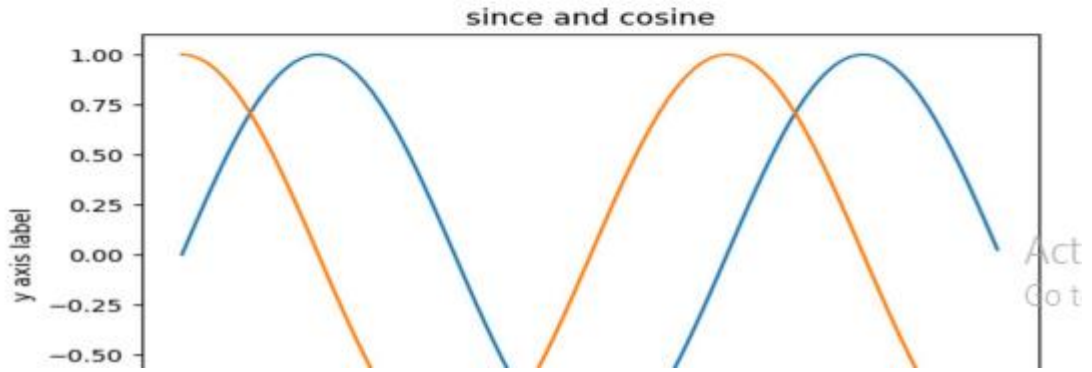
SET-1

1.
```
import numpy as np
a = np.array([1, 2, 3]) # Create a rank 1 array
print("One dimensional array a = ",a)
b = np.array([[1,2,3],[4,5,6]])
print("Two dimensional array b = ",b)
print("Size of the array: ",a.shape)
print("Element at indices 0,1,2 : ",a[0], a[1], a[2])
a[0] = 5 # Change an element of the array
print("Array after changing the element at index 0 : ",a)
a = np.zeros((2,2)) # Create an array of all zeros
print("An array of all zeros : ",a)
b = np.ones((1,2)) # Create an array of all ones
print("An array of all ones : ",b)
c = np.full((2,2), 7) # Create a constant array
print("A constant array : ",c)
d = np.eye (2) # Create a 2x2 identity matrix
print("A 2*2 identity matrix : ",d)
e = np.random.random((2,2)) # Create an array filled with random values
print("An array with random values : ",e)
```

2.
```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0,3*np.pi,0.1)
y_sin=np.sin(x)
y_cos=np.cos(x)
plt.plot(x,y_sin)
plt.plot(x,y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('since and cosine')
plt.legend(['since','cosine'])
plt.show()
```

OUTPUT



3. You have two NumPy arrays, arr1 and arr2, containing the following data:

arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([6, 7, 8, 9, 10])

Write NumPy code to perform the following operations:

1. Add arr1 and arr2 to create a new array called result_add.
2. Multiply arr1 and arr2 to create a new array called result_multiply.
3. Calculate the mean of result_add.
4. Find the maximum value in result_multiply.

ARRAY

```
import numpy as np

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([6, 7, 8, 9, 10])

result_add=arr1+arr2

print("sum:",result_add)

result_multy=arr1*arr2

print("mul",result_multy)

print("mean",np.mean(result_add))

print("maxval",np.max(result_multy))
```

4.create the dataset
a. create dataframe
b.show top 5 rows
c remove multiple columns at once
d. rename two of the columns by using the 'rename' method

DATASETS

```
import pandas as pd

# a. Create DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],
    'Age': [25, 30, 22, 35, 28],
    'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Boston'],
    'Salary': [60000, 80000, 55000, 70000, 75000],
    'Department': ['HR', 'IT', 'Marketing', 'Finance', 'Sales']
}

df = pd.DataFrame(data)
```

```
# b. Show top 5 rows
print("Top 5 rows:")
print(df.head())

# c. Remove multiple columns at once
columns_to_remove = ['City', 'Department']
df = df.drop(columns=columns_to_remove)

# d. Rename two columns using the 'rename' method
column_rename_mapping = {
    'Name': 'Full Name',
    'Age': 'Years Old'
}

df = df.rename(columns=column_rename_mapping)

# Display the modified DataFrame
print("\nDataFrame after removing columns and renaming:")
print(df)
```

5. Write a program to implement KNN algorithm using iris data Set

```
from sklearn.datasets import  load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
c_knn=KNeighborsClassifier(n_neighbors=3)
c_knn.fit(x_train,y_train)
y_pred=c_knn.predict(x_test)
print("accuracy",metrics.accuracy_score(y_test,y_pred))
sample=[[2,2,2,2]]
pred=c_knn.predict(sample)
pred_v=[iris.target_names[i] for i in pred]
print(pred_v)
```

6. Write a program to implement decision tree algorithm using the given data set

```
from sklearn.datasets import load_iris
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

iris = load_iris()
x = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)

clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)

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

plt.figure(figsize=(15, 15))
plot_tree(clf, fontsize=10, filled=True, rounded=True, class_names=list(iris.target_names),
feature_names=iris.feature_names)
plt.show()
```

7. Write a program to implement Linear Regression using appropriate data set.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets,linear_model
from sklearn.metrics import mean_squared_error,r2_score
df =datasets.load_diabetes()
df['feature_names']
diabetes_X,diabetes_y=datasets.load_diabetes(return_X_y=True)
diabetes_X.shape
diabetes_y.shape
diabetes_X=diabetes_X[:,np.newaxis,2]
diabetes_X.shape
diabetes_X_train=diabetes_X[:-20]
diabetes_X_test=diabetes_X[-20:]
```

```
diabetes_y_train=diabetes_y[:-20]
diabetes_y_test=diabetes_y[-20:]
regr=linear_model.LinearRegression()
regr.fit(diabetes_X_train,diabetes_y_train)
diabetes_y_pred=regr.predict(diabetes_X_test)
print("coefficients :\n",regr.coef_)
print("meansquareerror:%2f"%mean_squared_error(diabetes_y_test,diabetes_y_pred))
print("coefficient of determination:%2f"%r2_score(diabetes_y_test,diabetes_y_pred))
plt.scatter(diabetes_X_test,diabetes_y_pred,color="blue",linewidth=3)
plt.ylabel("diabetes progression")
plt.xticks(())
plt.yticks(())
plt.show(())
```

8. Write a program to implement naive bayes classification using iris dataset

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X,y=load_iris(return_X_y=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.5,random_state=0)
gnb=GaussianNB()
y_pred=gnb.fit(X_train,y_train).predict(X_test)
print(y_pred)
x_new=[[5,5,4,4]]
y_new=gnb.fit(X_train,y_train).predict(x_new)
print("predicted output for [[5,5,4,4]]:",y_new)
print("Naive bayes score:",gnb.score(X_test,y_test))
```