Practical No: 1
Name: Thorave Avishkar Shrikrushna
Roll No: 65

Class: BE AI&DS
Title : To use PCA Algorithm for dimensionality reduction. You have a dataset that includes measurements for different variables on wine (alcohol, ash, magnesium, and so on). Apply PCA algorithm & transform this data so that most variations in the measurements of the variables are captured by a small number of principal components so that it is easier to distinguish between red and white wine by inspecting these principal components.
Subject : Computer Laboratory 1 (Machine Learning) 417525

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

In [3]:
```
data=pd.read_csv('Wine.csv')
```

In [5]:
```
 data.head()
```

Out[5]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Non
0	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	

In [7]:
```
data.tail()
```

Out[7]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	N
173	13.71	5.65	2.45	20.5	95	1.68	0.61	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	

In [9]:
```
data.shape
```

Out[9]: (178, 14)

In [11]:
```
data.describe()
```

Out[11]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Fl
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	17
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	

In [13]:
```
data['Customer_Segment'].unique()
```

Out[13]:
```
array([1, 2, 3], dtype=int64)
```

In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Alcohol               178 non-null    float64
 1   Malic_Acid            178 non-null    float64
 2   Ash                   178 non-null    float64
 3   Ash_Alcanity          178 non-null    float64
 4   Magnesium             178 non-null    int64  
 5   Total_Phenols         178 non-null    float64
 6   Flavanoids            178 non-null    float64
 7   Nonflavanoid_Phenols  178 non-null    float64
 8   Proanthocyanins       178 non-null    float64
 9   Color_Intensity       178 non-null    float64
 10  Hue                   178 non-null    float64
 11  OD280                 178 non-null    float64
 12  Proline               178 non-null    int64  
 13  Customer_Segment      178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
Alcohol                 0
Malic_Acid              0
Ash                     0
Ash_Alcanity            0
Magnesium               0
Total_Phenols           0
Flavanoids              0
Nonflavanoid_Phenols    0
Proanthocyanins         0
Color_Intensity         0
Hue                     0
OD280                   0
Proline                 0
Customer_Segment        0
dtype: int64
```

In [19]:
```
x=data.drop('Customer_Segment',axis=1)
y=data['Customer_Segment']
```

In [21]:
```
x
```

Out[21]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	N
0	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	

178 rows × 13 columns

In [23]:
```
x.shape
```

Out[23]:
```
(178, 13)
```

In [25]:
```
 x_standardized = (x - x.mean()) / x.std()
```

In [27]:
```
y
```

Out[27]:
```
0      1
1      1
2      1
3      1
4      1
      ..
173    3
174    3
175    3
176    3
177    3
Name: Customer_Segment, Length: 178, dtype: int64
```

In [37]:
```
pca=PCA(n_components=3)
```

In [39]:
```
x_pca=pca.fit_transform(x)
```

In [41]:
```
x_pca.shape
```

Out[41]:
```
(178, 3)
```

In [43]:
```
pca_df = pd.DataFrame(x_pca, columns = ['pca_col1','pca_col2','pca_col3'])
```

In [45]:
```
pca_df
```

Out[45]:

	pca_col1	pca_col2	pca_col3
0	318.562979	21.492131	-3.130735
1	303.097420	-5.364718	-6.822835
2	438.061133	-6.537309	1.113223
3	733.240139	0.192729	0.917257
4	-11.571428	18.489995	0.554422
...	...	...	...
173	-6.980211	-4.541137	2.474707
174	3.131605	2.335191	4.309931
175	88.458074	18.776285	2.237577
176	93.456242	18.670819	1.788392
177	-186.943190	-0.213331	5.630510

178 rows × 3 columns

In [47]:
```
pca.explained_variance_ratio_
```

Out[47]:
```
array([9.98091230e-01, 1.73591562e-03, 9.49589576e-05])
```

Practical No: 2
Name : Thorave Avishkar Shrikrushna
Roll No : 65
Class: BE AI&DS
Title : Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks: 1. Pre-process the dataset. 2. Identify outliers. 3. Check the correlation. 4.Implement linear regression and ridge, Lasso regression models. 5.
Evaluate the models and compare their respective scores like R2, RMSE, etc. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset
Subject : Computer Laboratory 1 (Machine Learning) 417525

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
```

In [3]:
```
# 1. Load and Pre-process the Dataset
df = pd.read_csv('uber.csv')  # Change the name to the correct csv file
```

In [4]:
```
print("Initial Data Info:")
print(df.info())
print(df.head())
```

```
Initial Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
None
   Unnamed: 0                            key  fare_amount  \
0    24238194    2015-05-07 19:52:06.0000003          7.5   
1    27835199    2009-07-17 20:04:56.0000002          7.7   
2    44984355   2009-08-24 21:45:00.00000061         12.9   
3    25894730    2009-06-26 08:22:21.0000001          5.3   
4    17610152  2014-08-28 17:47:00.000000188         16.0   

           pickup_datetime  pickup_longitude  pickup_latitude  \
0  2015-05-07 19:52:06 UTC        -73.999817        40.738354   
1  2009-07-17 20:04:56 UTC        -73.994355        40.728225   
2  2009-08-24 21:45:00 UTC        -74.005043        40.740770   
3  2009-06-26 08:22:21 UTC        -73.976124        40.790844   
4  2014-08-28 17:47:00 UTC        -73.925023        40.744085   

   dropoff_longitude  dropoff_latitude  passenger_count  
0         -73.999512         40.723217                1  
1         -73.994710         40.750325                1  
2         -73.962565         40.772647                1  
3         -73.965316         40.803349                3  
4         -73.973082         40.761247                5  
```

In [5]:
```
# Drop rows with missing values
df = df.dropna()
```

In [6]:
```
# Convert datetime columns
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
```

In [7]:
```
# Feature Engineering: Extract hour, day, month
df['hour'] = df['pickup_datetime'].dt.hour
df['day'] = df['pickup_datetime'].dt.day
df['month'] = df['pickup_datetime'].dt.month
```

In [8]:
```
# Remove unneeded columns
df = df.drop(columns=['key', 'pickup_datetime'])
```

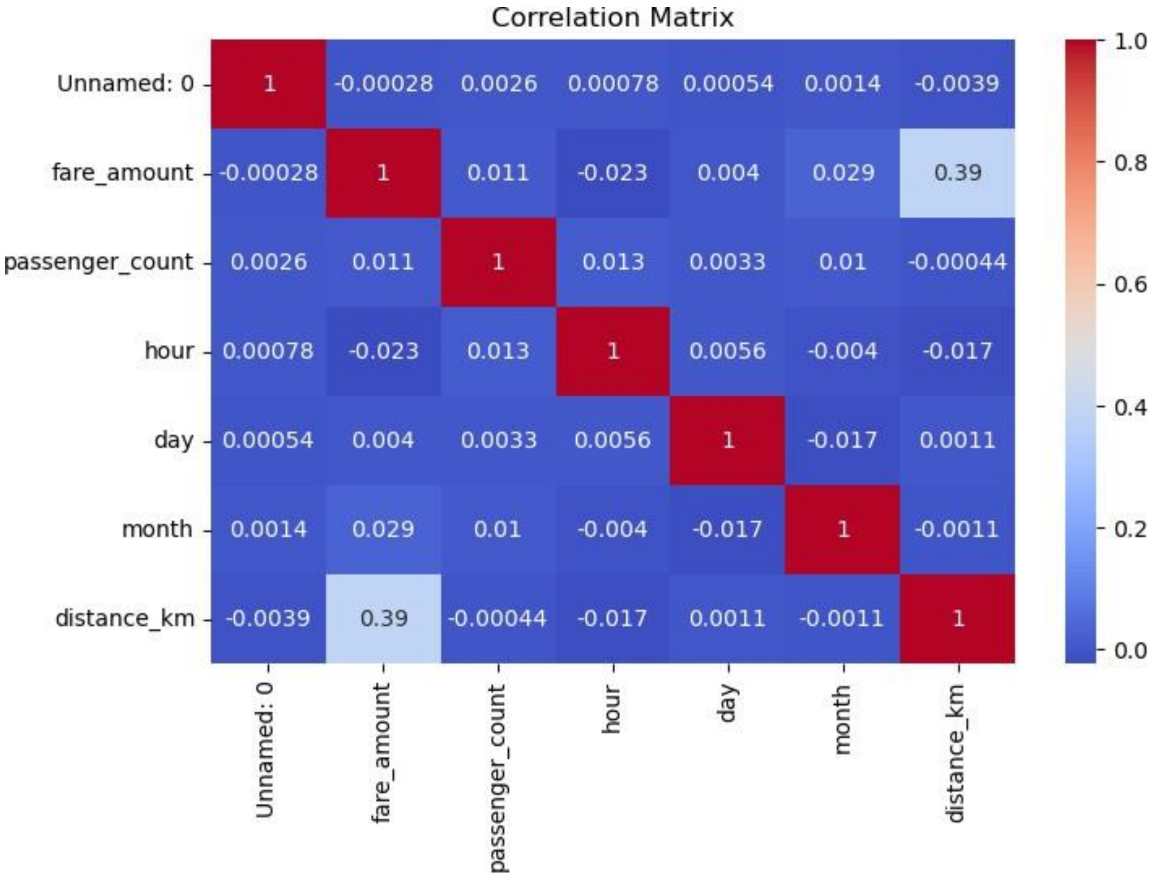
In [9]:
```
# Calculate distance (Haversine formula)
def haversine(lat1, lon1, lat2, lon2):
    R = 6371  # Earth radius in km
    phi1, phi2 = np.radians(lat1), np.radians(lat2)
    dphi = np.radians(lat2 - lat1)
    dlambda = np.radians(lon2 - lon1)
```

```
    a = np.sin(dphi/2)**2 + np.cos(phi1)*np.cos(phi2)*np.sin(dlambda/2)**2
    return 2 * R * np.arcsin(np.sqrt(a))

df['distance_km'] = haversine(df['pickup_latitude'], df['pickup_longitude'],
                              df['dropoff_latitude'], df['dropoff_longitude'])
```

In [10]:
```
# Drop original lat/lon columns
df = df.drop(columns=['pickup_latitude', 'pickup_longitude', 'dropoff_latitude',
```

In [11]:
```
# 2. Identify Outliers (using z-score on fare_amount and distance_km)
from scipy.stats import zscore
df = df[(np.abs(zscore(df[['fare_amount', 'distance_km']])) < 3).all(axis=1)]
```

In [12]:
```
# 3. Check Correlation
plt.figure(figsize=(8, 5))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



In [13]:
```
# 4. Regression Models
X = df.drop(columns=['fare_amount'])
y = df['fare_amount']
```

In [14]:
```
# One-hot encode categorical columns (if any)
X = pd.get_dummies(X, drop_first=True)
```

In [15]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [16]:
```
# Linear Regression
lr = LinearRegression()
```

```
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

In [18]:
```
# Ridge Regression
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred_ridge = ridge.predict(X_test)
```

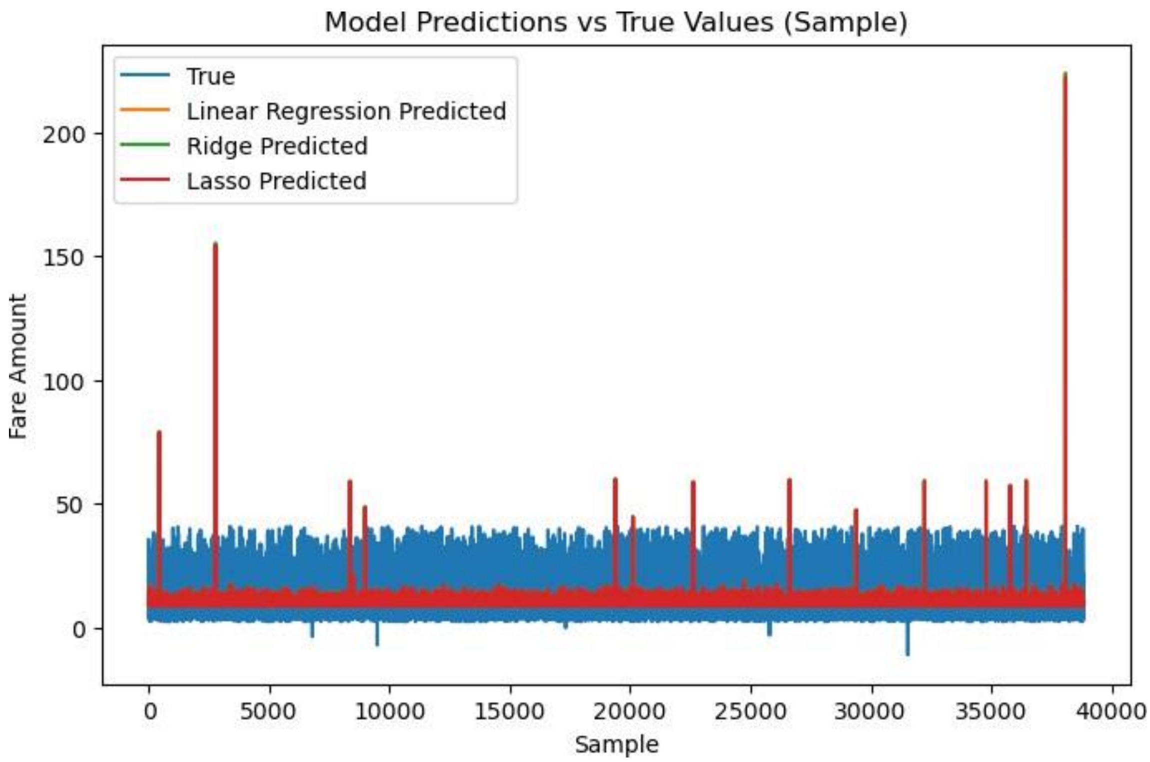
In [19]:
```
# Lasso Regression
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)
```

In [21]:
```
# 5. Model Evaluation
def print_scores(model_name, y_true, y_pred):
    print(f"\n{model_name}")
    print(f"R2 Score: {r2_score(y_true, y_pred):.4f}")
    print(f"RMSE: {np.sqrt(mean_squared_error(y_true, y_pred)):.4f}")
```

In [22]:
```
print_scores("Linear Regression", y_test, y_pred_lr)
print_scores("Ridge Regression", y_test, y_pred_ridge)
print_scores("Lasso Regression", y_test, y_pred_lasso)
```

```
Linear Regression
R2 Score: 0.1917
RMSE: 5.8158

Ridge Regression
R2 Score: 0.1917
RMSE: 5.8158

Lasso Regression
R2 Score: 0.1909
RMSE: 5.8187
```

In [25]:
```
# Optional: Compare visually
plt.figure(figsize=(8,5))
plt.plot(y_test.values, label='True')
plt.plot(y_pred_lr, label='Linear Regression Predicted')
plt.plot(y_pred_ridge, label='Ridge Predicted')
plt.plot(y_pred_lasso, label='Lasso Predicted')
plt.title('Model Predictions vs True Values (Sample)')
plt.xlabel('Sample')
plt.ylabel('Fare Amount')
plt.legend()
plt.show()
```



Model Predictions vs True Values (Sample)
True
Linear Regression Predicted
Ridge Predicted
Lasso Predicted
Fare Amount
Sample
0
50
100
150
200
0
5000
10000
15000
20000
25000
30000
35000
40000

Practical No: 4
Name: Thorave Avishkar Shrikrushna
Roll No: 65

Class: BE AI&DS

Title : Implement K-Means clustering on Iris.csv dataset. Determine the number of clustersusing the elbow method.

Subject : Computer Laboratory 1 (Machine Learning) 417525

In [27]:
```
import pandas as pd # Pandas (version : 1.1.5)
import numpy as np # Numpy (version : 1.19.2)
import matplotlib.pyplot as plt # Matplotlib (version :  3.3.2)
from sklearn.cluster import KMeans # Scikit Learn (version : 0.23.2)
import seaborn as sns # Seaborn (version : 0.11.1)
sns.set()
```

In [35]:
```
data = pd.read_csv('iris.csv')
print(data)
```

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

In [31]:
```
data.head()
```

Out[31]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [37]:
```
data.tail()
```

Out[37]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [39]:
```
len(data)
```

Out[39]:
```
150
```

In [41]:
```
data.shape
```

Out[41]:
```
(150, 5)
```

In [43]:
```
data.columns
```

Out[43]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [47]:
```
for i,col in enumerate(data.columns):
    print(f'Column number {1+i} is {col}')
```

```
Column number 1 is sepal_length
Column number 2 is sepal_width
Column number 3 is petal_length
Column number 4 is petal_width
Column number 5 is species
```

In [49]:
```
data.dtypes
```

Out[49]:
```
sepal_length     float64
sepal_width      float64
petal_length     float64
petal_width      float64
species           object
dtype: object
```

In [51]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [53]:
```
data.describe()
```

Out[53]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [55]:

```
#Checking data for missing values using isnull()
data.isnull()
```

Out[55]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [69]:

```
print(data.columns.tolist())
```

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

In [71]:

```
data = pd.read_csv('iris.csv',header=0)
```

In [77]:

```
data = pd.read_csv('iris.csv',header=None)
data.columns=["Id","sepal_length","sepal_width","petal_length","petal_width"]
```

In [79]:

```
print(data.columns.tolist())
```

```
['Id', 'sepal_length', 'sepal_width', 'petal_length', 'petal_width']
```

In [83]:

```
#Checking summary of missing values
```

```
data.isnull().sum()
```

Out[83]:

```
Id              0
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
dtype: int64
```

In [87]:

```
#Deleting 'customer_id' colummn using drop().
data.drop('Id', axis=1, inplace=True)
data.head()
```

Out[87]:

	sepal_length	sepal_width	petal_length	petal_width
0	sepal_width	petal_length	petal_width	species
1	3.5	1.4	0.2	setosa
2	3.0	1.4	0.2	setosa
3	3.2	1.3	0.2	setosa
4	3.1	1.5	0.2	setosa

In [89]:

```
data.isna().sum()
```

Out[89]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
dtype: int64
```

In [91]:

```
data.head()
```

Out[91]:

	sepal_length	sepal_width	petal_length	petal_width
0	sepal_width	petal_length	petal_width	species
1	3.5	1.4	0.2	setosa
2	3.0	1.4	0.2	setosa
3	3.2	1.3	0.2	setosa
4	3.1	1.5	0.2	setosa

In [111...

```
data = pd.read_csv('iris.csv',header=None)
data.columns=["sepal_length","sepal_width","petal_length","petal_width","Species
```

In [116...

```
data.head()
print(data.columns)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'Species'],
      dtype='object')
```

In [118...

```
data['Species'].value_counts()
```

Out[118…

```
Species
setosa        50
versicolor    50
virginica     50
species        1
Name: count, dtype: int64
```

In [120…

```
#Target Data
target_data = data.iloc[:,4]
target_data.head()
```

Out[120…

```
0     species
1      setosa
2      setosa
3      setosa
4      setosa
Name: Species, dtype: object
```

In [122…

```
#Training data
clustering_data = data.iloc[:,[0,1,2,3]]
clustering_data.head()
```

Out[122…

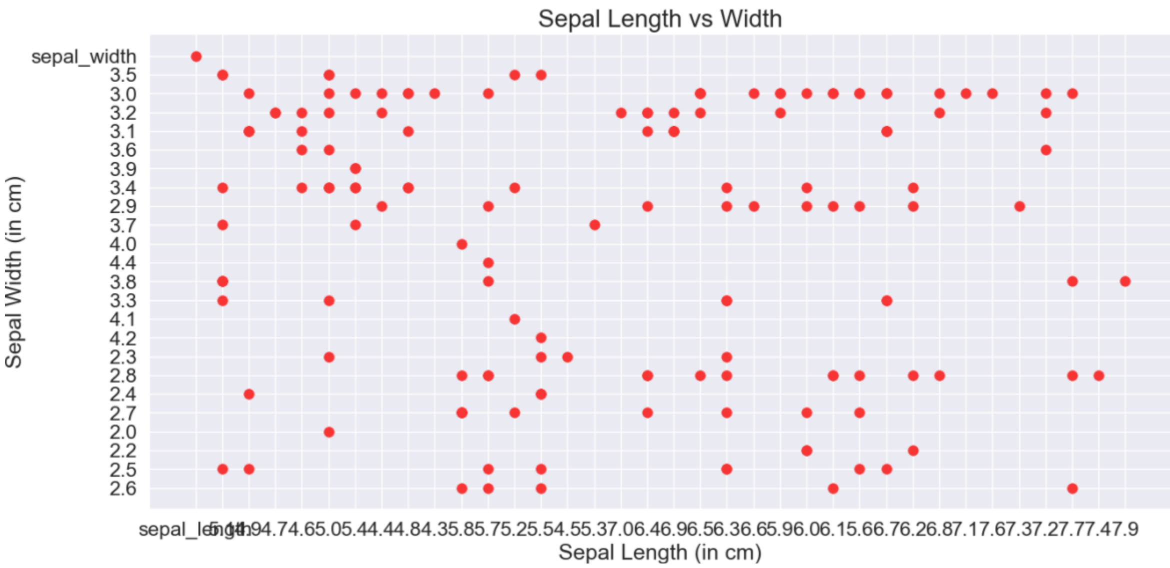
	sepal_length	sepal_width	petal_length	petal_width
0	sepal_length	sepal_width	petal_length	petal_width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2

In [154…

```
data.columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm","Spe
```

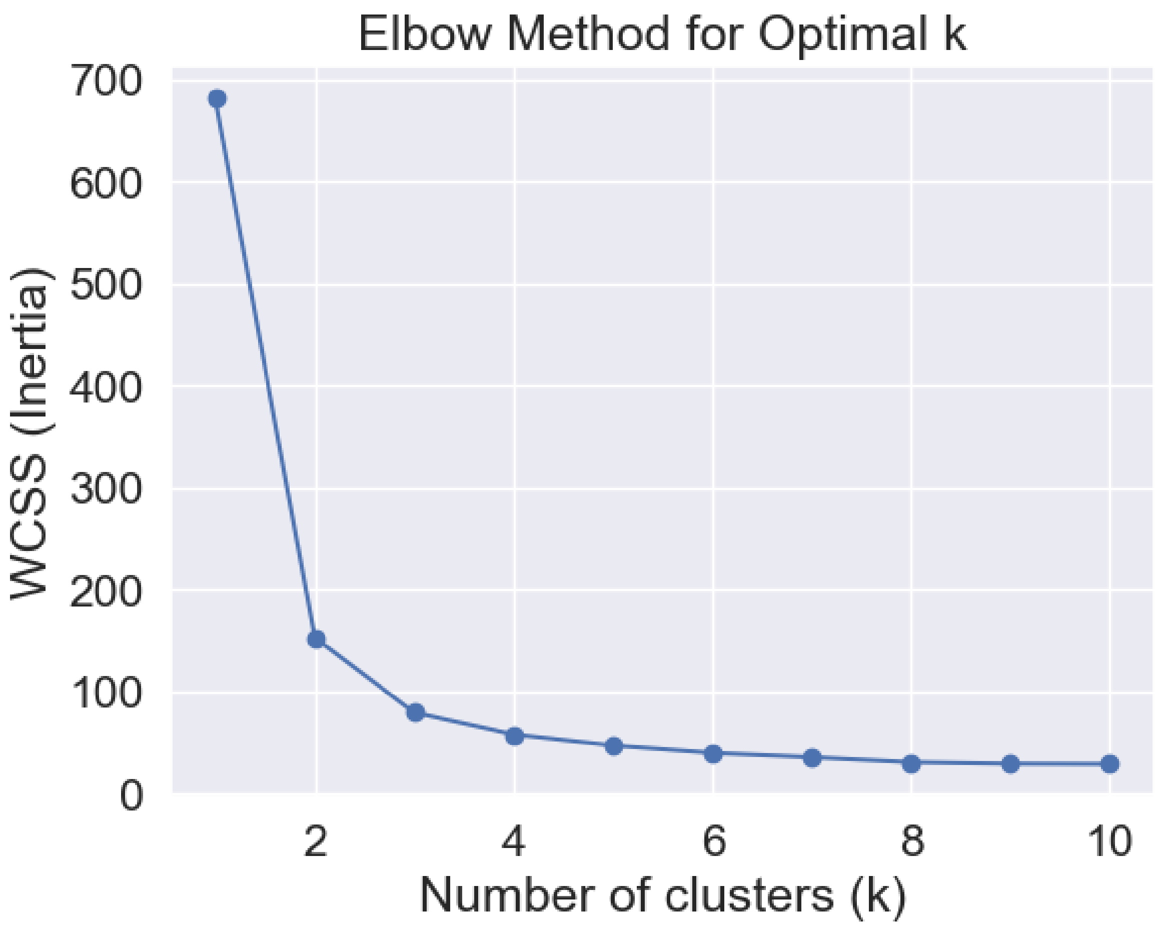
In [156…

```
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.scatterplot(x=data['SepalLengthCm'],y=data['SepalWidthCm'], s=70, color
ax.set_ylabel('Sepal Width (in cm)')
ax.set_xlabel('Sepal Length (in cm)')
plt.title('Sepal Length vs Width', fontsize = 20)
plt.show()
```



In [188…

```
#The Elbow Method
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Step 1: Load the dataset
clustering_data = pd.read_csv('iris.csv')

# Step 2: Select only numeric columns (exclude non-numeric like 'species' if pre
# If the dataset has columns: sepal_length, sepal_width, petal_length, petal_wid
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
X = clustering_data[features]

# Step 3: Elbow method to find optimal number of clusters
wcss = []
for i in range(1, 11):
    km = KMeans(n_clusters=i, random_state=42)
    km.fit(X)
    wcss.append(km.inertia_)

# Step 4: Plot the elbow curve
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('WCSS (Inertia)')
plt.grid(True)
plt.show()
```



In [194…

```
# clustering
from sklearn.cluster import KMeans

# Select only the numeric columns (exclude 'species')
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
X = clustering_data[features]

# Create and fit the model
kms = KMeans(n_clusters=3, init='k-means++', n_init='auto', random_state=42)
kms.fit(X)

# Create a copy and add cluster predictions
clusters = clustering_data.copy()
clusters['Cluster_Prediction'] = kms.predict(X)

# Show the result
clusters.head()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[194…

	sep al_length	sepal_width	petal_length	petal_width	species	Cluster_Prediction
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1

In [196…

```
kms.cluster_centers_
```

Out[196…

```
array([[6.85384615, 3.07692308, 5.71538462, 2.05384615],
       [5.006     , 3.418     , 1.464     , 0.244     ],
       [5.88360656, 2.74098361, 4.38852459, 1.43442623]])
```

In [214…

```
print(clusters.columns.tolist())
```

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species', 'cluste
r_prediction']
```

In [238…

```
from sklearn.cluster import KMeans

# Assume X is your input features, e.g., SepalLengthCm and SepalWidthCm
kms = KMeans(n_clusters=3, random_state=0)
clusters = X.copy()  # Copy your input DataFrame

# Add predicted cluster labels
clusters['Cluster_Prediction'] = kms.fit_predict(X)
```

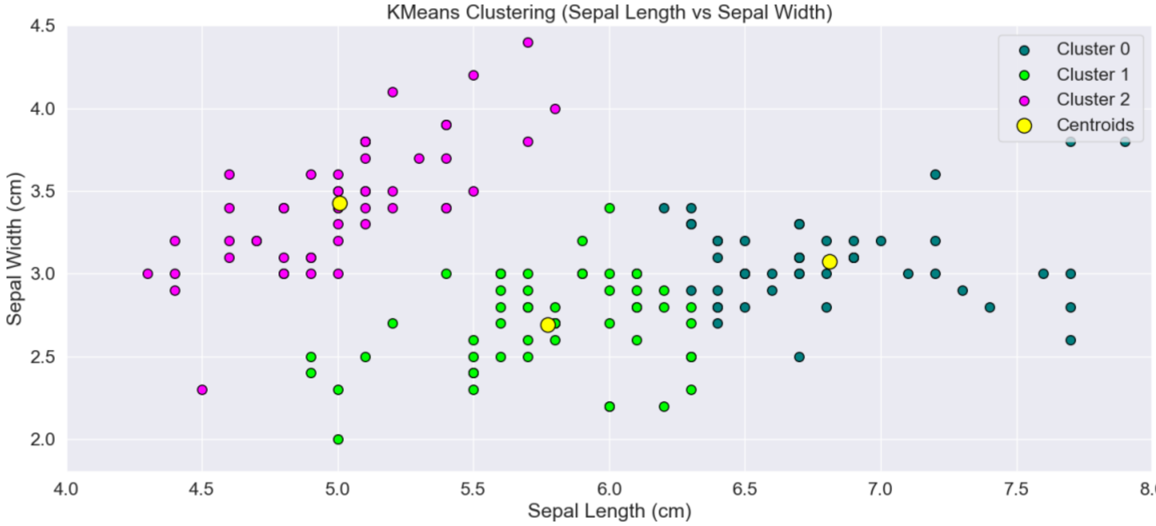
```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429:   UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [240…

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Load dataset (example: Iris dataset)
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df.columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']

# Select features for clustering
X = df[['SepalLengthCm', 'SepalWidthCm']]

# Fit KMeans
kms = KMeans(n_clusters=3, random_state=42)
df['Cluster_Prediction'] = kms.fit_predict(X)

# Plotting
fig, ax = plt.subplots(figsize=(15, 7))

# Cluster 0
plt.scatter(
```

```
    df[df['Cluster_Prediction'] == 0]['SepalLengthCm'],
    df[df['Cluster_Prediction'] == 0]['SepalWidthCm'],
    s=70, c='teal', edgecolor='black', label='Cluster 0'
)

# Cluster 1
plt.scatter(
    df[df['Cluster_Prediction'] == 1]['SepalLengthCm'],
    df[df['Cluster_Prediction'] == 1]['SepalWidthCm'],
    s=70, c='lime', edgecolor='black', label='Cluster 1'
)

# Cluster 2
plt.scatter(
    df[df['Cluster_Prediction'] == 2]['SepalLengthCm'],
    df[df['Cluster_Prediction'] == 2]['SepalWidthCm'],
    s=70, c='magenta', edgecolor='black', label='Cluster 2'
)

# Plot centroids
plt.scatter(
    kms.cluster_centers_[:, 0], kms.cluster_centers_[:, 1],
    s=170, c='yellow', edgecolor='black', label='Centroids'
)

# Labels, limits, etc.
plt.title('KMeans Clustering (Sepal Length vs Sepal Width)', fontsize=18)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.xlim(4, 8)
plt.ylim(1.8, 4.5)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [ ]:

Practical No: 5
Name : Thorave Avishkar Shrikrushna
Roll No: 65

Class : BE AI&DS

Title : Use different voting mechanism and Apply AdaBoost (Adaptive Boosting), GradientTree Boosting (GBM), XGBoost classification on Iris dataset and compare the performance of three models using different evaluation measures.

Subject : Computer Laboratory 1 (Machine Learning) 417525

In [1]:
```
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()
```

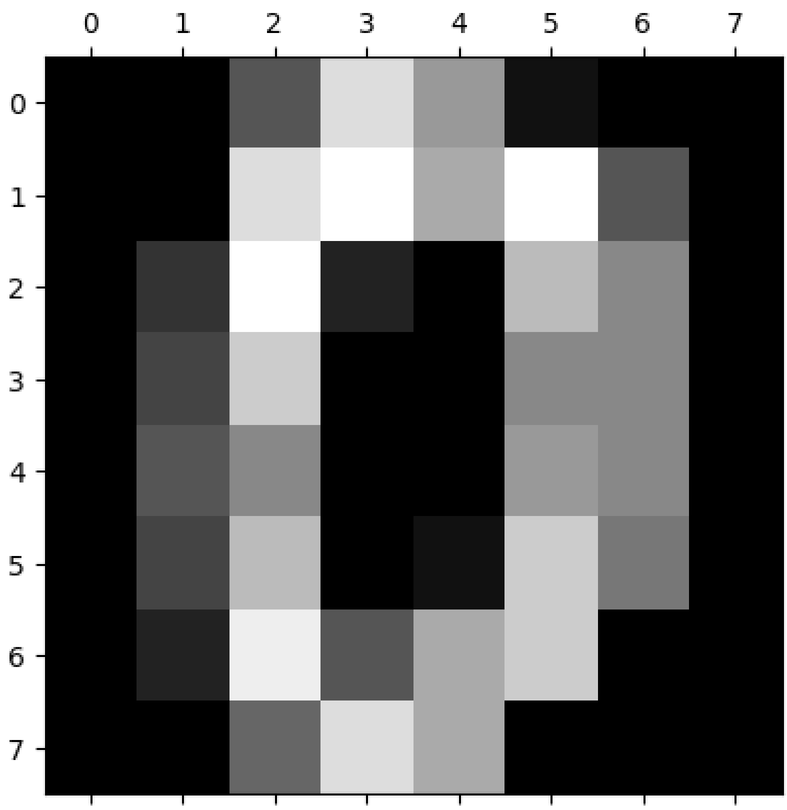
In [3]:
```
dir(digits)
```

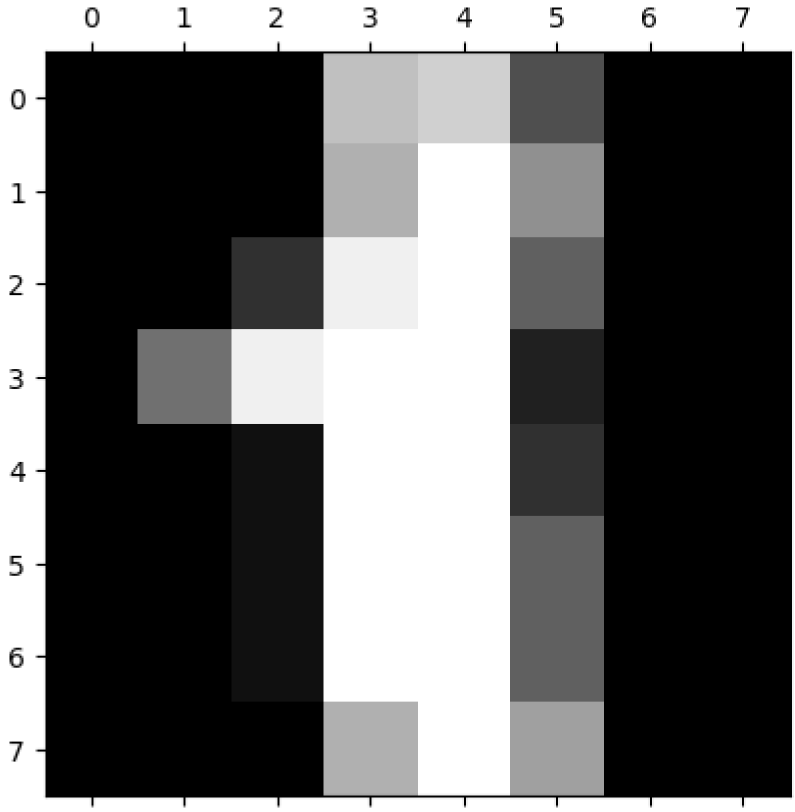
Out[3]:
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

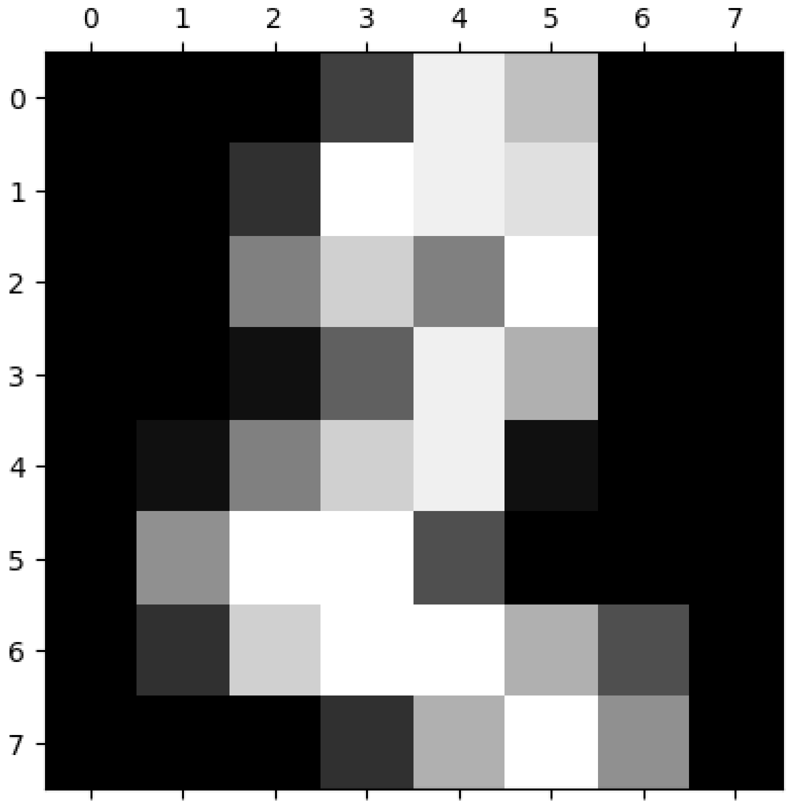
In [5]:
```
#%matplotlib inline
import matplotlib.pyplot as plt
```

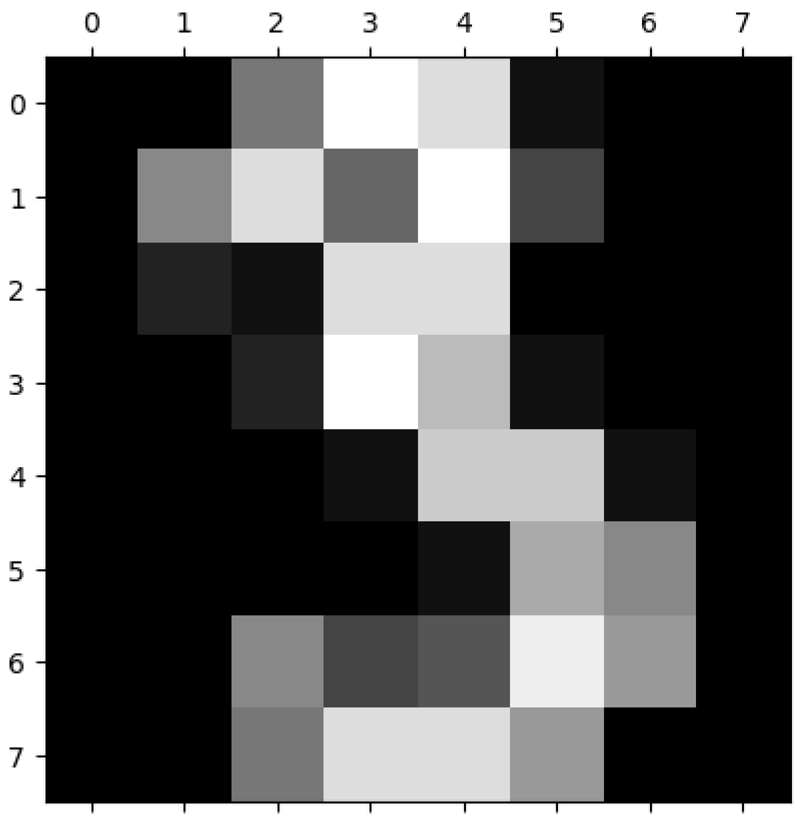
In [9]:
```
plt.gray()
for i in range(4):
    plt.matshow(digits.images[i])
```

```
<Figure size 640x480 with 0 Axes>
```




0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7



In [13]:
```
df = pd.DataFrame(digits.data)
df.head()
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	6
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0	10.
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0	16.
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0	11.
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	7.0	13.0	13.
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2.0	16.

5 rows × 64 columns

In [15]:
```
df['target'] = digits.target
df[0:12]
```

Out[15]:

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	13.0	10.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	16.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	11.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	13.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	16.0
5	0.0	0.0	12.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	9.0	16.0	16.0
6	0.0	0.0	0.0	12.0	13.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	9.0	15.0
7	0.0	0.0	7.0	8.0	13.0	16.0	15.0	1.0	0.0	0.0	...	0.0	0.0	0.0	13.0	5.0	0.0
8	0.0	0.0	9.0	14.0	8.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	11.0	16.0	15.0
9	0.0	0.0	11.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	9.0	12.0	13.0
10	0.0	0.0	1.0	9.0	15.0	11.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	10.0	13.0
11	0.0	0.0	0.0	0.0	14.0	13.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	13.0

12 rows × 65 columns

In [21]:
```
#Train and the model and prediction
X = df.drop('target',axis='columns')
y = df.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
RandomForestClassifier(n_estimators=20)
model.score(X_test, y_test)
```

Out[21]: 0.9722222222222222

In [23]:
```
y_predicted = model.predict(X_test)
```

In [25]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

Out[25]:
```
array([[34,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 40,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 34,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  0,  1, 32,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 33,  0,  0,  1,  0,  1],
       [ 0,  0,  0,  0,  0, 27,  0,  0,  0,  1],
       [ 1,  1,  0,  0,  0,  0, 38,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 39,  0,  0],
       [ 0,  1,  0,  0,  0,  0,  0,  0, 29,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 44]], dtype=int64)
```

In [27]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

```python
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm,  annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[27]:

```
Text(95.72222222222221, 0.5, 'Truth')
```