Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Prerequisite: 01

Aim: Basics of Python.

Source Code:

 # 1) Area of Circle

```
radius = int(input("Enter the radius of circle: "))
area = 3.14 * radius * radius
print("Area of circle is:", area)
```

Output:

```
Enter the radius of circle:  5
Area of circle is: 78.5
```

 # 2) Circumference of Circle

```
radius = int(input("Enter the radius of circle: "))
circumference = 2 * 3.14 * radius
print("Circumference of circle is:", circumference)
```

Output:

```
Enter the radius of circle:  5
Circumference of circle is: 31.400000000000002
```

 # 3) Volume of Cylinder

```
radius = int(input("Enter the radius of cylinder: "))
height = int(input("Enter the height of cylinder: "))
volume = 3.14 * radius * radius * height
print("Volume of cylinder is:", volume)
```

Output:

```
Enter the radius of cylinder:  6
Enter the height of cylinder:  8
Volume of cylinder is: 904.3199999999999
```

 # 4) Check if Number is Greater than 50

```
num1 = int(input("Enter the number: "))
if num1 > 50:
    print("Provided number is greater than 50")
else:
    print("Provided number is smaller than or equal to 50")
```

Output:

```
Enter the number:  78
Provided number is greater than 50
```

 # 5) Student Marks, Average, Percentage and Grade

```
sub1 = int(input("Enter marks of subject 1: "))
sub2 = int(input("Enter marks of subject 2: "))
sub3 = int(input("Enter marks of subject 3: "))
sub4 = int(input("Enter marks of subject 4: "))
sub5 = int(input("Enter marks of subject 5: "))

total = sub1 + sub2 + sub3 + sub4 + sub5
avg = total / 5
percentage = (total / 500) * 100

print("Your total is:", total)
print("Your Average is:", avg)
print("Your percentage is:", percentage)

if percentage > 90:
```

```
    print("Grade is A")
elif percentage > 80:
    print("Grade is B")
elif percentage > 70:
    print("Grade is C")
elif percentage > 60:
    print("Grade is D")
elif percentage > 50:
    print("Grade is E")
elif percentage < 35:
    print("Grade is F")
else:
    print("Grade is Pass")
```

Output:

```
Enter marks of subject 1:  67
Enter marks of subject 2:  78
Enter marks of subject 3:  98
Enter marks of subject 4:  90
Enter marks of subject 5:  67
Your total is: 400
Your Average is: 80.0
Your percentage is: 80.0
Grade is C
```

 # 6) Print Name 10 Times

```
for i in range(10):
    print("Sayali")
```

Output:

```
Sayali
Sayali
Sayali
Sayali
Sayali
Sayali
Sayali
Sayali
Sayali
Sayali
```

 # 7) Check Even or Odd Number

```
num = int(input("Enter the number: "))
if num % 2 == 0:
    print("Number is even")
else:
    print("Number is odd")
```

Output:

```
Enter the number:  9
Number is odd
```

 # 8) Print Even and Odd Numbers from 0 to 19

```
for i in range(20):
    if i % 2 == 0:
        print(i, "is even")
    else:
        print(i, "is odd")
```

Output:

```
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
11 is odd
12 is even
13 is odd
14 is even
15 is odd
16 is even
17 is odd
18 is even
19 is odd
```

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Prerequisite: 02

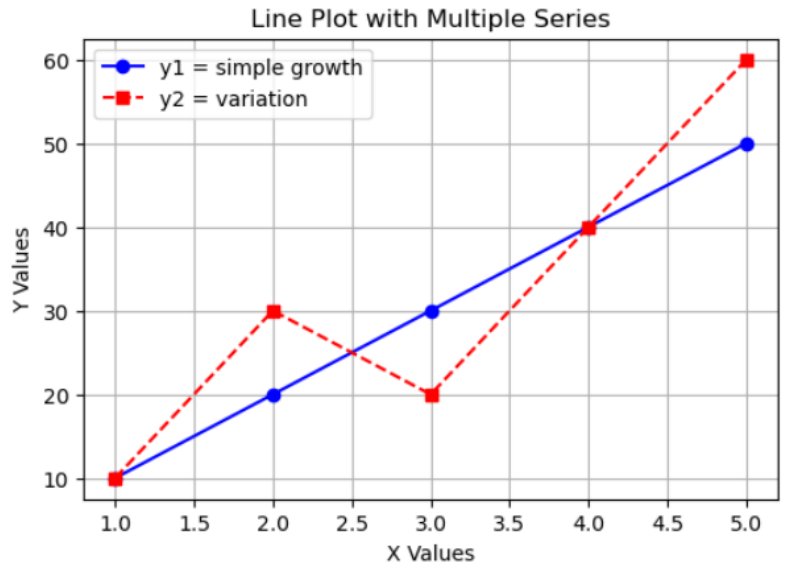
Aim: Ploting Libraries.

Source Code with Output:

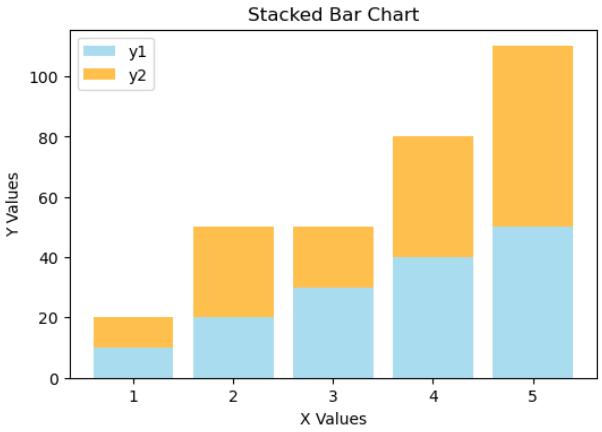
```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Sample Data
x = [1, 2, 3, 4, 5]
y1 = [10, 20, 30, 40, 50]
y2 = [10, 30, 20, 40, 60]
```

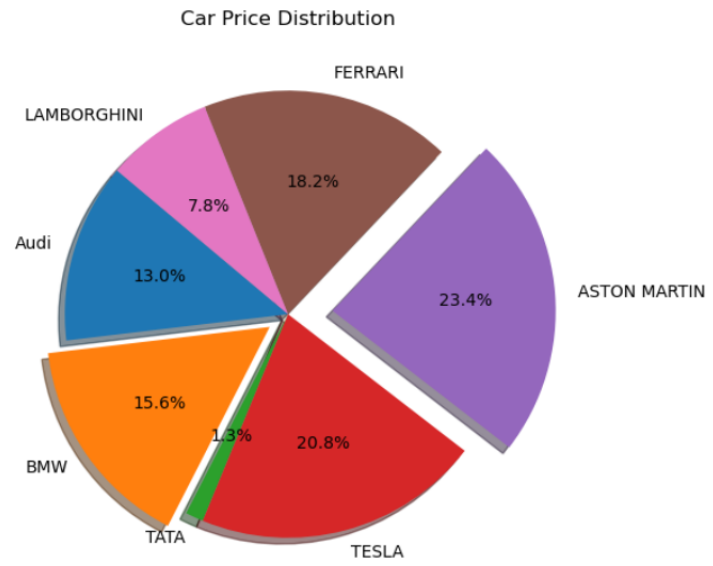
```
# Line Plot
plt.figure(figsize=(6, 4))
plt.plot(x, y1, label="y1 = simple growth", marker='o', linestyle='-', color="blue")
plt.plot(x, y2, label="y2 = variation", marker='s', linestyle='--', color="red")
plt.xlabel("X Values")
plt.ylabel("Y Values")
plt.title("Line Plot with Multiple Series")
plt.legend()
plt.grid(True)
plt.savefig("line_plot.png")
plt.show()
```



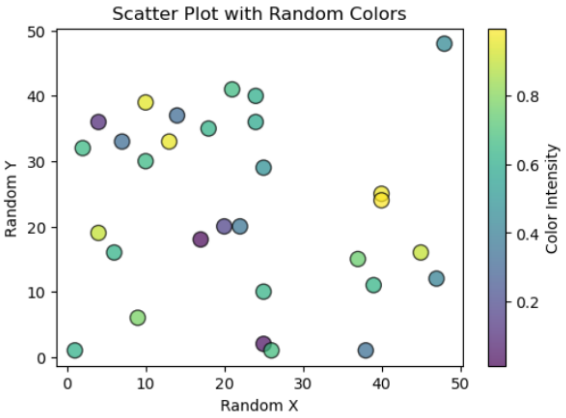
```python
#  Bar Chart
plt.figure(figsize=(6, 4))
plt.bar(x, y1, label="y1", color="skyblue", alpha=0.7)
plt.bar(x, y2, label="y2", color="orange", alpha=0.7, bottom=y1)  # stacked bars
plt.xlabel("X Values")
plt.ylabel("Y Values")
plt.title("Stacked Bar Chart")
plt.legend()
plt.savefig("bar_chart.png")
plt.show()
```



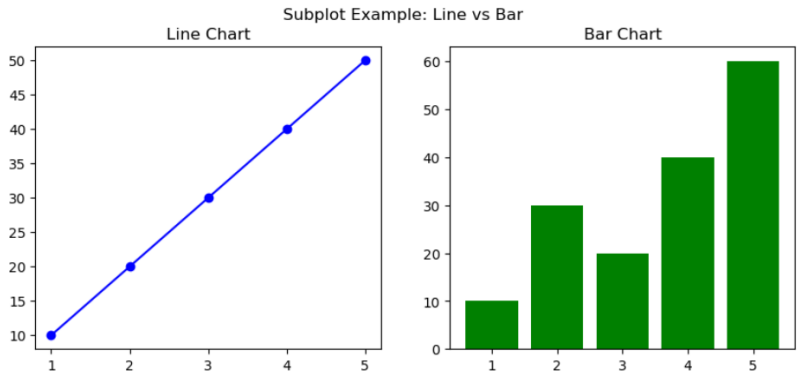

```python
#  Pie Chart
cars = ["Audi", "BMW", "TATA", "TESLA", "ASTON MARTIN", "FERRARI", "LAMBORGHINI"]
price = [50, 60, 5, 80, 90, 70, 30]

plt.figure(figsize=(6, 6))
explode = [0, 0.1, 0, 0, 0.2, 0, 0]  # highlight BMW & Aston Martin
plt.pie(price, labels=cars, autopct='%1.1f%%', startangle=140, explode=explode, shadow=True)
plt.title("Car Price Distribution")
plt.savefig("pie_chart.png")
plt.show()
```



```python
# Scatter Plot
np.random.seed(0)
x_scatter = np.random.randint(1, 50, 30)
y_scatter = np.random.randint(1, 50, 30)
colors = np.random.rand(30)

plt.figure(figsize=(6, 4))
plt.scatter(x_scatter, y_scatter, c=colors, cmap="viridis", s=100, alpha=0.7, edgecolors="black")
plt.xlabel("Random X")
plt.ylabel("Random Y")
plt.title("Scatter Plot with Random Colors")
plt.colorbar(label="Color Intensity")
plt.savefig("scatter_plot.png")
plt.show()
```




```python
# Subplots Comparison
fig, axs = plt.subplots(1, 2, figsize=(10, 4))

axs[0].plot(x, y1, marker='o', color="blue")
axs[0].set_title("Line Chart")

axs[1].bar(x, y2, color="green")
axs[1].set_title("Bar Chart")

plt.suptitle("Subplot Example: Line vs Bar")
plt.savefig("subplots.png")
plt.show()
```



Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Prerequisite: 03

Aim: Data Analysis Using Python on a Cricket Dataset.

Source Code with Output:

```
import pandas as pd
import numpy as np
cric=pd.read_csv("cric_data .csv")
```

```
cric.head()
```

		Sachin Tendulkar	Rahul Dravid	India
0	0	100	78	342
1	1	11	62	191
2	2	8	85	252
3	3	71	24	307
4	4	104	17	229

```
cric.describe()
```

		Sachin Tendulkar	Rahul Dravid	India
count	225.000000	225.000000	225.000000	225.000000
mean	112.000000	39.875556	32.062222	220.795556
std	65.096083	41.324096	30.812156	80.416122
min	0.000000	0.000000	0.000000	0.000000
25%	56.000000	6.000000	7.000000	175.000000
50%	112.000000	27.000000	22.000000	216.000000
75%	168.000000	63.000000	53.000000	273.000000
max	224.000000	186.000000	153.000000	499.000000

```
cric.shape
```

```
(225, 4)
```

```
cric.describe()
```

		Sachin Tendulkar	Rahul Dravid	India
count	225.000000	225.000000	225.000000	225.000000
mean	112.000000	39.875556	32.062222	220.795556
std	65.096083	41.324096	30.812156	80.416122
min	0.000000	0.000000	0.000000	0.000000
25%	56.000000	6.000000	7.000000	175.000000
50%	112.000000	27.000000	22.000000	216.000000
75%	168.000000	63.000000	53.000000	273.000000
max	224.000000	186.000000	153.000000	499.000000

```
cric.info
```

```
<bound method DataFrame.info of           Sachin Tendulkar  Rahul Dravid  India
0      0                 100            78    342
1      1                  11            62    191
2      2                   8            85    252
3      3                  71            24    307
4      4                 104            17    229
..   ...                 ...           ...    ...
220  220                  25            14    101
221  221                 102            50    470
222  222                   0            22    165
223  223                  27             0    192
224  224                  40             0    213

[225 rows x 4 columns]>
```

```
#Sachin Information
sachin=cric.iloc[:,1]
```

```
print("Sachin Analysis:")
print("First 5 records of Sachin:")
sachin.head()
```

```
Sachin Analysis:
First 5 records of Sachin:
0    100
1     11
2      8
3     71
4    104
Name: Sachin Tendulkar, dtype: int64
```

```
print("Maximum Score of Sachin:")
sachin.max()
```

```
Maximum Score of Sachin:
186
```

```
print("Minimum Score of Sachin:")
sachin.min()
```

```
Minimum Score of Sachin:
0
```

```
print("Minimum Score of Sachin:")
sachin.min()
```

```
Minimum Score of Sachin:
0
```

```
print("Average Score of Sachin:")
sachin.mean()
```

```
Average Score of Sachin:
np.float64(39.87555555555556)
```

```
print("No. of Half-centuries of Sachin:")
np.count_nonzero((sachin<100)&(sachin>49))
```

```
No. of Half-centuries of Sachin:
42
```

```
print("No. of Centuries of Sachin:")
np.count_nonzero((sachin>100))
```

```
No. of Centuries of Sachin:
23
```

```
print("Cumulative sum of Sachin Score:")
sachin_cumsum=np.cumsum(sachin)
sachin_cumsum
```

```
Cumulative sum of Sachin Score:
0       100
1       111
2       119
3       190
4       294
       ...
220    8803
221    8905
222    8905
223    8932
224    8972
Name: Sachin Tendulkar, Length: 225, dtype: int64
```

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 01

Aim: On the fruit dataset, compare the performance of Logistic Regression, SVM, KNN on the basis of their accuracy.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
fruits = pd.read_csv('fruit_data.csv')
print(fruits.head())
print(fruits.tail())
print (fruits.describe())
feature_names=['mass','width','height','color_score']
X = fruits[feature_names]
Y = fruits['fruit_label']
X_train, X_test,y_train,y_test=train_test_split(X,Y,random_state=0)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
print('Accuracy of K-NN classifier on training set:{:.2f}'.format(knn.score(X_train,y_train)))
print('Accuracy of K-NN classifier on test set:{:.2f}'.format(knn.score(X_test,y_test)))
svm = SVC()
svm.fit(X_train,y_train)
print('Accuracy of SVM classifier on trainning set:{:.2f}'.format(svm.score(X_train,y_train)))
print('Accuracy of SVM classifier on test set:{:.2f}'.format(svm.score(X_test,y_test)))
```

 # Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print('Accuracy of Logistic Regression on training set: {:.2f}'.format(model.score(X_train,
y_train)))
print('Accuracy of Logistic Regressionr on test set: {:.2f}'.format(model.score(X_test,
y_test)))
classification_report(y_test, predictions)
```

Output:

```
   fruit_label fruit_name fruit_subtype  mass  width  height  color_score
0            1      apple  granny_smith   192    8.4     7.3         0.55
1            1      apple  granny_smith   180    8.0     6.8         0.59
2            1      apple  granny_smith   176    7.4     7.2         0.60
3            2   mandarin      mandarin    86    6.2     4.7         0.80
4            2   mandarin      mandarin    84    6.0     4.6         0.79
    fruit_label fruit_name fruit_subtype  mass  width  height  color_score
54            4      lemon       unknown   116    6.1     8.5         0.71
55            4      lemon       unknown   116    6.3     7.7         0.72
56            4      lemon       unknown   116    5.9     8.1         0.73
57            4      lemon       unknown   152    6.5     8.5         0.72
58            4      lemon       unknown   118    6.1     8.1         0.70
       fruit_label        mass      width     height  color_score
count    59.000000   59.000000  59.000000  59.000000    59.000000
mean      2.542373  163.118644   7.105085   7.693220     0.762881
std       1.208048   55.018832   0.816938   1.361017     0.076857
min       1.000000   76.000000   5.800000   4.000000     0.550000
25%       1.000000  140.000000   6.600000   7.200000     0.720000
50%       3.000000  158.000000   7.200000   7.600000     0.750000
75%       4.000000  177.000000   7.500000   8.200000     0.810000
max       4.000000  362.000000   9.600000  10.500000     0.930000
Accuracy of K-NN classifier on training set:0.95
Accuracy of K-NN classifier on test set:1.00
Accuracy of SVM classifier on trainning set:0.91
Accuracy of SVM classifier on test set:0.80
Accuracy of Logistic Regression on training set: 0.75
Accuracy of Logistic Regressionr on test set: 0.47
```

```
'              precision    recall  f1-score   support\n\n           1       0.36      1.00      0.53         4\n           2       0.00      0.00      0.0
0         1\n           3       1.00      0.12      0.22         8\n           4       0.67      1.00      0.80         2\n\n    accuracy
0.47        15\n   macro avg       0.51      0.53      0.39        15\nweighted avg       0.72      0.47      0.37        15\n'
```

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 02

Aim: On the iris dataset, perform KNN algorithm and discuss result.

Source Code:

```
# Import libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load iris dataset
iris = load_iris()
X = iris.data[:, :2]   # Only sepal length and sepal width
y = iris.target

# Split dataset (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Apply KNN with k=5
knn = KNeighborsClassifier(n_neighbors=5)
```

```
knn.fit(X_train, y_train)


# Predictions
y_pred = knn.predict(X_test)


# Evaluation
print("✅ Accuracy (sepal only):", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred,
target_names=iris.target_names))
```

Output:

```
✅ Accuracy (sepal only): 0.7666666666666667

Confusion Matrix:
 [[10  0  0]
 [ 0  6  3]
 [ 0  4  7]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.60      0.67      0.63         9
   virginica       0.70      0.64      0.67        11

    accuracy                           0.77        30
   macro avg       0.77      0.77      0.77        30
weighted avg       0.77      0.77      0.77        30
```

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 03

Aim: Implement apriori algorithm on Online retail dataset and discuss results.

Source Code:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
data = pd.read_csv("Online_Retail.csv", encoding='latin1')
data.dropna(inplace=True)
data = data[~data['InvoiceNo'].astype(str).str.startswith('C')]
data = data[data['Quantity'] > 0]
basket = (data[data['Country'] == "United Kingdom"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
def encode_units(x):
 return 1 if x >= 1 else 0
basket_sets = basket.apply(lambda x: x.map(encode_units))
frequent_itemsets = apriori(basket_sets, min_support=0.02, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print("Frequent Itemsets:\n", frequent_itemsets.head())
print("\nAssociation Rules:\n",
rules[['antecedents','consequents','support','confidence','lift']].head())
```

Output:

```
Frequent Itemsets:
     support                          itemsets
0  0.022404         (3 STRIPEY MICE FELTCRAFT)
1  0.037720           (6 RIBBONS RUSTIC CHARM)
2  0.025767  (60 CAKE CASES VINTAGE CHRISTMAS)
3  0.035257      (60 TEATIME FAIRY CAKE CASES)
4  0.026668   (72 SWEETHEART FAIRY CAKE CASES)
```

Association Rules:

	antecedents	consequents	support	confidence	lift
0	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.027269	0.657971	14.451925
1	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.027269	0.598945	14.451925
2	(GARDENERS KNEELING PAD KEEP CALM)	(GARDENERS KNEELING PAD CUP OF TEA)	0.027509	0.617251	16.390122
3	(GARDENERS KNEELING PAD CUP OF TEA)	(GARDENERS KNEELING PAD KEEP CALM)	0.027509	0.730463	16.390122
4	(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER)	0.024266	0.819473	22.293137

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 04

Aim: Implement Naive Bayes Classifier and K-Nearest Neighbor Classifier on Data set of your choice. Test and Compare for Accuracy and Precision.

Source Code:

```
# Import libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, classification_report

# Load dataset (Iris for demo)
iris = load_iris()
X = iris.data
y = iris.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# ---------- Naive Bayes ----------
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
```

```
nb_accuracy = accuracy_score(y_test, y_pred_nb)
nb_precision = precision_score(y_test, y_pred_nb, average="weighted")

# ---------- K-Nearest Neighbor ----------
knn = KNeighborsClassifier(n_neighbors=5)  # k=5
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

knn_accuracy = accuracy_score(y_test, y_pred_knn)
knn_precision = precision_score(y_test, y_pred_knn, average="weighted")

# ---------- Results ----------
print("Naive Bayes Classifier Results")
print("Accuracy :", nb_accuracy)
print("Precision:", nb_precision)
print("\nClassification Report:\n", classification_report(y_test, y_pred_nb))

print("\nK-Nearest Neighbor Classifier Results")
print("Accuracy :", knn_accuracy)
print("Precision:", knn_precision)
print("\nClassification Report:\n", classification_report(y_test, y_pred_knn))

# Final Comparison
print("\n---- Model Comparison ----")
print(f"Naive Bayes -> Accuracy: {nb_accuracy:.3f}, Precision: {nb_precision:.3f}")
print(f"KNN         -> Accuracy: {knn_accuracy:.3f}, Precision: {knn_precision:.3f}")
```

Output:

```
Naive Bayes Classifier Results
Accuracy : 0.9777777777777777
Precision: 0.9793650793650793

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45


K-Nearest Neighbor Classifier Results
Accuracy : 1.0
Precision: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


---- Model Comparison ----
Naive Bayes -> Accuracy: 0.978, Precision: 0.979
KNN         -> Accuracy: 1.000, Precision: 1.000
```

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 05

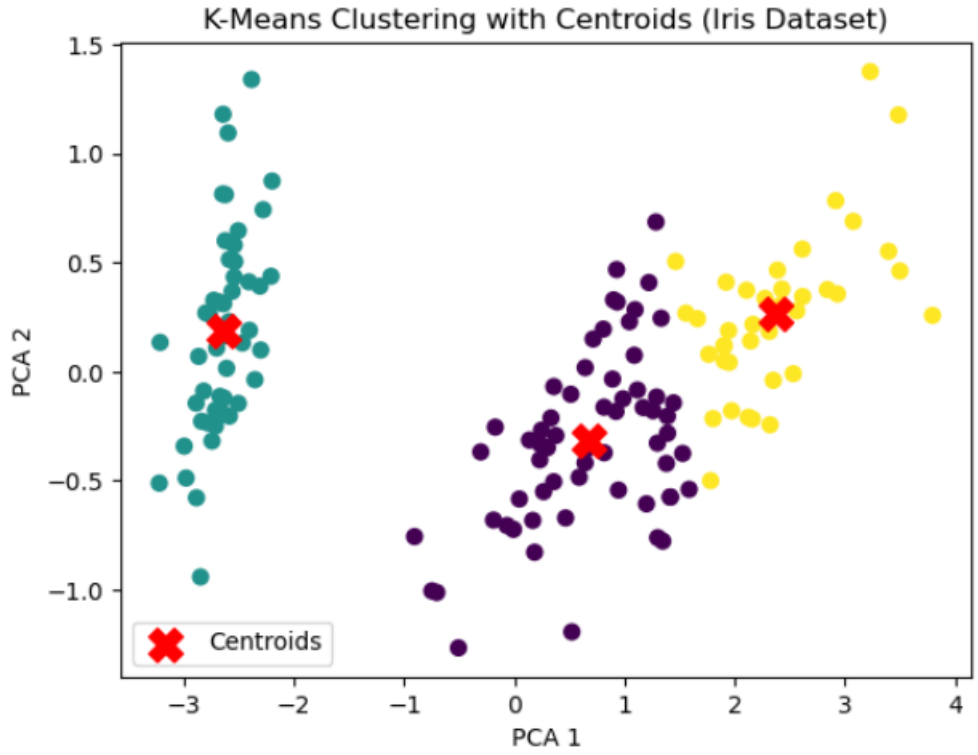
Aim: Implement K-Means Clustering on the proper data set of your choice.

Source Code:

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load data
iris = load_iris()
X = iris.data

# Apply K-Means with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
y_pred = kmeans.fit_predict(X)

# Reduce to 2D for plotting
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
centroids_pca = pca.transform(kmeans.cluster_centers_)  # project centroids

# Plot clusters
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_pred, cmap="viridis", s=40)
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1],
            c="red", marker="X", s=200, label="Centroids")
plt.title("K-Means Clustering with Centroids (Iris Dataset)")
plt.xlabel("PCA 1")
```

```
plt.ylabel("PCA 2")
plt.legend()
plt.show()

# Print centroid coordinates
print("Cluster Centers (original feature space):\n", kmeans.cluster_centers_)
```

Output:




```
Cluster Centers (original feature space):
 [[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.428      1.462      0.246     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 06

Aim: Design and implement SVM for classification with the proper dataset of your choice commenton design and implementation for linearly non sepearble datsset

Source Code:

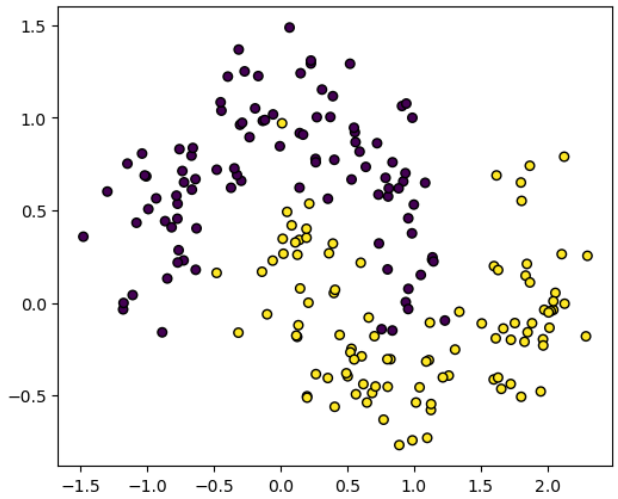
```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris, make_moons
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# ---------- LINEARLY SEPARABLE DATA (Iris) ----------
print("===== SVM on Iris Dataset (Linearly Separable) =====")
iris = load_iris()
X, y = iris.data, iris.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Linear Kernel SVM
svm_linear = SVC(kernel="linear", C=1.0)
svm_linear.fit(X_train, y_train)
y_pred = svm_linear.predict(X_test)

print("Accuracy (Linear Kernel):", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# ---------- LINEARLY NON-SEPARABLE DATA (Moons) ----------
print("\n===== SVM on Moons Dataset (Linearly Non-Separable) =====")
```

```
X, y = make_moons(n_samples=200, noise=0.2, random_state=42)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# RBF Kernel SVM (handles non-linear data)
svm_rbf = SVC(kernel="rbf", C=1.0, gamma="scale")
svm_rbf.fit(X_train, y_train)
y_pred = svm_rbf.predict(X_test)

print("Accuracy (RBF Kernel):", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# ---------- VISUALIZATION ----------
def plot_decision_boundary(model, X, y, title):
    plt.figure(figsize=(6, 5))
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap="viridis", s=30, edgecolors="k")

    # Create grid
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    xx, yy = np.meshgrid(
        np.linspace(xlim[0], xlim[1], 200),
        np.linspace(ylim[0], ylim[1], 200)
    )

    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
```

```
    plt.contour(xx, yy, Z, colors="k", levels=[-1, 0, 1], alpha=0.7,
                linestyles=["--", "-", "--"])
    plt.title(title)
    plt.show()

# Visualize only Moons dataset (2D)
plot_decision_boundary(svm_rbf, X, y, "SVM with RBF Kernel (Moons Dataset)")
```

Output:

```
===== SVM on Iris Dataset (Linearly Separable) =====
Accuracy (Linear Kernel): 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


===== SVM on Moons Dataset (Linearly Non-Separable) =====
Accuracy (RBF Kernel): 0.9666666666666667
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        32
           1       1.00      0.93      0.96        28

    accuracy                           0.97        60
   macro avg       0.97      0.96      0.97        60
weighted avg       0.97      0.97      0.97        60
```



Name: Sayali Salunke **Subject:** Machine Learning Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Dr. Kapil K Misal

---

Practical No: 07

Aim: Implement a basic not gate using perceptron.

Source Code:

```
# importing Python library
import numpy as np
# define Unit Step Function
def unitStep(v):
 if v >= 0:
  return 1
 else:
  return 0
# design Perceptron Model
def perceptronModel(x, w, b):
 v = np.dot(w, x) + b
 y = unitStep(v)
 return y
# NOT Logic Function
# w = -1, b = 0.5
def NOT_logicFunction(x):
 w = -1
 b = 0.5
 return perceptronModel(x, w, b)
# testing the Perceptron Model
test1 = np.array(1)
test2 = np.array(0)
print("NOT({}) = {}".format(1, NOT_logicFunction(test1)))
print("NOT({}) = {}".format(0, NOT_logicFunction(test2)))
```

Output:

```
NOT(1) = 0
NOT(0) = 1
```