame(dataframe)
# data = pd.read_csv('file_name.csv)
print("shape", data.shape)
print("describe", data.describe)
print("null count", data.isnull().sum())
print("duplicated", data.duplicated().sum())
print("datatype \n", data.dtypes)

plt.plot(data['Open'],data['Close'])
plt.show();
sns.heatmap(data.drop(columns=['Stock_Name']).corr(), annot=True)
plt.show()

sns.boxplot(data['Open'])
plt.show()
```

Practical No : 2

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, mean_absolute_error, accuracy_score, r2_score,confusion_matrix

x = data[['Open']]

y = data['Close']

x_train, x_test, y_train,y_test = train_test_split(x, y , test_size = 0.2,random_state=42)

model = LinearRegression()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

mae = mean_absolute_error(y_test,y_pred)

print(mae)

mse = mean_squared_error(y_test,y_pred)

print(mse)

r2 = r2_score(y_test,y_pred)

print(r2)

plt.plot(x_test,y_pred,color='red')


plt.xlabel('Open')

plt.ylabel('Close')

plt.title('Linear Regression')

plt.show()
```

Practical 3

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score,
precision_score, f1_score
from sklearn.metrics import roc_curve


df = {
   'Age':[35,30,35,20,45,50,55,70,65,40],
   'Salary':[50000,60000,70000,80000,90000,100000,10000, 15000, 20000, 25000],
   'Purchased':[0,0,1,1,0,1,0,1,0,1]
}
data = pd.DataFrame(df)
print(data)
x = data[['Age','Salary']]
y = data['Purchased']
x_train, x_test, y_train, y_test = train_test_split(x, y ,test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(x_train, y_train)
z = model.predict(x_test)
print("accuracy", accuracy_score(y_test, z))
print("confusion matrix", confusion_matrix(y_test, z))
print("recall", recall_score(y_test, z))
print("precision", precision_score(y_test, z))
sns.heatmap(confusion_matrix(y_test, z))
plt.show()
fpr , tpr,threshold  = roc_curve(y_test, model.predict_proba(x_test)[:,1])
plt.plot(fpr, tpr)
plt.show()
```

Practical 4

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
data = {
   'Hours_Studied': [2,3,4,6,7,8,1,9,10,5,4,6,8,2,3,7],
   'Attendance': [60,65,70,75,80,85,50,90,95,70,68,77,84,55,60,82],
   'Assignments_Submitted': [2,3,4,4,5,6,1,6,6,3,3,4,6,2,2,5],
   'Passed': [0,0,0,1,1,1,0,1,1,0,0,1,1,0,0,1]
}
df = pd.DataFrame(data)
X = df[['Hours_Studied', 'Attendance', 'Assignments_Submitted']]
y = df['Passed']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
print("decision tRee")
print('accuracy', accuracy_score(y_test, y_predict))
print('precision', precision_score(y_test, y_predict))
print('recall', recall_score(y_test, y_predict))
print('f1 score', f1_score(y_test, y_predict))
model = SVC()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
print('svm')
print('accuracy', accuracy_score(y_test, y_predict))
print('precision', precision_score(y_test, y_predict))
```

```
print('recall',recall_score(y_test, y_predict))
print('f1 score', f1_score(y_test, y_predict))
```

Practical 5

```
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Create dataset
data = {
    'Hours_Studied': [
        1,2,2,3,3,4,5,6,7,8,   # Group 1: lower performers
        9,10,11,12,13,14,15,16,17,18,  # Group 2: average performers
        19,20,21,22,23,24,25,26,27,28  # Group 3: top performers
    ],
    'Attendance': [
        40,45,47,50,52,55,58,60,62,65,   # around 40–65
        75,78,80,82,84,86,87,88,89,90,   # around 75–90
        92,93,94,95,96,97,98,99,99,100   # around 92–100
    ]
}

df = pd.DataFrame(data)
print("Sample data:")
print(df.head())
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(df)
```

```
dbscan=DBSCAN(eps=0.5,min_samples=3)
labels = dbscan.fit_predict(X_Scaled)

df['Cluster']=labels
print("\nCluster labels assigned by DBSCAN:")
print(df)
plt.figure(figsize=(8,6))
plt.scatter(df['Hours_Studied'], df['Attendance'], c=df['Cluster'], cmap='Accent')
plt.title('DBSCAN Clustering of Students')
plt.xlabel('Hours Studied')
plt.ylabel('Attendance')
plt.colorbar(label='Cluster ID')
plt.show()
```

Practical 6:

```
from sklearn.ensemble import RandomForestClassifier , AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
classification_report
df = {
   'Age':[35,30,35,20,45,50,55,70,65,40],
   'Salary':[50000,60000,70000,80000,90000,100000,10000, 15000, 20000, 25000],
   'Purchased':[0,0,1,1,0,1,0,1,0,1]
}
data = pd.DataFrame(df)
print(data)
X = data[['Age', 'Salary']] # Independent Variables
y = data['Purchased'] # Dependent Variable
X_train, X_test, y_train, y_test = train_test_split(
```

```
 X, y, test_size=0.3, random_state=42
)
model = RandomForestClassifier(n_estimators=10, random_state = 42)
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
print("\n\n********** BAGGING (RANDOM FOREST) PERFORMANCE **********")
print("Accuracy:", accuracy_score(y_test, y_predict))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_predict))

boost_model = AdaBoostClassifier(
 n_estimators=50, # number of weak learners
 learning_rate=1.0

)

boost_model.fit(X_train, y_train)
boost_pred = boost_model.predict(X_test)
print("\n\n********** BOOSTING (ADABOOST) PERFORMANCE **********")
print("Accuracy:", accuracy_score(y_test, boost_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, boost_pred))
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict))
```