VELAMMAL ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

21CS404L _ DATA SCIENCE USING PYTHON LABORATORY

LAB MANUAL

Experiment 1
Reading and writing different types of datasets

AIM:
To install pandas in Jupyter notebook and to read, write data in both csv
CODE:

```
#Creating dataset
pip install pandas
data = {
'CHN': {'COUNTRY': 'China', 'POP': 1_398.72, 'AREA': 9_596.96,
'GDP': 12_234.78, 'CONT': 'Asia'},
'IND': {'COUNTRY': 'India', 'POP': 1_351.16, 'AREA': 3_287.26,
'GDP': 2_575.67, 'CONT': 'Asia', 'IND_DAY': '1947-08-15'},
'USA': {'COUNTRY': 'US', 'POP': 329.74, 'AREA': 9_833.52,
'GDP': 19_485.39, 'CONT': 'N.America',
'IND_DAY': '1776-07-04'},
'IDN': {'COUNTRY': 'Indonesia', 'POP': 268.07, 'AREA': 1_910.93,
'GDP': 1_015.54, 'CONT': 'Asia', 'IND_DAY': '1945-08-17'},
'BRA': {'COUNTRY': 'Brazil', 'POP': 210.32, 'AREA': 8_515.77,
'GDP': 2_055.51, 'CONT': 'S.America', 'IND_DAY': '1822-09-07'},
'PAK': {'COUNTRY': 'Pakistan', 'POP': 205.71, 'AREA': 881.91,
'GDP': 302.14, 'CONT': 'Asia', 'IND_DAY': '1947-08-14'},
'NGA': {'COUNTRY': 'Nigeria', 'POP': 200.96, 'AREA': 923.77,
'GDP': 375.77, 'CONT': 'Africa', 'IND_DAY': '1960-10-01'},
'BGD': {'COUNTRY': 'Bangladesh', 'POP': 167.09, 'AREA': 147.57,
'GDP': 245.63, 'CONT': 'Asia', 'IND_DAY': '1971-03-26'},
'RUS': {'COUNTRY': 'Russia', 'POP': 146.79, 'AREA': 17_098.25,
'GDP': 1_530.75, 'IND_DAY': '1992-06-12'},
'MEX': {'COUNTRY': 'Mexico', 'POP': 126.58, 'AREA': 1_964.38,
'GDP': 1_158.23, 'CONT': 'N.America', 'IND_DAY': '1810-09-16'},
'JPN': {'COUNTRY': 'Japan', 'POP': 126.22, 'AREA': 377.97,
'GDP': 4_872.42, 'CONT': 'Asia'}
}
```

```
columns = ('COUNTRY', 'POP', 'AREA', 'GDP', 'CONT', 'IND_DAY')


print(data)
print(columns)

import pandas as pd
df = pd.DataFrame(data=data).T
df
df = pd.DataFrame(data=data, index=columns).T
df
#Write a CSV File
df.to_csv('data.csv')
#Read a CSV file
df = pd.read_csv('data.csv', index_col=0)
df
#Write an Excel File
df.to_excel('data.xlsx')
#Read an Excel File
df = pd.read_excel('data.xlsx', index_col=0)
df
```

RESULT:
Thus the above python code was executed and verified successfully.

Experiment 2 **Python Program to implement sorting and ranking**

AIM:

To sort and rank the data in a list in Python

CODE:

Sorting

```
import pandas as pd
import numpy as np
s = pd.Series(range(5),index = ['e', 'd', 'a', 'b', 'c'])
s
#Sorting for Series
s.sort_index()
#Sorting for Data Frame
df = pd.DataFrame(np.arange(12).reshape(3,4),
index = ['Two', 'One', 'Three'],
columns = ['d','a','b','c']
)
df
#sort by index
df.sort_index()
#sort by columns
df.sort_index(axis=1)
```

```
In [2]: import pandas as pd
        import numpy as np
        s = pd.Series(range(5),index = ['e', 'd', 'a', 'b', 'c'])
        s

Out[2]: e    0
        d    1
        a    2
        b    3
        c    4
        dtype: int64

In [3]: s.sort_index()

Out[3]: a    2
        b    3
        c    4
        d    1
        e    0
        dtype: int64
```

In [4]:
```
df = pd.DataFrame(np.arange(12).reshape(3,4),
index = ['Two', 'One', 'Three'],
columns = ['d','a','b','c']
)
df
```

Out[4]:

	d	a	b	c
Two	0	1	2	3
One	4	5	6	7
Three	8	9	10	11

In [5]:
```
df.sort_index()
```

Out[5]:

	d	a	b	c
One	4	5	6	7
Three	8	9	10	11
Two	0	1	2	3

In [6]:
```
df.sort_index(axis=1)
```

Out[6]:

	a	b	c	d
Two	1	2	3	0
One	5	6	7	4
Three	9	10	11	8

Ranking

```
import pandas as pd
df = pd.DataFrame({
"name": ["John","Jane","Emily","Lisa","Matt","Jenny","Adam"],
"current": [92,94,87,82,90,78,84],
"overall":                [184,173,184,201,208,182,185],
"group":["A","B","C","A","A","C","B"]
})
df


df["rank_default"] = df["overall"].rank()
df

df["rank_default_desc"] = df["overall"].rank(ascending=False)
df = df.sort_values(by="rank_default_desc", ignore_index=True)
df
```

In [1]:
```python
import pandas as pd
df = pd.DataFrame({
"name": ["John","Jane","Emily","Lisa","Matt","Jenny","Adam"],
"current": [92,94,87,82,90,78,84],
"overall": [184,173,184,201,208,182,185],
"group":["A","B","C","A","A","C","B"]
})
df
```

Out[1]:

	name	current	overall	group
0	John	92	184	A
1	Jane	94	173	B
2	Emily	87	184	C
3	Lisa	82	201	A
4	Matt	90	208	A
5	Jenny	78	182	C
6	Adam	84	185	B

In [2]:
```python
df["rank_default"] = df["overall"].rank()
df
```

Out[2]:

	name	current	overall	group	rank_default
0	John	92	184	A	3.5
1	Jane	94	173	B	1.0
2	Emily	87	184	C	3.5
3	Lisa	82	201	A	6.0
4	Matt	90	208	A	7.0
5	Jenny	78	182	C	2.0
6	Adam	84	185	B	5.0

In [3]:
```python
df["rank_default_desc"] = df["overall"].rank(ascending=False)
df = df.sort_values(by="rank_default_desc", ignore_index=True)
df
```

Out[3]:

	name	current	overall	group	rank_default	rank_default_desc
0	Matt	90	208	A	7.0	1.0
1	Lisa	82	201	A	6.0	2.0
2	Adam	84	185	B	5.0	3.0
3	John	92	184	A	3.5	4.5
4	Emily	87	184	C	3.5	4.5
5	Jenny	78	182	C	2.0	6.0
6	Jane	94	173	B	1.0	7.0

RESULT:

Thus the above python code was executed and verified successfully.

Experiment 3: **Program to implement linear regression**

AIM:
To implement simple linear regression and multiple linear regression using python

CODE:
Simple Linear Regression **(Prerequisite: Boston dataset)**

```
import pandas as pd
data = pd.read_csv('C:/Users/91979/Downloads/Boston.csv')
data.head()
#Have a glance at the dependent and independent variables
data_=data.loc[:,['LSTAT','MEDV']]
data_.head(5)
#Visualize the change in the variables
import matplotlib.pyplot as plt
data.plot(x='LSTAT', y='MEDV', style = 'o')
plt.xlabel('lstat')
plt.ylabel('medv')
plt.show()

#Divide the data into independent and dependent variables
x=pd.DataFrame(data['LSTAT'])
y=pd.DataFrame(data['MEDV'])
#Split the data into train and test sets
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,random_state=1)
#Shape of the train and test sets
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
# Train the algorithm
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

regressor.fit(x_train, y_train)
#Predicted value
y_pred = regressor.predict(x_test)
y_pred
#Actual value
y_test

```
In [1]: import pandas as pd
        data = pd.read_csv('C:/Users/91979/Downloads/Boston.csv')
        data.head()
```

Out[1]:

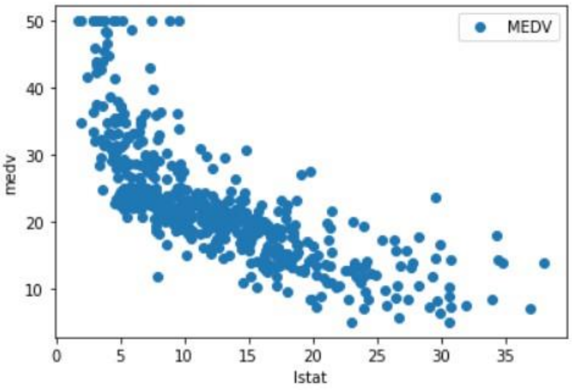
	ï»¿CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	CAT. MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	0
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	1
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	1
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	1

```
In [2]: data_=data.loc[:,['LSTAT','MEDV']]
        data_.head(5)
```

Out[2]:

	LSTAT	MEDV
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

```
In [3]: import matplotlib.pyplot as plt
        data.plot(x='LSTAT', y='MEDV', style = 'o')
        plt.xlabel('lstat')
        plt.ylabel('medv')
        plt.show()
```




```
In [4]: x=pd.DataFrame(data['LSTAT'])
        y=pd.DataFrame(data['MEDV'])
```

```
In [5]: 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,random_state=1)
```

```
In [6]: print(x_train.shape)
        print(x_test.shape)
        print(y_train.shape)
        print(y_test.shape)

        (404, 1)
        (102, 1)
        (404, 1)
```

```
In [7]: from sklearn.linear_model import LinearRegression
        regressor = LinearRegression()
```

```
In [8]: regressor.fit(x_train, y_train)
```

```
Out[8]: LinearRegression()
```

```
In [9]: y_pred = regressor.predict(x_test)
        y_pred
```

```
Out[9]: array([[27.37411725],
               [27.69766325],
               [16.95593597],
               [26.84719947],
               [24.91516763],
               [24.05545968],
               [29.99021779],
               [22.28057875],
               [17.76942306],
               [26.1908633 ],
               [27.17998965],
               [30.07341533],
               [21.75366098],
               [24.86894677],
               [23.50080939],
               [23.12179836],
               [12.85152382],
```

```
In [10]: y_test
```

Out[10]:

	MEDV
307	28.2
343	23.9
47	16.6
67	22.0
362	20.8
...	...
92	22.9
224	44.8
110	21.7
426	10.2
443	15.4

102 rows × 1 columns

Multiple Linear Regression **(Prerequisite: Boston dataset)**

import pandas as pd

data = pd.read_csv('C:/Users/91979/Downloads/Boston.csv')

data

#Set up dependent and independent variable x = pd.DataFrame(data.iloc[:,:-1])

y = pd.DataFrame(data.iloc[:,-1])

#Divide the data into train and test sets

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,random_state=5)
#Shape of the train and test sets
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
# Train the algorithm
from sklearn.linear_model import
LinearRegression regressor =
LinearRegression() regressor.fit(x_train,
y_train)
#Comparing the predicted value to the actual
value y_pred = regressor.predict(x_test)
y_pred

y_test

In [1]:
```
import pandas as pd
data = pd.read_csv('C:/Users/91979/Downloads/Boston.csv')
data
```

Out[1]:

	ï»¿CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	CAT. MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	0
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	1
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	1
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4	0
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6	0
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9	0
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0	0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9	0

506 rows × 15 columns

In [2]:
```
x = pd.DataFrame(data.iloc[:,:-1])
y = pd.DataFrame(data.iloc[:,-1])
```

In [3]:
```
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,random_state=5)
```

In [4]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(404, 14)
(102, 14)
(404, 1)
(102, 1)
```

In [5]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[5]:
```
LinearRegression()
```

In [6]:
```
y_pred = regressor.predict(x_test)
y_pred
```

Out[6]:
```
array([[ 7.21008983e-01],
       [ 3.98536327e-01],
       [ 1.43949175e-01],
       [ 4.28725711e-02],
       [ 5.99398593e-01],
       [-3.42744555e-02],
       [ 1.70409142e-01],
```

In [7]:
```
y_test
```

Out[7]:

	CAT. MEDV
226	1
292	0
90	0
373	0
273	1
...	...
349	0
212	0
156	0
480	0
248	0

102 rows × 1 columns

RESULT:

Thus the above python code was executed and verified successfully.

Experiment 4 **K-Nearest Neighbors Classification**

AIM:
To implement K-Nearest Neighbours classification algorithm in python

CODE:

```
from sklearn.datasets import fetch_california_housing
california_housing = fetch_california_housing(as_frame=True)
df = california_housing.frame
import pandas as pd
df.head()
#Preprocessing data
df["MedHouseValCat"] = pd.qcut(df["MedHouseVal"], 4, retbins=False, labels=[1, 2, 3, 4])
y = df['MedHouseValCat']
X = df.drop(['MedHouseVal', 'MedHouseValCat'], axis = 1)
#Split data to train and test sets
from sklearn.model_selection import train_test_split
SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=SEED)
#Feature scaling for classification
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
#Training and predicting data
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
#Evaluating KNN
acc = classifier.score(X_test, y_test)
```

```
print(acc)
#Finding best K for KNN and plotting
from  sklearn.metrics  import  f1_score
f1s = []
for i in range(1, 40):
    knn    =    KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    f1s.append(f1_score(y_test,      pred_i,      average='weighted'))
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
plt.plot(range(1,      40),      f1s,linestyle='dashed',      marker='o')
plt.title('F1 Score K Value')
plt.xlabel('K    Value')
plt.ylabel('F1   Score')
#Classification report
from     sklearn.metrics     import     classification_report
classifier15   =   KNeighborsClassifier(n_neighbors=15)
classifier15.fit(X_train, y_train)
y_pred15     =     classifier15.predict(X_test)
print(classification_report(y_test, y_pred15))
```

In [1]:
```
from sklearn.datasets import fetch_california_housing
california_housing = fetch_california_housing(as_frame=True)
df = california_housing.frame
import pandas as pd
df.head()
```

Out[1]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

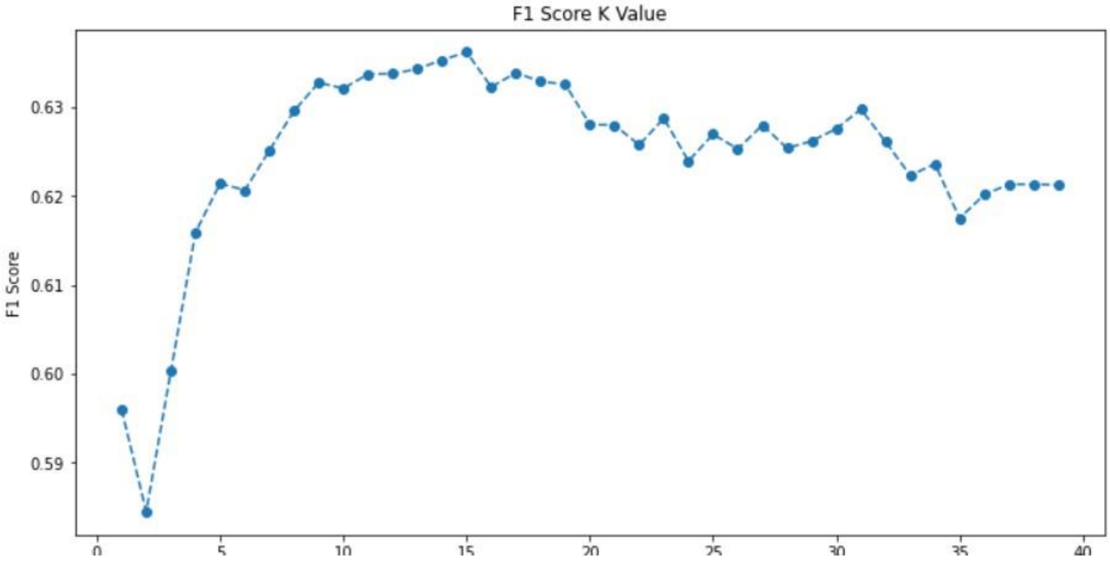
In [2]:
```
df["MedHouseValCat"] = pd.qcut(df["MedHouseVal"], 4, retbins=False, labels=[1, 2, 3, 4])
y = df['MedHouseValCat']
X = df.drop(['MedHouseVal', 'MedHouseValCat'], axis = 1)
```

In [3]:
```
from sklearn.model_selection import train_test_split
SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=SEED)
```

In [4]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [5]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
In [7]: from sklearn.metrics import f1_score
        f1s = []
        for i in range(1, 40):
            knn = KNeighborsClassifier(n_neighbors=i)
            knn.fit(X_train, y_train)
            pred_i = knn.predict(X_test)
            f1s.append(f1_score(y_test, pred_i, average='weighted'))
        import matplotlib.pyplot as plt
        plt.figure(figsize=(12, 6))
        plt.plot(range(1, 40), f1s,linestyle='dashed', marker='o')
        plt.title('F1 Score K Value')
        plt.xlabel('K Value')
        plt.ylabel('F1 Score')

Out[7]: Text(0, 0.5, 'F1 Score')
```




```
In [9]: from sklearn.metrics import classification_report
        classifier15 = KNeighborsClassifier(n_neighbors=15)
        classifier15.fit(X_train, y_train)
        y_pred15 = classifier15.predict(X_test)
        print(classification_report(y_test, y_pred15))

              precision    recall  f1-score   support

           1       0.77      0.79      0.78      1292
           2       0.52      0.58      0.55      1283
           3       0.51      0.53      0.52      1292
           4       0.77      0.64      0.70      1293

    accuracy                           0.63      5160
   macro avg       0.64      0.63      0.64      5160
weighted avg       0.64      0.63      0.64      5160
```

RESULT:
Thus the above python code was executed and verified successfully.

Experiment 5a
Data distribution using box and scatter plot

AIM:
To visualize data distribution using box and scatter plot

CODE:
Data distributions using scatter plot **(Prerequisite: Iris dataset)**

```
import pandas as pd
import  matplotlib.pyplot  as  plt
import seaborn as sns
data  =  pd.read_csv('C:/Users/91979/Downloads/Iris.csv')
data

plt.scatter(data['SepalLengthCm'],data['SepalWidthCm'])
plt.xlabel('Sepal length')
plt.ylabel('Sepal                    width')
plt.title('Scatter  plot  on  Iris  dataset')
sns.set_style("whitegrid")
```

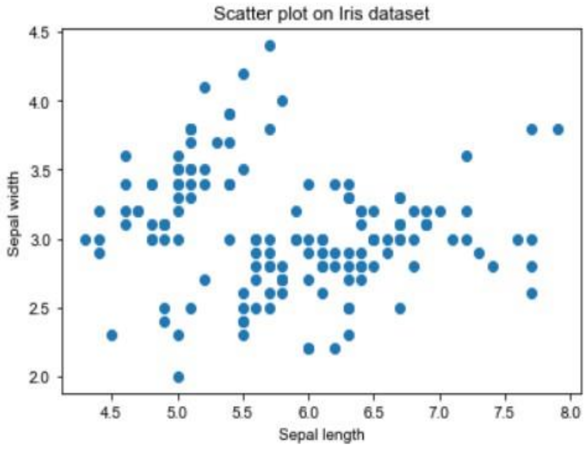
```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        data = pd.read_csv('C:/Users/91979/Downloads/Iris.csv')
        data
```

Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

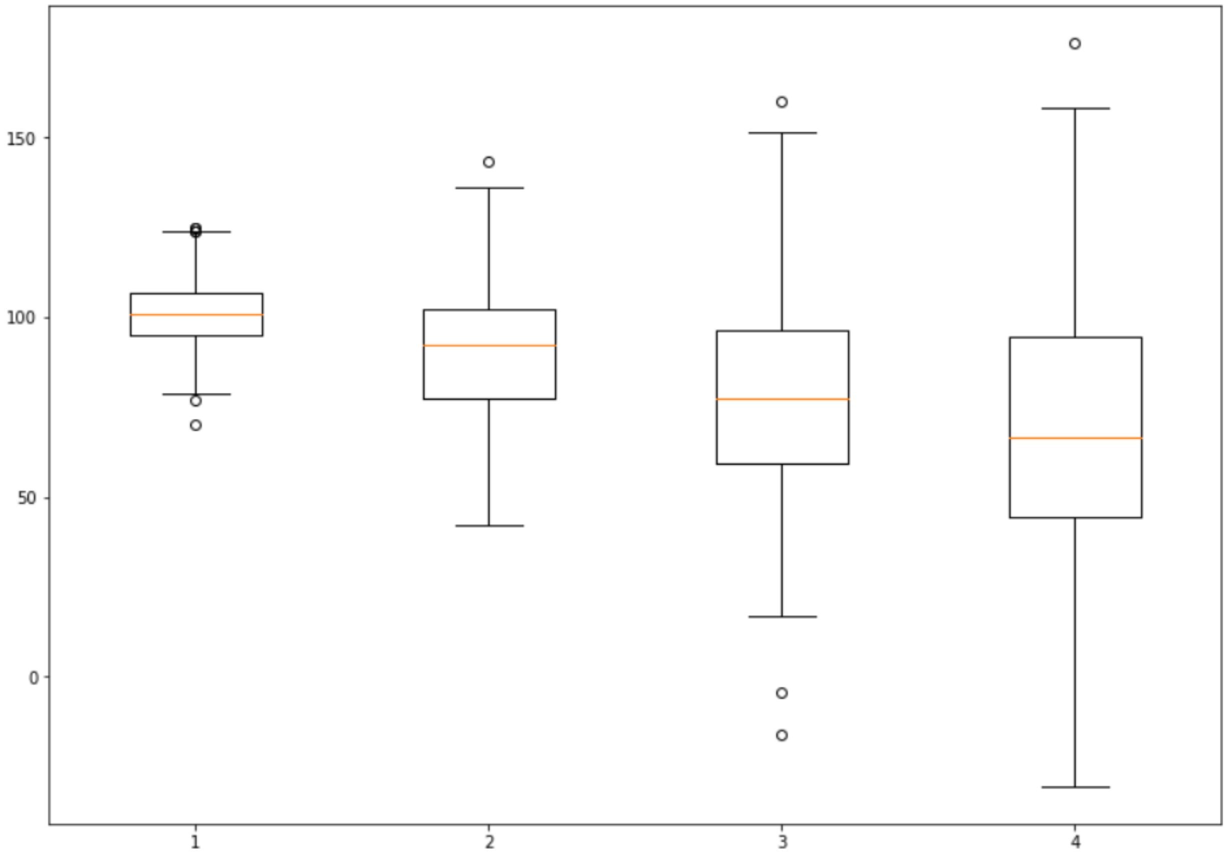
150 rows × 6 columns

```
In [2]: plt.scatter(data['SepalLengthCm'],data['SepalWidthCm'])
        plt.xlabel('Sepal length')
        plt.ylabel('Sepal width')
        plt.title('Scatter plot on Iris dataset')
        sns.set_style("whitegrid")
```



Data distributions using box plot **(Prerequisite: Iris dataset)**

```
import matplotlib.pyplot as plt
import numpy as np
# Creating dataset
np.random.seed(10)
data_1 = np.random.normal(100, 10, 200)
data_2 = np.random.normal(90, 20, 200)
data_3 = np.random.normal(80, 30, 200)
data_4 = np.random.normal(70, 40, 200)
data = [data_1, data_2, data_3, data_4]

fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(data)
plt.show()
```



RESULT:
Thus the above python code was executed and verified successfully.

Experiment 5 b
Finding outliers using plot

AIM:
To visualize the outliers using plot

CODE: **(Prerequisite: Uber dataset)**

import pandas as pd
import numpy as np
import plotly.express as px
df = pd.read_csv('C:/Users/91979/Downloads/Uber.csv')
df
#Find outliers
df.describe()[['fare_amount','passenger_count']]
#Visualising outliers using box plot
fig = px.box(df,y = 'fare_amount')
fig.show()

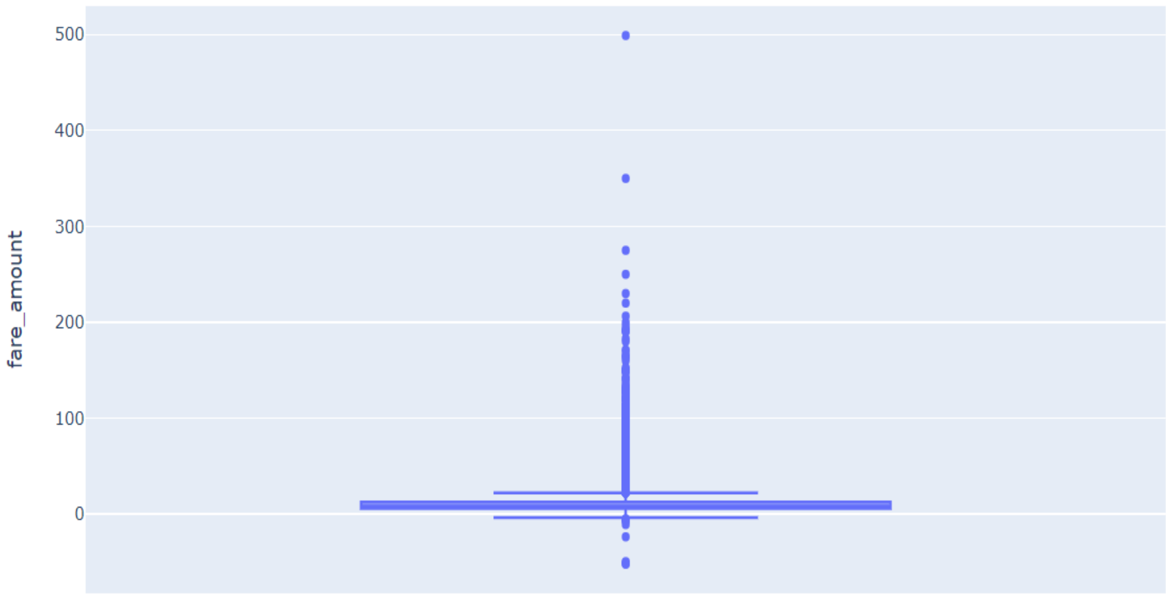
In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
df = pd.read_csv('C:/Users/91979/Downloads/Uber.csv')
df
```

Out[1]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5
...	...	...	...	...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.986525	40.740297	1
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-74.006672	40.739620	1
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.858957	40.692588	2
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	-73.983215	40.695415	1
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	-73.985508	40.768793	1

200000 rows × 9 columns

In [2]:
```
df.describe()[['fare_amount','passenger_count']]
#Visualising outliers using box plot
fig = px.box(df,y = 'fare_amount')
fig.show()
```



RESULT:
Thus the above python code was executed and verified successfully.

Experiment 5 c **Plot the histogram, bar chart & pie chart on sample data**

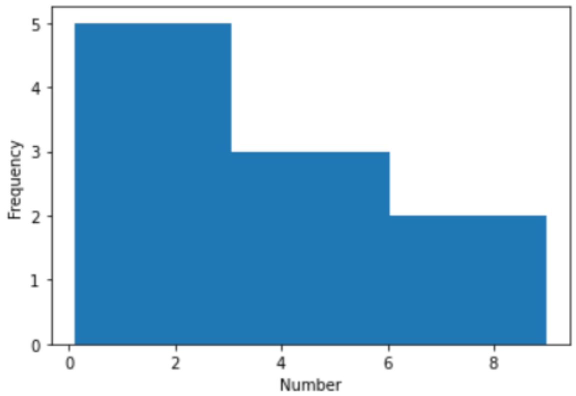
AIM:
To plot the histogram, bar chart & pie chart on sample data

CODE:

#Histogram

```
import matplotlib.pyplot as plt
numbers = [0.1, 0.5, 1, 1.5, 2, 4, 5.5, 6, 8, 9]
plt.hist(numbers,    bins    =    3)
plt.xlabel("Number")
plt.ylabel("Frequency")
plt.show()
```

```
In [1]: import matplotlib.pyplot as plt
        numbers = [0.1, 0.5, 1, 1.5, 2, 4, 5.5, 6, 8, 9]
        plt.hist(numbers, bins = 3)
        plt.xlabel("Number")
        plt.ylabel("Frequency")
        plt.show()
```

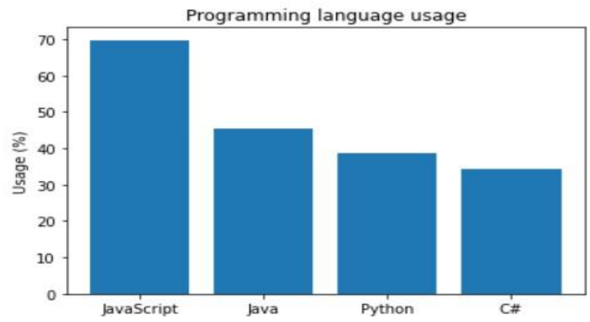


#Barchart

```
import matplotlib.pyplot as plt
# Our data
labels  =  ["JavaScript",  "Java",  "Python",  "C#"]
usage = [69.8, 45.3, 38.8, 34.4]
#  Generating  the  y  positions.
y_positions  =  range(len(labels))
#    Creating    our    bar    plot
plt.bar(y_positions, usage)
```

plt.xticks(y_positions, labels)
plt.ylabel("Usage (%)")
plt.title("Programming language usage")
plt.show()

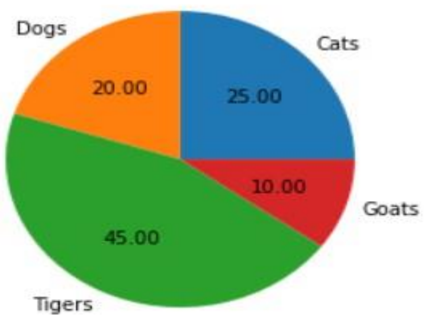
```
In [2]: import matplotlib.pyplot as plt
        # Our data
        labels = ["JavaScript", "Java", "Python", "C#"]
        usage = [69.8, 45.3, 38.8, 34.4]
        # Generating the y positions.
        y_positions = range(len(labels))
        # Creating our bar plot
        plt.bar(y_positions, usage)
        plt.xticks(y_positions, labels)
        plt.ylabel("Usage (%)")
        plt.title("Programming language usage")
        plt.show()
```



#Piechart

import matplotlib.pyplot as plt
sizes = [25, 20, 45, 10]
labels = ["Cats", "Dogs", "Tigers", "Goats"]
plt.pie(sizes, labels = labels, autopct = "%.2f")
plt.show()

```
In [3]: import matplotlib.pyplot as plt
        sizes = [25, 20, 45, 10]
        labels = ["Cats", "Dogs", "Tigers", "Goats"]
        plt.pie(sizes, labels = labels, autopct = "%.2f")
        plt.show()
```



RESULT:
Thus the above python code was executed and verified successfully.

Experiment 6a **Corelation matrix**

AIM:
To find the corelation matrix

CODE:
```
import pandas as pd
# Collect data
data = {
   'x': [45, 37, 42, 35, 39],
   'y': [38, 31, 26, 28, 33],
   'z': [10, 15, 17, 21, 12]
}
# Form dataframe
dataframe = pd.DataFrame(data, columns=['x', 'y', 'z'])
print("Dataframe is : ")
print(dataframe)
# Form correlation matrix
matrix = dataframe.corr()
print("Correlation matrix is : ")
print(matrix)
```

RESULT:
```
Dataframe is :
    x   y   z
0  45  38  10
1  37  31  15
2  42  26  17
3  35  28  21
4  39  33  12
Correlation matrix is :
          x         y         z
x  1.000000  0.518457 -0.701886
y  0.518457  1.000000 -0.860941
z -0.701886 -0.860941  1.000000
```
Thus the above python code was executed and verified successfully.

Experiment 6b Plot the correlation plot on dataset and visualize

AIM:

To plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data

CODE: **(Prerequisite: Iris dataset)**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
dataframe = pd.read_csv("C:/Users/91979/Downloads/Iris.csv")
dataframe

sns.FacetGrid(dataframe, hue="Species", size=5) \
   .map(plt.scatter, "SepalLengthCm", "SepalWidthCm") \
   .add_legend()


corr = dataframe.corr()
sns.heatmap(corr,
        xticklabels=corr.columns.values,
        yticklabels=corr.columns.values)
plt.show()
```

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
dataframe = pd.read_csv("C:/Users/91979/Downloads/Iris.csv")
dataframe
```

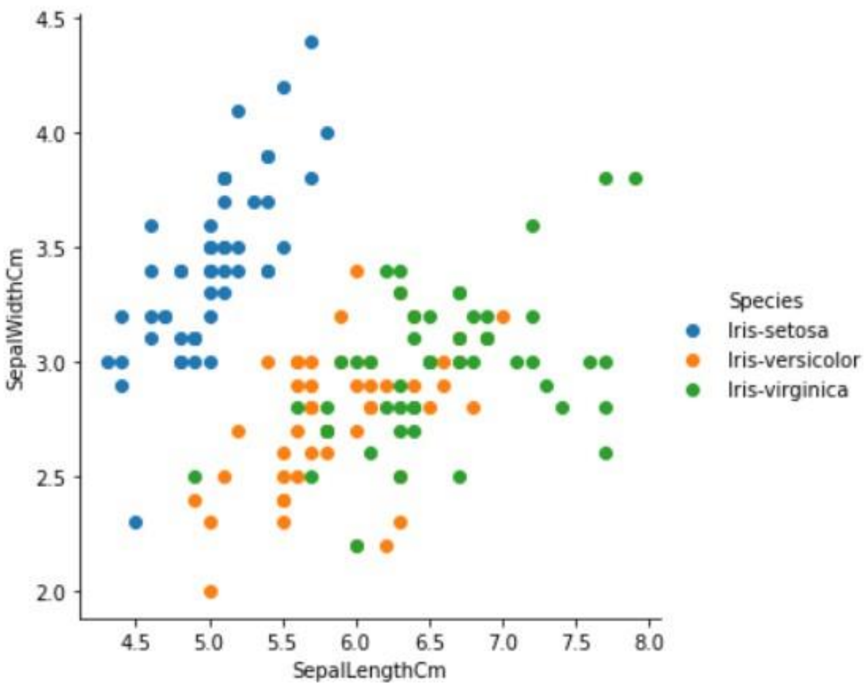
Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

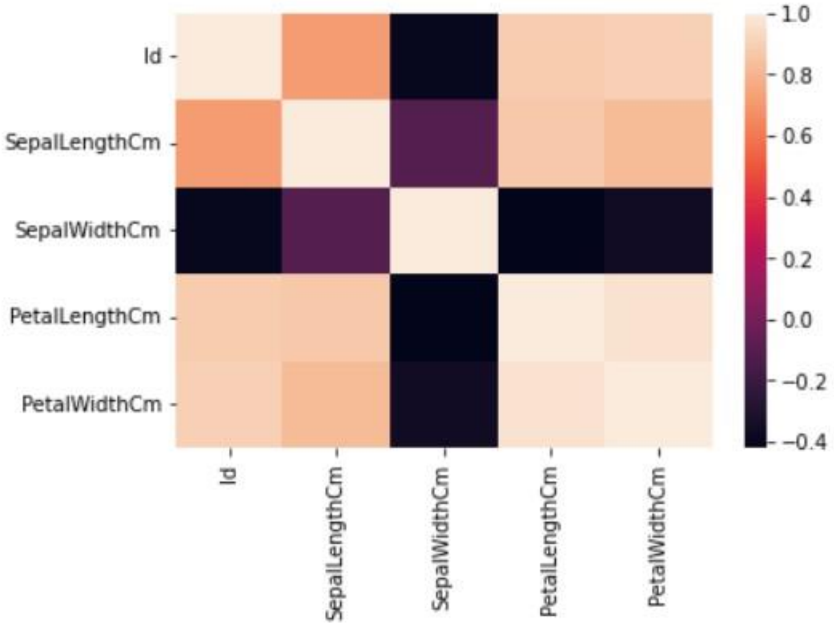
150 rows × 6 columns

In [2]:
```
sns.FacetGrid(dataframe, hue="Species", size=5) \
   .map(plt.scatter, "SepalLengthCm", "SepalWidthCm") \
   .add_legend()
```

Out[2]: <seaborn.axisgrid.FacetGrid at 0x1aadc298850>



In [3]:
```
corr = dataframe.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
plt.show()
```



RESULT:
Thus the above python code was executed and verified successfully.

Experiment 6c **Analysis of covariance: variance (ANOVA)**

AIM:
To analyse covariance, variance (ANOVA), if data have categorical variables on iris data

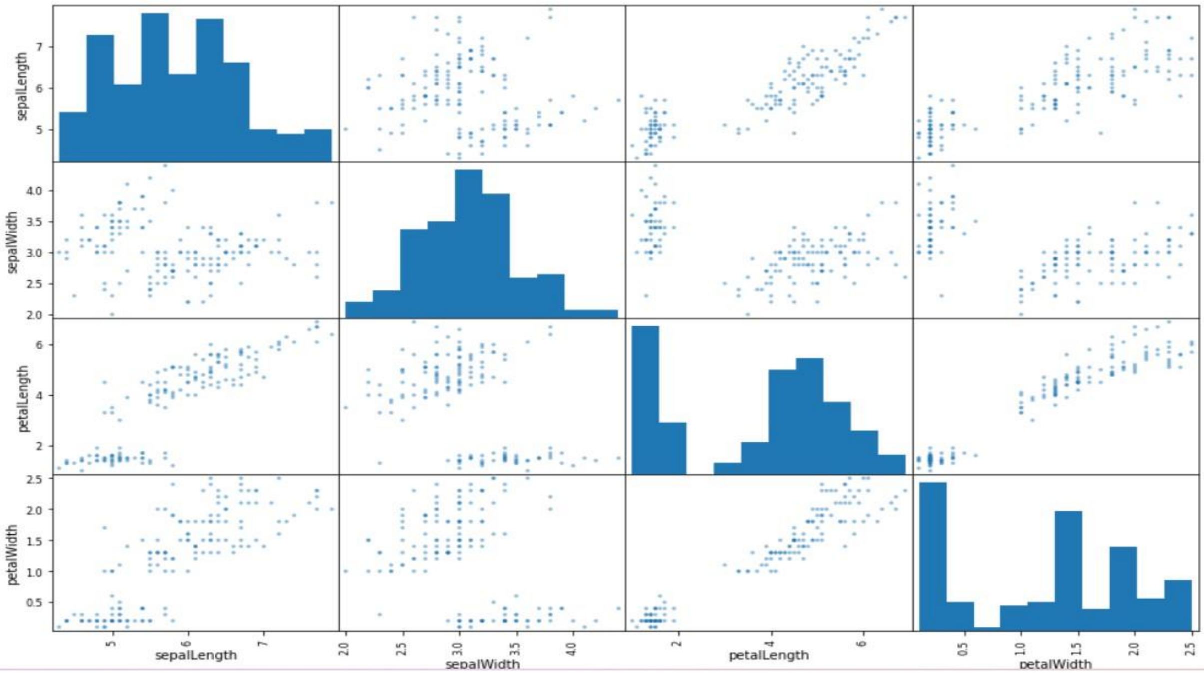
CODE:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
import pandas as pd
import seaborn as sns
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import SelectKBest
from scipy.stats import shapiro
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.sandbox.stats.multicomp import TukeyHSDResults
from statsmodels.graphics.factorplots import interaction_plot
from pandas.plotting import scatter_matrix
iris=load_iris()
iris.target

dataframe_iris=pd.DataFrame(iris.data,columns=['sepalLength','sepalWidth','petalLength','petalWidth'])
dataframe_iris.shape
dataframe_iris1=pd.DataFrame(iris.target,columns=['target'])
dataframe_iris1.shape
scatter_matrix(dataframe_iris[['sepalLength','sepalWidth','petalLength','petalWidth']],figsize=(15,10))
plt.show()
```

```
ID=[]
for i in range(0,150):
    ID.append(i)
dataframe=pd.DataFrame(ID,columns=['ID'])
dataframe_iris_new=pd.concat([dataframe_iris,dataframe_iris1,dataframe],axis=1)
dataframe_iris_new.columns
fig = interaction_plot(dataframe_iris_new.sepalWidth,dataframe_iris_new.target,
            dataframe_iris_new.ID,colors=['red','blue','green'], ms=12)
dataframe_iris_new.info()
dataframe_iris_new.describe()
print(dataframe_iris_new['sepalWidth'].groupby(dataframe_iris_new['target']).mean())
dataframe_iris_new.mean()
stats.shapiro(dataframe_iris_new['sepalWidth'][dataframe_iris_new['target']])
p_value=stats.levene(dataframe_iris_new['sepalWidth'],dataframe_iris_new['target'])
p_value

F_value,P_value=stats.f_oneway(dataframe_iris_new['sepalWidth'],dataframe_iris_new['target'])
print("F_value=",F_value,",","P_value=",P_value)

if F_value>1.0:
    print("******SAMPLES HAVE DIFFERENT MEAN******")
else:
    print("******SAMPLES HAVE EQUAL MEAN******")


if P_value<0.05:
```

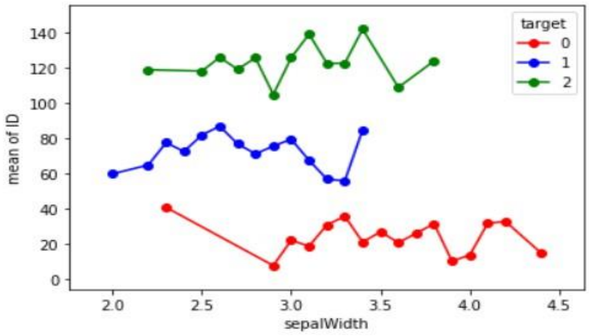
    print("******REJECT NULL HYPOTHESIS******")
else:
    print("******ACCEPT NULL HYPOTHESIS******")

tukey = pairwise_tukeyhsd(endog=dataframe_iris_new['sepalWidth'], groups=dataframe_iris_new['target'], alpha=0.05)
print(tukey)

```
In [4]: scatter_matrix(dataframe_iris[['sepalLength','sepalWidth','petalLength','petalWidth']],figsize=(15,10))
        plt.show()
```




```
In [6]: fig = interaction_plot(dataframe_iris_new.sepalWidth,dataframe_iris_new.target,
                               dataframe_iris_new.ID,colors=['red','blue','green'], ms=12)
```




```
In [7]: dataframe_iris_new.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepalLength  150 non-null    float64
 1   sepalWidth   150 non-null    float64
 2   petalLength  150 non-null    float64
 3   petalWidth   150 non-null    float64
 4   target       150 non-null    int32  
 5   ID           150 non-null    int64  
dtypes: float64(4), int32(1), int64(1)
memory usage: 6.6 KB
```

In [8]:
```
dataframe_iris_new.describe()
```

Out[8]:

	sepalLength	sepalWidth	petalLength	petalWidth	target	ID
count	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000	74.500000
std	0.828066	0.435866	1.765298	0.762238	0.819232	43.445368
min	4.300000	2.000000	1.000000	0.100000	0.000000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000	37.250000
50%	5.800000	3.000000	4.350000	1.300000	1.000000	74.500000
75%	6.400000	3.300000	5.100000	1.800000	2.000000	111.750000
max	7.900000	4.400000	6.900000	2.500000	2.000000	149.000000

In [9]:
```
print(dataframe_iris_new['sepalWidth'].groupby(dataframe_iris_new['target']).mean())
```

```
target
0    3.428
1    2.770
2    2.974
Name: sepalWidth, dtype: float64
```

In [10]:
```
dataframe_iris_new.mean()
```

Out[10]:
```
sepalLength     5.843333
sepalWidth      3.057333
petalLength     3.758000
petalWidth      1.199333
target          1.000000
ID             74.500000
dtype: float64
```

In [12]:
```
F_value,P_value=stats.f_oneway(dataframe_iris_new['sepalWidth'],dataframe_iris_new['target'])
print("F_value=",F_value,",","P_value=",P_value)
```

```
F_value= 737.2872570149498 , P_value= 1.418242288711535e-82
```

In [13]:
```
if F_value>1.0:
    print("******SAMPLES HAVE DIFFERENT MEAN******")
else:
    print("******SAMPLES HAVE EQUAL MEAN******")
```

```
******SAMPLES HAVE DIFFERENT MEAN******
```

In [14]:
```
if P_value<0.05:
    print("******REJECT NULL HYPOTHESIS******")
else:
    print("******ACCEPT NULL HYPOTHESIS******")
```

```
******REJECT NULL HYPOTHESIS******
```

In [15]:
```
tukey = pairwise_tukeyhsd(endog=dataframe_iris_new['sepalWidth'], groups=dataframe_iris_new['target'], alpha=0.05)
print(tukey)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj   lower   upper  reject
----------------------------------------------------
     0      1   -0.658    0.0 -0.8189 -0.4971   True
     0      2   -0.454    0.0 -0.6149 -0.2931   True
     1      2    0.204 0.0088  0.0431  0.3649   True
----------------------------------------------------
```

RESULT:

Thus the above python code was executed and verified successfully.

Experiment 7: Behavioural analysis of customers for any online purchase model

AIM:
To perform behavioural analysis of customers for any online purchase model

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('C:/Users/91979/Downloads/Social_Network_Ads.csv')
dataset

X = dataset.iloc[:, 2:4].values
y = dataset.iloc[:, -1].values

#split the dataset into train and test
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)
print(X_train)
print(X_test)
print(y_train)
print(y_test)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Build model with logistic regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
```

```
classifier.fit(X_train, y_train)


#Test result prediction
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1))

#Accuracy score with confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

#Predicing new results

```
age=int(input("Enter the age: "))
salary = int(input("Enter the estimated salary: "))
result = classifier.predict(sc.transform([[age,salary]]))
if result==[1]:
  print("Yay! This customer can buy a car!")
else:
  print("Sorry! It seems this customer won't buy a car")
```

```
In [18]: #Predicing new results
age=int(input("Enter the age: "))
salary = int(input("Enter the estimated salary: "))
result = classifier.predict(sc.transform([[age,salary]]))
if result==[1]:
  print("Yay! This customer can buy a car!")
else:
  print("Sorry! It seems this customer won't buy a car")

Enter the age: 45
Enter the estimated salary: 100000
Yay! This customer can buy a car!
```

RESULT:

Thus the above python code was executed and verified successfully.

Experiment 8:
Analysis of tweet and retweet data to identify the spread of fake news

AIM:
To analyse tweet and retweet data to identify the spread of fake news

INTRODUCTION
Fake news can confuse many people in the area of politics, culture, healthcare, etc. Fake news refers to news containing misleading or fabricated contents that are actually groundless; they are intentionally exaggerated or provide false information. As such, fake news can distort reality and cause social problems, such as self-misdiagnosis of medical issues. Many academic researchers have been collecting data from social and medical media, which are sources of various information flows, and conducting studies to analyse and detect fake news. However, in the case of conventional studies, the features used for analysis are limited, and the consideration for newly added features of social media is lacking

Twitter

The name Twitter originated from the word 'tweet', a bird's chirping sound. Its service was launched in 2006, and it has become a highly recognised global social media platform, along with Facebook. A Twitter user can become a follower of a certain user, and based on this feature, a person's social recognition, status, and influence in a certain area can be checked. When a Twitter user has many followers, it means that the user is highly recognised in the area he/she belongs to. Tweets by an influential Twitter user have strong influence in terms of information delivery in Twitter because they are highly likely to be read by many people.

Major Functions of Twitter

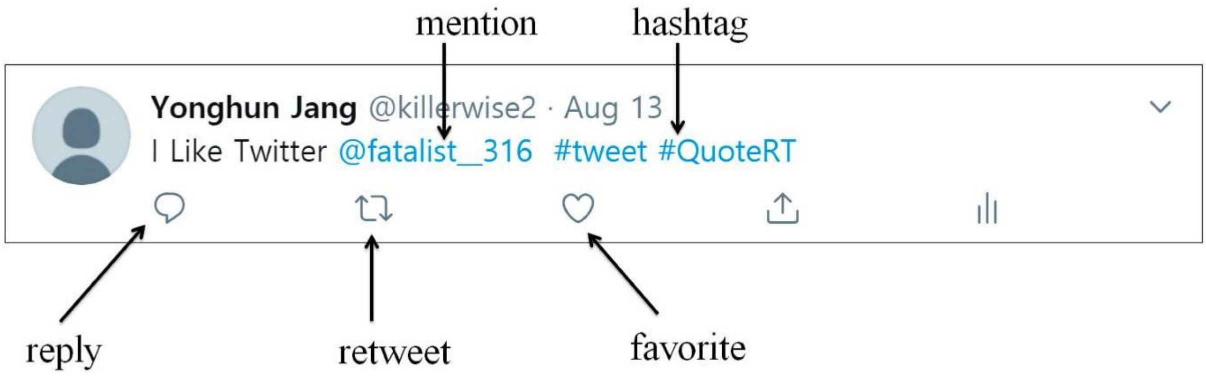
In Twitter, a user can follow a certain user using the Follow button, and convey or share opinions with followers through features such as Tweet, Retweet, and Mention, and express interest in a certain Tweet using the Like button.

Follow

In Twitter, a relationship between each user is made through a feature called Follow. When user A follows user B, A becomes a follower of B, and B becomes a part of A's following. When A and B follow each other, they become virtual friends. In another popular social media platform, Facebook, users have to build friendship with each other in order to exchange information, but in Twitter, information can be shared by a certain user by simply following that user. When following, a special qualification or permission from a corresponding user is not required. Twitter users can continue to receive information from each following user unless they are blocked by their following users. In addition, on Twitter, like other social media, it is possible to socialize and share information through each other.

Tweeting

Tweeting refers to sharing one's thoughts or opinions with their followers. A text message of up to 280 characters can be posted, and in addition, links, photos, and videos can be uploaded. The followers who see a Tweet of a user can use additional features such as Like (heart), Retweet, and Reply.



Retweet

A Retweet is often expressed as the term RT, and its purpose is to re-share an already-shared Tweet to one's followers while maintaining the original writer and content. In general, followers who have accessed a Tweet express their interest in the information to other people through Retweets. As Retweets do not contain one's own comments and are usually used when expressing agreement with or interest in the original Tweets, users do not usually Retweet when a Tweet contains information they do not like or are not interested in. Information is generated through Tweets, but in general, information is spread using Retweets

Quote Retweet

Quote Retweet is an added feature of Twitter that was introduced in 2015. The conventional Retweet only posts an original Tweet a follower read to his/her own followers without writing any comment. In contrast, Quote Retweet lets a follower post an original Tweet to his/her own followers and at the same time, write his/her comment regarding the original Tweet

An example of using Quote Retweet.

Fake News

Yellow journalism has existed for a long time. When social media was advancing rapidly in the 2010s, it was exploited to distribute completely fabricated information, which was disguised in the form of journalism.

Recently, the use of the expression ‘fake news’ has also sharply increased, as the acts of spreading unverified, inaccurate ‘news’ or maliciously distorted information have been prevalent in the form of news/newspaper articles through social media. Fake news became a widespread expression familiar to even ordinary people especially after Donald Trump, who was elected the 45th US president in 2016, claimed that some news reports were fake news.

Fake news and yellow journalism have some similarities; they use news report formats to spread information and gain public trust

People tend to accept only what they want to believe, and if they repeatedly exposed to the wrong information, they are very likely to accept it. Generally, materials related to fake news spreading in social media have the following commonalities: satire, parody, misinterpretation, foment, and heavily biased contents

In the 2016 US presidential election, fake news had enormous impact on the election, and at the time, a large fraction of the news reports mentioned in social media were proven to be fake news . Fake news has become a serious issue globally, and many countries are taking measures to introduce laws and countermeasures against fake news, but effective solutions have yet to be presented. Furthermore, the providers of social media, such as Twitter and Facebook, that are agents of information spread have endeavoured to minimise the problem through a reporting feature, but there is a fairly high possibility that the reporting function can be misused. Furthermore, it has become increasingly more difficult to identify fake news because the ways of spreading fake news is evolving every day.

RESULT:

Tweet and retweet data was analysed to identify the spread of fake news.

Experiment 9: Develop an application to a Text Data Analysis using Tensorflow

AIM:
To develop an application to a TextData Analysis using Tensorflow

CODE:

```
dataset_dir <- file.path("aclImdb")
list.files(dataset_dir)

remove_dir <- file.path(train_dir, 'unsup')
unlink(remove_dir, recursive = TRUE)

batch_size <- 32
seed <- 42
raw_train_ds <- text_dataset_from_directory(
'aclImdb/train',
 batch_size = batch_size,
validation_split = 0.2,
subset = 'training',
seed = seed )

batch <- raw_train_ds %>%
reticulate::as_iterator() %>%
coro::collect(n = 1)
batch[[1]][[1]][1]

batch[[1]][[2]][1]
tf.Tensor(0, shape=(), dtype=int32)


cat("Label 0 corresponds to", raw_train_ds$class_names[1])
cat("Label 1 corresponds to", raw_train_ds$class_names[2])
raw_val_ds <- text_dataset_from_directory(
```

```
  'aclImdb/train',
  batch_size = batch_size,
  validation_split = 0.2,
  subset = 'validation',
  seed = seed
)
raw_test_ds <- text_dataset_from_directory(
  'aclImdb/test',
  batch_size = batch_size
)
# creating a regex with all punctuation characters for replacing. re <- reticulate::import("re")
punctuation <- c("!", "\\", "\"", "#", "$", "%", "&", "'", "(", ")", "*", "+", ",", "-", ".", "/", ":", ";", "<",
"=", ">", "?", "@", "[", "\\", "\\", "]", "^", "_", "`", "{", "|", "}", "~") punctuation_group <-
punctuation %>%
sapply(re$escape) %>%
paste0(collapse = "") %>%
sprintf("[%s]", .) custom_standardization <- function(input_data) { lowercase <-
tf$strings$lower(input_data)
stripped_html <- tf$strings$regex_replace(lowercase, '<br />', ' ') tf$strings$regex_replace(
stripped_html, punctuation_group,
 "" ) }


max_features <- 10000
sequence_length <- 250
vectorize_layer <- layer_text_vectorization(
standardize = custom_standardization,
max_tokens = max_features,
output_mode = "int",
output_sequence_length = sequence_length )

# Make a text-only dataset (without labels), then call adapt
train_text <- raw_train_ds %>%
dataset_map(function(text, label) text)
vectorize_layer %>% adapt(train_text)
```

```
vectorize_text <- function(text, label) {
text <- tf$expand_dims(text, -1L)
list(vectorize_layer(text), label) }
vectorize_text <- function(text, label) {
  text <- tf$expand_dims(text, -1L)
  list(vectorize_layer(text), label)
}
# retrieve a batch (of 32 reviews and labels) from the dataset
batch <- reticulate::as_iterator(raw_train_ds) %>%
reticulate::iter_next()
first_review <- as.array(batch[[1]][1])
first_label <- as.array(batch[[2]][1])
cat("Review:\n", first_review)


cat("Label: ", raw_train_ds$class_names[first_label+1])
Label:  neg
cat("Vectorized review: \n")
Vectorized review:
print(vectorize_text(first_review, first_label))
[[1]]
tf.Tensor(
[[  86   17  260    2  222    1  571   31  229   11 2418    1   51   22
    25  404  251   12  306  282    0    0    0    0    0    0    0    0

     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]], shape=(1, 250), dtype=int64)


cat("9257 ---> ",get_vocabulary(vectorize_layer)[9257 + 1])

9257 --->  recipe
cat(" 15 ---> ",get_vocabulary(vectorize_layer)[15 + 1])
 15 --->  for
```

cat("Vocabulary size: " , length(get_vocabulary(vectorize_layer)))
Vocabulary size: 10000

train_ds <- raw_train_ds %>% dataset_map(vectorize_text)
val_ds <- raw_val_ds %>% dataset_map(vectorize_text)
test_ds <- raw_test_ds %>% dataset_map(vectorize_text)
Configure the dataset for performance
dataset_cache() keeps data in memory after it’s loaded off disk.
dataset_prefetch() overlaps data preprocessing and model execution while training
#Create the model
model <- keras_model_sequential() %>%
  layer_embedding(max_features + 1, embedding_dim) %>%
  layer_dropout(0.2) %>%
  layer_global_average_pooling_1d() %>%
  layer_dropout(0.2) %>%
  layer_dense(1)

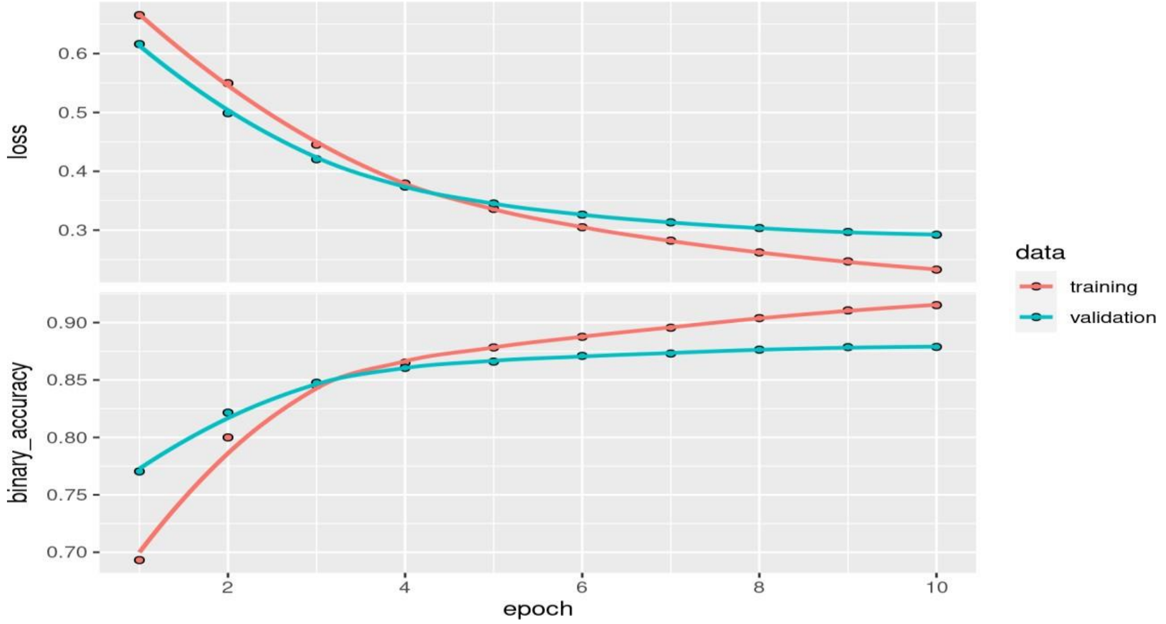
summary(model)
Model: "sequential"
____

 Layer (type)   Output Shape   Param #
==
 embedding (Embedding)   (None, None, 16)   160016
 dropout_1 (Dropout)   (None, None, 16)   0
 global_average_pooling1d (Global (None, 16)   0
 AveragePooling1D)
 dropout (Dropout)   (None, 16)   0
 dense (Dense)   (None, 1)   17

===
====
Total params: 160,033
Trainable params: 160,033
Non-trainable params: 0

#Evaluate the model
model %>% evaluate(test_ds)

loss binary_accuracy
0.3104765 0.8734400
#Create a plot of accuracy and loss over time
model %>% fit()
as.data.frame(history)
plot(history)



#Export the model

export_model <- keras_model_sequential() %>%
vectorize_layer() %>%
model() %>%
layer_activation(activation = "sigmoid")
export_model %>% compile(
loss = loss_binary_crossentropy(from_logits = FALSE),
optimizer = "adam",
metrics = 'accuracy'

```
) # Test it with `raw_test_ds`, which yields raw strings export_model %>% evaluate(raw_test_ds)
   loss      accuracy
0.3104761 0.8734400
#Inference on new data

examples <- c( "The movie was great!",
"The movie was okay.",
"The movie was terrible..."
)
predict(export_model, examples)
          [,1]
[1,] 0.6113217
[2,] 0.4314919
[3,] 0.3499118
```

RESULT:

Thus the above python code was executed and verified successfully.

Experiment 10: Develop an application to Analyse the twitter data with Tweepy

AIM:

To develop an application to a analyse the twitter data with Tweepy

CODE:

Interacting with the API through Tweepy

```
import os
import tweepy as tw
import pandas as pd
#Enter keys
consumer_key= 'XXX'
consumer_secret= 'XXX'
access_token= 'XXX'
access_token_secret= 'XXX'
#Authentification process
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)
```

```
# Define the search term and the date_since date as variables

search_words = "#harassment"

date_since = "2020-07-14"

# Collect tweets

tweets = tw.Cursor(api.search,

        q=search_words,

        lang="en",

        since=date_since).items(1000)

tweets

<tweepy.cursor.ItemIterator at 0x2211566cf28>

# Iterate and print tweets
```

```
for tweet in tweets:

print(tweet.text)

# Collect tweets

tweets = tw.Cursor(api.search,

                       q=search_words,

                       lang="en",

                       since=date_since).items(30)

# Collect a list of tweets

[tweet.text for tweet in tweets]

#Take care of retweets

new_search = search_words + " -filter:retweets"

new_search

'#harassment -filter:retweets'

tweets = tw.Cursor(api.search,
                       q=new_search,

                       lang="en",

                       since=date_since).items(30)

[tweet.text for tweet in tweets]

tweets = tw.Cursor(api.search,

                           q=new_search,

                           lang="en",

                           since=date_since).items(30)

users_locs = [[tweet.user.screen_name, tweet.user.location] for tweet in tweets]

users_locs

#Create a pandas dataframe from a list of tweet data

tweet_text = pd.DataFrame(data=users_locs,

                    columns=['user', "location"])
```

```
tweet_text

#Customize your twitter queries

new_search = "sexualharassment+workplace -filter:retweets"

tweets = tw.Cursor(api.search,

                   q=new_search,

                   lang="en",

                   since='2020-07-14').items(100)

all_tweets = [tweet.text for tweet in tweets]

all_tweets[:5]
```

RESULT:
Thus the above python code was executed and verified successfully.