INDEX

Subject:- CA LAB-VII(A): LAB on Machine Learning

Sr.No.	Name Of The Practical	Date	Remark
1	Introduction to Pycharm, Pandas Library, DataFrames, And Loading CSV File in DataFrame.		
2	Implement the Find-S Inductive Learning algorithm.		
3	Implement the Candidate-Elimination Inductive Learning algorithm.		
4	Write program for linear regression and find parameters like Mean Squared Error		
5.1	Write a program to implement Decision tree using the Python/R/Programming language of your choice		
5.2	Write a program to calculate popular attribute selection measures (ASM) like Information Gain, Gain Ratio, and Gini Index etc. for decision tree.		
6	Implement simple KNN using Euclidean distance in python.		
7	Write a program to implement k-Nearest Neighbour algorithm to classify the iris dataset. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.		
8	Write a Program for Confusion Matrix and calculate Precision, Recall, F-Measure		
9	Write a program for linear regression and find parameters like Sum of Squared Errors (SSE), Total Sum of Squares (SST), R2, Adjusted R2, etc.		
10	Write a program to implement the naïve Bayesian classifier for a sample training dataset stored as a . CSV file. Compute the accuracy of the classifier, considering a few test data sets.		
11.1	Implementing Agglomerative Clustering in Python		
11.2	Write a Program for Fuzzy c-means clustering in Python.		
12	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select the appropriate data set for your experiment and draw graphs.		
13.1	Build a Simple Artificial Neural Network		
13.2	Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.		

Practical in-Charge

Name :- Nilesh Vijay Patil
Roll No. :- 140
Practical No. :- 1
Practical Name :- Introduction to pycharm , Pandas Library, DataFrames, And Loading CSV File in DataFrame

---

```
import pandas as pd
'''pd.__version__'''

df1 = pd.DataFrame({"A": [1, 2, 3], "B": [2, 3, 4]}, index=[0, 1, 2])
print("df1:\n", df1)

df2 = pd.DataFrame({"B": [4, 5, 7], "C": ["x", "y", "z"]}, index=[4, 5, 6])
print("\ndf2:\n", df2)

df3 = df1.combine_first(df2)
print("\n combination of df1 and df2:\n", df3)


classes = pd.Series(["mathematics", "chemistry", "physics", "history", "geography", "german"])
grades = pd.Series([90, 54, 77, 22, 25, 40])
year = pd. Series([2015, 2016, 2017, 2018, 2019, 2020])
df4 = pd. DataFrame({"Classes": classes, "Grades": grades, "Year": year})
print("\n", df4)

# upload a csv file  in sample_data section
# load the .csv in data frame

data_frame = pd.read_csv("C:/Users/nilesh/PycharmProjects/dataset.csv")
print("\n", data_frame)
```

OUTPUT :

```
C:\Users\nilesh\MCA-I_ML\Scripts\python.exe C:/Users/nilesh/PycharmProjects/MCA-I_ML/1_prat.py
df1:
   A  B
0  1  2
1  2  3
2  3  4

df2:
   B  C
4  4  x
5  5  y
6  7  z
```

```
combination of df1 and df2:
     A  B    C
0  1.0  2  NaN
1  2.0  3  NaN
2  3.0  4  NaN
4  NaN  4    x
5  NaN  5    y
6  NaN  7    z

       Classes  Grades  Year
0  mathematics      90  2015
1    chemistry      54  2016
2      physics      77  2017
3      history      22  2018
4    geography      25  2019
5       german      40  2020

     sky  temp humidity water    wind forcast enjoy-sport
0  sunny  warm     high  cool  strong    same         yes
1  sunny  warm     high  warm  strong    same         yes
2  rainy  cold      low  warm    weak  change          no
3  rainy  cold     high  warm    weak  change          no
4  sunny  warm     high  warm  strong    same         yes
5  sunny  cold     high  warm  strong    same          no
6  sunny  warm     high  cool  strong  change          no
7  rainy  cold      low  warm  strong    same         yes

Process finished with exit code 0
```

Name :- Nilesh Vijay Patil
Roll No :- 140
Practical No :- 2
Practical Name :- Implement the Find-S Inductive Learning algorithm.

```
import pandas as pd
import numpy as np
data=pd.read_csv("C:/Users/comp/PycharmProjects/dataset2.csv")
print("Given data set")
print(data)

#making an array of all the attributes
d=np.array(data)[:,:-1]
print("The attributes are:\n",d)

#segrating the target that has positive and negative example
target=np.array(data)[:,-1]
print("The target is:",target)

#traing function to implement find s algoritham
def train(c,t):
    for i,val in enumerate(t):
        if val == "Yes":
            specific_hypothesis=c[i].copy()
            break
    for i,val in enumerate(c):
        if t[i] == "Yes":
            for x in range(len(specific_hypothesis)):
                if val[x] != specific_hypothesis[x]:
                    specific_hypothesis[x] = '?'
                else:
                    pass
    return specific_hypothesis

#obtaining the final hypothesis
print("The final hypothesis is:",train(d,target))
```

OUTPUT:-

C:\Users\comp\PycharmProjects\Mlpract\venv\Scripts\python.exe C:/Users/comp/PycharmProjects/Mlpract/mlfirst.py

Given data set

	sky	air temp	humidity	wind	water	forcast	enjoy_sport
0	sunny	warm	normal	strong	warm	same	Yes
1	sunny	warm	hight	strong	warm	same	Yes
2	rainy	cold	hight	strong	warm	change	No
3	sunny	warm	hight	strong	cool	change	Yes
4	sunny	warm	normal	strong	cool	same	Yes
5	rainy	cold	hight	strong	warm	change	No

The attributes are:

[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'hight' 'strong' 'warm' 'same']

['rainy' 'cold' 'hight' 'strong' 'warm' 'change']

['sunny' 'warm' 'hight' 'strong' 'cool' 'change']

['sunny' 'warm' 'normal' 'strong' 'cool' 'same']

['rainy' 'cold' 'hight' 'strong' 'warm' 'change']]

The target is: ['Yes' 'Yes' 'No' 'Yes' 'Yes' 'No']

The final hypothesis is: ['sunny' 'warm' '?' 'strong' '?' '?']

Process finished with exit code 0

Name :- Nilesh Vijay Patil
Roll No :- 140
Practical No :- 3
Practical Name :- Implement the Candidate-Elimination Inductive Learning algorithm.

```python
import numpy as np
import pandas as pd
data=pd.read_csv('C:/Users/comp/PycharmProjects/dataset2.csv')
concepts=np.array(data.iloc[:,0:-1])
print("\nInstance are:\n",concepts)
target=np.array(data.iloc[:,-1])
print("\nTarget values are:\n",target)
def learn(cocepts,target):
    specific_h=concepts[0].copy()
    print("\nInitialization of specific_h and general_h")
    print("\nSpecific boundary:",specific_h)
    general_h=[["?" for i in range(len(specific_h))]for i in range(len(specific_h))]
    print("\nGeneric boundaries:",general_h)
    for i, h in enumerate(concepts):
        print("\nInstance",i+1, "is", h)
        if target[i]=="Yes":
            print("Instance is positive")
            for x in range(len(specific_h)):
                if h[x] != specific_h[x]:
                    specific_h[x]='?'
                    general_h[x][x]='?'
        else:
            print("Instance is negative")
            for x in range(len(specific_h)):
                if h[x] != specific_h[x] and specific_h[x] !='?':
                    general_h[x][x]= specific_h[x]
                else:
                    general_h[x][x]='?'
                    specific_h[x]='?'
        print("Specific boundary after",i+1,"Instance is",specific_h)
        print("Generic boundary after",i+1,  "Instance is",general_h)
        print("\n")
    indices=[i for i, val in enumerate(general_h) if val == ['?','?','?','?','?','?']]
    for i in indices:
        general_h.remove(['?','?','?','?','?','?'])
    return specific_h, general_h
s_final, g_final=learn(concepts, target)

print("Final specific_h:", s_final, sep="\n")
print("Final general_h:", g_final, sep="\n")
```

OUTPUT:-

C:\Users\comp\PycharmProjects\Mlpract\venv\Scripts\python.exe C:/Users/comp/PycharmProjects/Mlpract/mlsecond.py

Instance are:

[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'hight' 'strong' 'warm' 'same']

['rainy' 'cold' 'hight' 'strong' 'warm' 'change']

['sunny' 'warm' 'hight' 'strong' 'cool' 'change']

['sunny' 'warm' 'normal' 'strong' 'cool' 'same']

['rainy' 'cold' 'hight' 'strong' 'warm' 'change']]

Target values are:

['Yes' 'Yes' 'No' 'Yes' 'Yes' 'No']

Initialization of specific_h and general_h

Specific boundary: ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Generic boundaries: [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 1 is ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Instance is positive

Specific boundary after 1 Instance is ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Generic boundary after 1 Instance is [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 2 is ['sunny' 'warm' 'hight' 'strong' 'warm' 'same']

Instance is positive

Specific boundary after 2 Instance is ['sunny' 'warm' '?' 'strong' 'warm' 'same']

Generic boundary after 2 Instance is [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 3 is ['rainy' 'cold' 'hight' 'strong' 'warm' 'change']

Instance is negative

Specific boundary after 3 Instance is ['sunny' 'warm' '?' '?' '?' 'same']

Generic boundary after 3 Instance is [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]

Instance 4 is ['sunny' 'warm' 'hight' 'strong' 'cool' 'change']

Instance is positive

Specific boundary after 4 Instance is ['sunny' 'warm' '?' '?' '?' '?']

Generic boundary after 4 Instance is [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 5 is ['sunny' 'warm' 'normal' 'strong' 'cool' 'same']

Instance is positive

Specific boundary after 5 Instance is ['sunny' 'warm' '?' '?' '?' '?']

Generic boundary after 5 Instance is [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 6 is ['rainy' 'cold' 'hight' 'strong' 'warm' 'change']

Instance is negative

Specific boundary after 6 Instance is ['sunny' 'warm' '?' '?' '?' '?']

Generic boundary after 6 Instance is [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final specific_h:

['sunny' 'warm' '?' '?' '?' '?']

Final general_h:

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]

Process finished with exit code 0

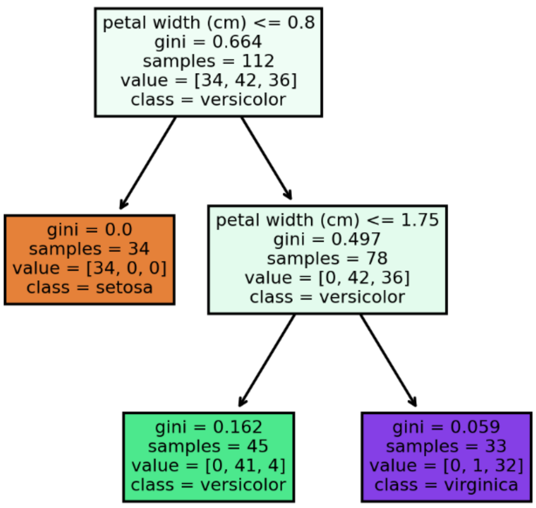
Name :- Nilesh Vijay Patil
Roll No :- 140
Practical No :- 4
Practical Name :- Write a program to implement Decision tree using python programming.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_iris
data_b= load_iris()
df = pd.DataFrame(data_b.data,columns = data_b.feature_names)
df['target'] = data_b.target
#df['target']
print(df)
#print(data_b)
print("Dataset Labels=",data_b.target_names)
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df[data_b.feature_names],
df['target'])
#print(x_train)
#print(x_test)
#print(y_train)
#print(y_test)
clf = DecisionTreeClassifier(max_depth = 2,random_state= 1, criterion='gini')
#'gini'
clf = clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
#print(y_test,y_pred)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
fn = ['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
cn=['setosa','versicolor','virginica']

fig, axes = plt.subplots(nrows = 1, ncols = 1,figsize = (4,4),dpi = 300)
tree.plot_tree(clf, feature_names = fn, class_names = cn,filled = True)
fig.savefig('dstimg.png')
```

Output:-

	sepal length (cm)	sepal width (cm)	...	petal width (cm)	target
0	5.1	3.5	...	0.2	0
1	4.9	3.0	...	0.2	0
2	4.7	3.2	...	0.2	0
3	4.6	3.1	...	0.2	0
4	5.0	3.6	...	0.2	0
..	...	...	...	...	...
145	6.7	3.0	...	2.3	2
146	6.3	2.5	...	1.9	2
147	6.5	3.0	...	2.0	2
148	6.2	3.4	...	2.3	2
149	5.9	3.0	...	1.8	2

[150 rows x 5 columns]
Dataset Labels= ['setosa' 'versicolor' 'virginica']
Accuracy: 0.9736842105263158



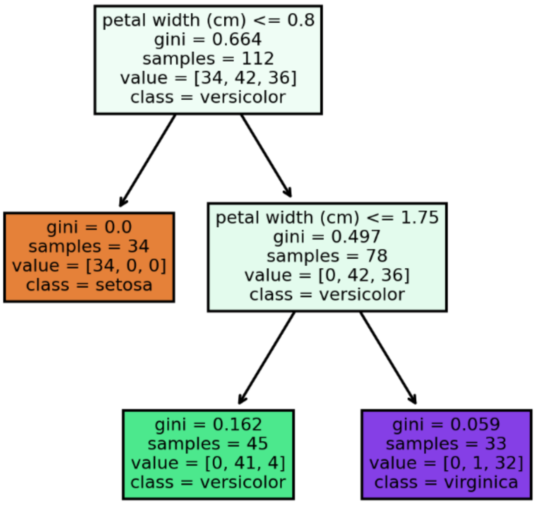
Name :- Nilesh Vijay Patil
Roll No :- 140
Practical No :- 5.1
Practical Name :- Write a program to implement Decision tree using python programming.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_iris
data_b= load_iris()
df = pd.DataFrame(data_b.data,columns = data_b.feature_names)
df['target'] = data_b.target
#df['target']
print(df)
#print(data_b)
print("Dataset Labels=",data_b.target_names)
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df[data_b.feature_names],
df['target'])
#print(x_train)
#print(x_test)
#print(y_train)
#print(y_test)
clf = DecisionTreeClassifier(max_depth = 2,random_state= 1, criterion='gini')
#'gini'
clf = clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
#print(y_test,y_pred)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
fn = ['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
cn=['setosa','versicolor','virginica']

fig, axes = plt.subplots(nrows = 1, ncols = 1,figsize = (4,4),dpi = 300)
tree.plot_tree(clf, feature_names = fn, class_names = cn,filled = True)
fig.savefig('dstimg.png')
```

Output:-

	sepal length (cm)	sepal width (cm)	...	petal width (cm)	target
0	5.1	3.5	...	0.2	0
1	4.9	3.0	...	0.2	0
2	4.7	3.2	...	0.2	0
3	4.6	3.1	...	0.2	0
4	5.0	3.6	...	0.2	0
..	...	...	...	...	...
145	6.7	3.0	...	2.3	2
146	6.3	2.5	...	1.9	2
147	6.5	3.0	...	2.0	2
148	6.2	3.4	...	2.3	2
149	5.9	3.0	...	1.8	2

[150 rows x 5 columns]
Dataset Labels= ['setosa' 'versicolor' 'virginica']
Accuracy: 0.9736842105263158



Name :- Nilesh Vijay Patil
Roll No :- 140
Practical No:- 5.2
Practical Name:- Write a program implement to decision tree to popular attribute selection measure like information gain,gini index etc.for decision tree.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_iris
data_b = load_iris()
df=pd.DataFrame(data_b.data,columns=data_b.feature_names)
df['target'] = data_b.target
#df['target']
print(df)
#print(data_b)
print("Dataset Labels=",data_b.target_names)
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df[data_b.feature_names], df['target'])
print(x_train)
print(x_test)
print(y_train)
print(y_test)
clf = DecisionTreeClassifier(max_depth = 5,random_state =1, criterion='gini')  #'gini'
clf = clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
fn=['sepal length  (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']

fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (4, 4), dpi = 300)
tree.plot_tree(clf, feature_names = fn, class_names = cn, filled = True); fig.savefig('dstimg.png')
```

output:-

```
C:\Users\patil\PycharmProjects\ml\venv\Scripts\python.exe
C:\Users\patil\PycharmProjects\ml\ml4.py

   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0              5.1               3.5  ...               0.2       0
1              4.9               3.0  ...               0.2       0
2              4.7               3.2  ...               0.2       0
3              4.6               3.1  ...               0.2       0
```

```
4          5.0        3.6  ...        0.2     0
..         ...        ...  ...        ...   ...
145        6.7        3.0  ...        2.3     2
146        6.3        2.5  ...        1.9     2
147        6.5        3.0  ...        2.0     2
148        6.2        3.4  ...        2.3     2
149        5.9        3.0  ...        1.8     2

[150 rows x 5 columns]
Dataset Labels= ['setosa' 'versicolor' 'virginica']
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
15         5.7        4.4        1.5        0.4
73         6.1        2.8        4.7        1.2
53         5.5        2.3        4.0        1.3
104        6.5        3.0        5.8        2.2
69         5.6        2.5        3.9        1.1
..         ...        ...        ...        ...
96         5.7        2.9        4.2        1.3
18         5.7        3.8        1.7        0.3
77         6.7        3.0        5.0        1.7
88         5.6        3.0        4.1        1.3
24         4.8        3.4        1.9        0.2

[112 rows x 4 columns]
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
148        6.2        3.4        5.4        2.3
1          4.9        3.0        1.4        0.2
30         4.8        3.1        1.6        0.2
107        7.3        2.9        6.3        1.8
123        6.3        2.7        4.9        1.8
149        5.9        3.0        5.1        1.8
```

132	6.4	2.8	5.6	2.2
9	4.9	3.1	1.5	0.1
112	6.8	3.0	5.5	2.1
117	7.7	3.8	6.7	2.2
75	6.6	3.0	4.4	1.4
102	7.1	3.0	5.9	2.1
89	5.5	2.5	4.0	1.3
127	6.1	3.0	4.9	1.8
37	4.9	3.6	1.4	0.1
16	5.4	3.9	1.3	0.4
29	4.7	3.2	1.6	0.2
83	6.0	2.7	5.1	1.6
133	6.3	2.8	5.1	1.5
135	7.7	3.0	6.1	2.3
40	5.0	3.5	1.3	0.3
59	5.2	2.7	3.9	1.4
43	5.0	3.5	1.6	0.6
106	4.9	2.5	4.5	1.7
131	7.9	3.8	6.4	2.0
23	5.1	3.3	1.7	0.5
26	5.0	3.4	1.6	0.4
74	6.4	2.9	4.3	1.3
70	5.9	3.2	4.8	1.8
109	7.2	3.6	6.1	2.5
90	5.5	2.6	4.4	1.2
99	5.7	2.8	4.1	1.3
139	6.9	3.1	5.4	2.1
20	5.4	3.4	1.7	0.2
62	6.0	2.2	4.0	1.0
147	6.5	3.0	5.2	2.0
116	6.5	3.0	5.5	1.8

118	7.7	2.6	6.9	2.3

15 0

73 1

53 1

104 2

69 1

..

96 1

18 0

77 1

88 1

24 0

Name: target, Length: 112, dtype: int32

148 2

1 0

30 0

107 2

123 2

149 2

132 2

9 0

112 2

117 2

75 1

102 2

89 1

127 2

37 0

16 0

29 0

83 1

133   2

135   2

40   0

59   1

43   0

106   2

131   2

23   0

26   0

74   1

70   1

109   2

90   1

99   1

139   2

20   0

62   1

147   2

116   2

118   2

Name: target, dtype: int32

148   2

1   0

30   0

107   2

123   2

149   2

132   2

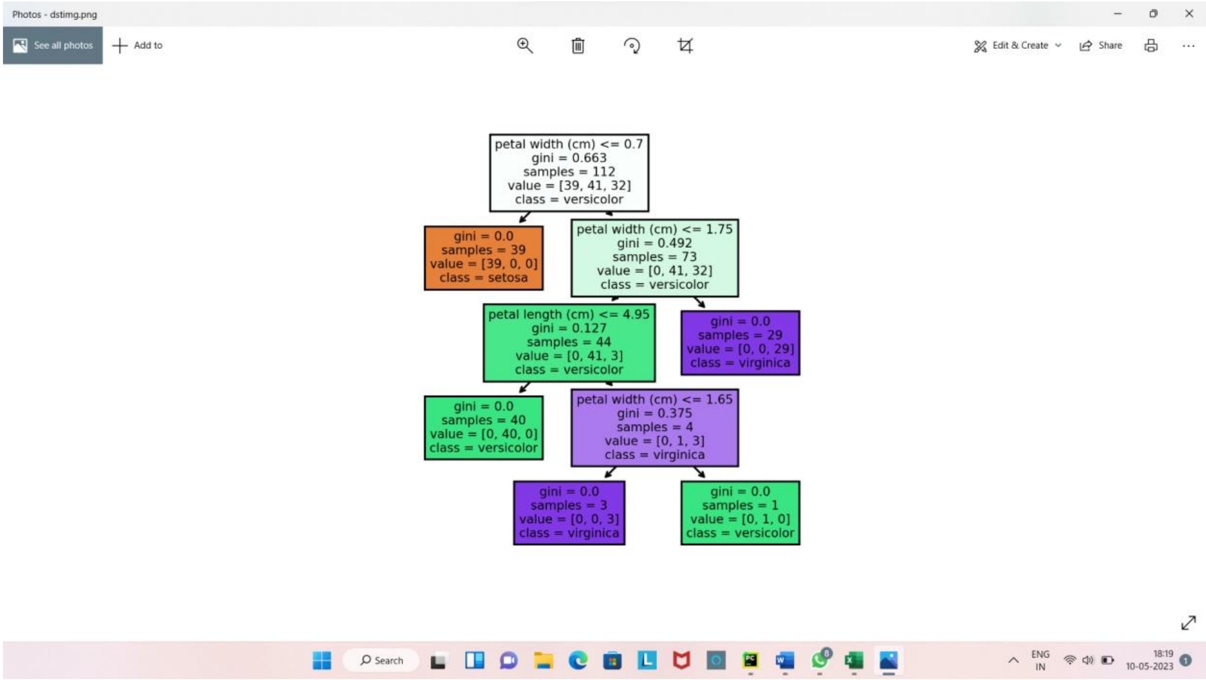
9   0

112   2

117   2

```
75     1
102    2
89     1
127    2
37     0
16     0
29     0
83     1
133    2
135    2
40     0
59     1
43     0
106    2
131    2
23     0
26     0
74     1
70     1
109    2
90     1
99     1
139    2
20     0
62     1
147    2
116    2
118    2
Name: target, dtype: int32 [2 0 0 2 2 2 2 0 2 2 1 2 1 2 0 0 0 2 2 2 0 1 0 1 2 0 0 1 2 2 1 1 2 0 1 2 2
 2]
Accuracy: 0.9210526315789473
```

Process finished with exit code 0



Name: Nilesh Vijay Patil
Roll No: 140
Practical No: 6
Practical Name: Implement simple KNN using Euclidean distance in Python.

Code: KNN using Euclidean distance.

```
from pandas import DataFrame
from sklearn.datasets import load_iris
data_b = load_iris()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

OUTPUT :

```
C:\Users\nilesh\MCA-I_ML\Scripts\python.exe
C:/Users/nilesh/PycharmProjects/MCA-I_ML/KNN.py
Dataset Labels= ['setosa' 'versicolor' 'virginica']
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
54                6.5               2.8                4.6               1.5
```

```
108         6.7         2.5         5.8         1.8
112         6.8         3.0         5.5         2.1
17          5.1         3.5         1.4         0.3
119         6.0         2.2         5.0         1.5
103         6.3         2.9         5.6         1.8
54     1
108    2
112    2
17     0
119    2
103    2
Name: target, dtype: int32
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
14          5.8         4.0         1.2         0.2
98          5.1         2.5         3.0         1.1
75          6.6         3.0         4.4         1.4
16          5.4         3.9         1.3         0.4
131         7.9         3.8         6.4         2.0
Accuracy: 1.0
Confusion Matrix:
[[13  0  0]
 [ 0 16  0]
 [ 0  0  9]]

Process finished with exit code 0
```

Code: For Breast Cancer Data Set

```
from pandas import DataFrame
#from sklearn.datasets import load_iris
from sklearn.datasets import load_breast_cancer

from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
#data_b = load_iris()
data_b = load_breast_cancer()
df = DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
# print(df)
# print(data_b.DESCR)
print("Dataset Labels=", data_b.target_names)

X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train)  # model is trained
y_pred = clf.predict(X_test)
# print(y_test, y_pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
```

OUTPUT:

```
C:\Users\nilesh\MCA-I_ML\Scripts\python.exe
C:/Users/nilesh/PycharmProjects/MCA-I_ML/KNN.py
Dataset Labels= ['malignant' 'benign']
     mean radius  mean texture  ...  worst symmetry  worst fractal dimension
562        15.22         30.62  ...          0.4089                  0.14090
291        14.96         19.10  ...          0.2962                  0.08472
16         14.68         20.13  ...          0.3029                  0.08216
546        10.32         16.35  ...          0.2681                  0.07399
293        11.85         17.46  ...          0.3101                  0.07007
350        11.66         17.07  ...          0.2731                  0.06825

[6 rows x 30 columns]
562    0
291    1
16     0
```

```
546    1
293    1
350    1
Name: target, dtype: int32
     mean radius  mean texture  ...  worst symmetry  worst fractal dimension
421        14.69         13.98  ...          0.2827                  0.09208
47         13.17         18.66  ...          0.3900                  0.11790
292        12.95         16.02  ...          0.3380                  0.09584
186        18.31         18.58  ...          0.3206                  0.06938
414        15.13         29.81  ...          0.3233                  0.06165

[5 rows x 30 columns]
Accuracy: 0.9370629370629371
Confusion Matrix:
[[51  4]
 [ 5 83]]
Number of correct predictions= 134
Number of wrong predictions =  9

Process finished with exit code 0
```

Name :- Nilesh Vijay Patil

Roll No :- 140

Practical No 7 :- Write a program to implement k-Nearest Neighbour algorithm to classify the iris dataset. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

Code:

```
from pandas import  DataFrame
from sklearn.datasets import load_iris
data_b=load_iris()
df=DataFrame(data_b.data,columns=data_b.feature_names)
df['target']=data_b.target
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df[data_b.feature_names],df['target'],random_sta
te=1)
print(x_train)
print(x_test)
clf=KNeighborsClassifier(n_neighbors=6)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("Accurancy:",metrics.accuracy_score(y_test,y_pred))
cm=confusion_matrix(y_test,y_pred)
print("Confussion Matrix:")
print(cm)
```

Output :-

```
Dataset Labels= ['setosa' 'versicolor' 'virginica']
Accurancy: 1.0
Confussion Matrix:
[[13  0  0]
 [ 0 16  0]
 [ 0  0  9]]
Process finished with exit code 0
```

Name : Nilesh Vijay Patil
Roll No.: 140
Practical No.: 8
Practical Name: Write a Program for Confusion Matrix and calculate Precision, Recall, F-Measure

---

```
from sklearn.datasets import load_iris, load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# Load the Irish dataset
iris = load_iris()
X_iris = iris.data
y_iris = iris.target

# Split the Irish dataset into training and testing sets
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris, y_iris, test_size=0.2,
random_state=42)

# Train the KNN classifier on the Irish d3ataset
knn_iris = KNeighborsClassifier()
knn_iris.fit(X_train_iris, y_train_iris)

# Make predictions on the Irish testing set
y_pred_iris = knn_iris.predict(X_test_iris)

# Calculate the confusion matrix for Irish dataset
cm_iris = confusion_matrix(y_test_iris, y_pred_iris)
print("Confusion Matrix (Irish Dataset):")
print(cm_iris)

# Calculate precision, recall, and F-measure for Irish dataset
precision_iris = precision_score(y_test_iris, y_pred_iris, average='macro')
recall_iris = recall_score(y_test_iris, y_pred_iris, average='macro')
f1_iris = f1_score(y_test_iris, y_pred_iris, average='macro')

print("Precision (Irish Dataset):", precision_iris)
print("Recall (Irish Dataset):", recall_iris)
print("F-measure (Irish Dataset):", f1_iris)


# Load the Breast Cancer dataset
cancer = load_breast_cancer()
X_cancer = cancer.data
y_cancer = cancer.target
```

```
# Split the Breast Cancer dataset into training and testing sets
X_train_cancer, X_test_cancer, y_train_cancer, y_test_cancer = train_test_split(X_cancer, y_cancer,
                                                                  test_size=0.2, random_state=42)

# Train the KNN classifier on the Breast Cancer dataset
knn_cancer = KNeighborsClassifier()
knn_cancer.fit(X_train_cancer, y_train_cancer)

# Make predictions on the Breast Cancer testing set
y_pred_cancer = knn_cancer.predict(X_test_cancer)

# Calculate the confusion matrix for Breast Cancer dataset
cm_cancer = confusion_matrix(y_test_cancer, y_pred_cancer)
print("\nConfusion Matrix (Breast Cancer Dataset):")
print(cm_cancer)

# Calculate precision, recall, and F-measure for Breast Cancer dataset
precision_cancer = precision_score(y_test_cancer, y_pred_cancer)
recall_cancer = recall_score(y_test_cancer, y_pred_cancer)
f1_cancer = f1_score(y_test_cancer, y_pred_cancer)

print("Precision (Breast Cancer Dataset):", precision_cancer)
print("Recall (Irish Dataset):", recall_cancer)
print("F-measure (Irish Dataset):", f1_cancer)
```

OUTPUT:

```
Confusion Matrix (Irish Dataset):
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Precision (Irish Dataset): 1.0
Recall (Irish Dataset): 1.0
F-measure (Irish Dataset): 1.0

Confusion Matrix (Breast Cancer Dataset):
[[38  5]
 [ 0 71]]
Precision (Breast Cancer Dataset): 0.9342105263157895
Recall (Irish Dataset): 1.0
F-measure (Irish Dataset): 0.9659863945578232
```

Name : Nilesh Vijay Patil
Roll No.: 140
Practical No.: 9
Practical NameWrite a program for linear regression and find parameters like Sum of Squared Errors (SSE), Total Sum of Squares (SST), R2, Adjusted R2, etc.

```
 import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Input data
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.array([3, 4, 5, 6])

model = LinearRegression()   # Create a linear regression model

model.fit(X, y)   # Fit the model to the data

y_pred = model.predict(X)  # Predict the output

sse = np.sum((y_pred - y) ** 2)   # Calculate SSE (Sum of Squared Errors)

sst = np.sum((y - np.mean(y)) ** 2)  # Calculate SST (Total Sum of Squares)

r2 = r2_score(y, y_pred)  # Calculate R2 score

# Calculate adjusted R2
n = X.shape[0]  # Number of samples
p = X.shape[1]  # Number of predictors
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)

# Print the results
print("Sum of Squared Errors(SSE):- ", sse)
print("Total Sum of Squares(SST):- ", sst)
print("R Square(R2):- ", r2)
print("Adjusted Square(R2):- ", adjusted_r2 )
```

OUTPUT:

```
Sum of Squared Errors(SSE):-  0.0
Total Sum of Squares(SST):-  5.0
R Square(R2):-  1.0
Adjusted Square(R2):-  1.0
```

Name :- Nilesh Vijay Patil
Roll No. :- 140
Practical No :- 10
Practical Name :- Write the program to implement the naive Bayesian Classifier for a sample training dataset stored as a .CSV file. Compute the accuracy of the classifier considering a few test dataset.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
iris = datasets.load_iris()   #load dataset
x = iris.data   #input
y = iris.target #traget
print("Features :", iris['feature_names'])

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
NB = GaussianNB()
NB.fit(x_train, y_train)
y_pred = NB.predict(x_test)
cm = confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)
```

OUTPUT:

```
Features : ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Confusion Matrix
[ [13  0  0]
  [ 0 16  0]
  [ 0  0   9] ]
```

Name :- Nilesh Vijay Patil
Roll No :- 140
Program No. :- 11.2
Practical Name :- Write a Program for Fuzzy c-means clustering in python.

```
import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl


# Generate some example data

np.random.seed(0)

data = np.random.rand(100, 2)


# Define the number of clusters

n_clusters = 3


# Apply fuzzy c-means clustering

cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(

data.T, n_clusters, 2, error=0.005, maxiter=1000, init=None)


# Predict cluster membership for each data point

cluster_membership = np.argmax(u, axis=0)


# Print the cluster centers

print('Cluster Centers:', cntr)


# Print the cluster membership for each data point

print('Cluster Membership:', cluster_membership)
```

Output :-

Cluster Centers: [[0.22645397 0.71840176]

[0.52083891 0.18668653]

[0.76252289 0.60239021]]

Cluster Membership: [2 2 0 0 2 2 2 1 0 2 2 0 0 0 1 0

0 0 2 2 1 1 2 1 1 2 1 1 1 1 1 1 0 1 1 2 2

1 1 1 1 0 1 1 2 0 0 1 1 1 1 2 0 2 0 0 1 2 2 2 2 2 0

0 1 2 1 2 2 2 2 0 2 0

2 0 0 0 2 1 2 2 2 0 1 1 1 1 0 1 0 1 2 2 1 1 0 2 1 0]

Name :- Nilesh Vijay Patil

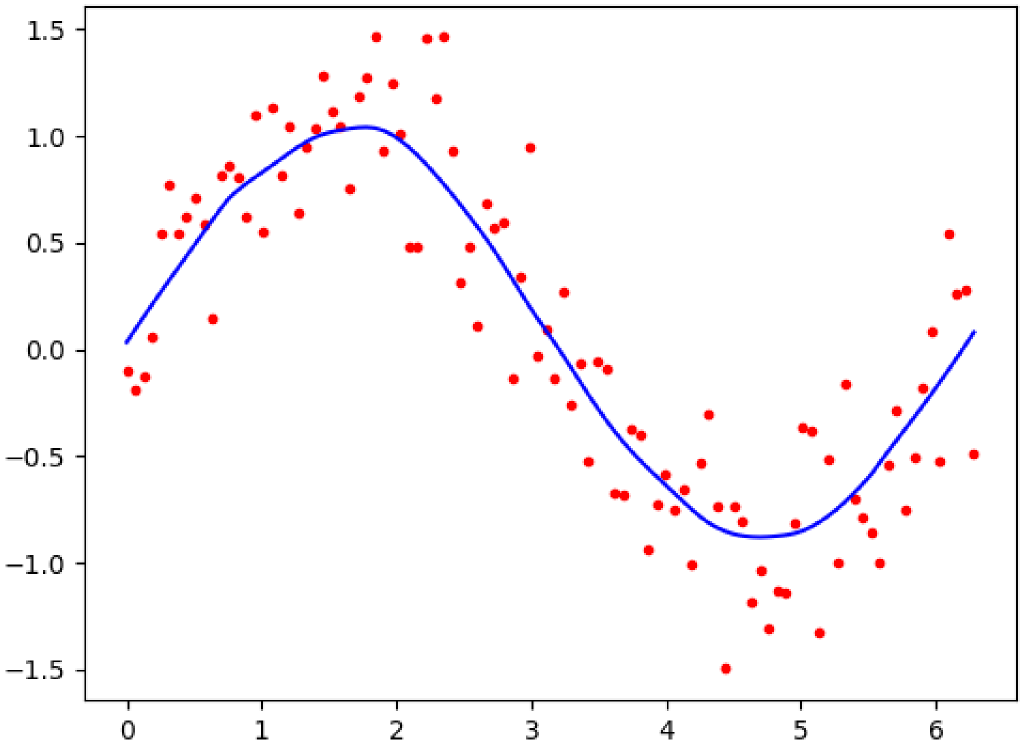
Roll No :- 140

Practical No. :- 12

Practical Name :- Implement the non-parametric locally weighted regression algorithm in order to fit data points. select the appropriate data set for your experiment and draw graphs.

```
from math import ceil
import numpy as np
from scipy import linalg


def lowess(x, y, f, iterations):
    n = len(x)
    r = int(ceil(f * n))
    h = [np.sort(np.abs(x - x[i]))[r] for i in range(n)]
    w = np.clip(np.abs((x[:, None] - x[None, :]) / h), 0.0, 1.0)
    w = (1 - w ** 3) ** 3
    yest = np.zeros(n)
    delta = np.ones(n)
    for iteration in range(iterations):
        for i in range(n):
            weights = delta * w[:, i]
            b = np.array([np.sum(weights * y), np.sum(weights * y * x)])
            A = np.array([[np.sum(weights), np.sum(weights * x)], [np.sum(weights * x), np.sum(weights
* x * x)]])
            beta = linalg.solve(A, b)
            yest[i] = beta[0] + beta[1] * x[i]

        residuals = y - yest
        s = np.median(np.abs(residuals))
        delta = np.clip(residuals / (6.0 * s), -1, 1)
        delta = (1 - delta ** 2) ** 2

    return yest


import math

n = 100
x = np.linspace(0, 2 * math.pi, n)
y = np.sin(x) + 0.3 * np.random.randn(n)
f = 0.25
iterations = 3
yest = lowess(x, y, f, iterations)

import matplotlib.pyplot as plt
plt.plot(x, y, "r.")
plt.plot(x, yest, "b-")
plt.show()
```

OUTPUT:



Name :- Nilesh Vijay Patil
Roll No. :- 140
Practical No. :- 13.1
Practical Name :- Construction Of simple Neural Network using Python

---

Code:-

```
import numpy as np
from scipy.special import expit as activation_function
from scipy.stats import truncnorm

# define the network
# generate numbers within a truncated (bounded)
# normal Distribution


def truncated_normal(mean=0, sd=1, low=0, upp=10):
    return truncnorm((low - mean) / sd, (upp - mean) / sd, loc=mean, scale=sd)

# creat the Network class and define the arguments:
# set the no. of neurons/nodes for each layer
# and initialize the weight matrices


class Nnetwork:
    def __init__(self, no_of_in_nodes, no_of_out_nodes, no_of_hidden_nodes, learning_rate):
        self.no_of_in_nodes = no_of_in_nodes
        self.no_of_out_nodes = no_of_out_nodes
        self.no_of_hidden_nodes = no_of_hidden_nodes
        self.learning_rate = learning_rate
        self.create_weight_matrices()

    def create_weight_matrices(self):
        """A method to initialize the weight matrices of the neural network"""
        rad = 1 / np.sqrt(self.no_of_in_nodes)  # rad = 0.2707
        x = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
        self.weight_in_hidden = x.rvs((self.no_of_hidden_nodes, self.no_of_in_nodes))
        print("weights_in_hidden = ", self.weight_in_hidden)
        rad = 1/np.sqrt(self.no_of_hidden_nodes)
        x = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
        self.weight_in_hidden_out = x.rvs((self.no_of_out_nodes, self.no_of_hidden_nodes))
        print("weights_in_hidden_out = ", self.weight_in_hidden_out)

    def train(self, input_vector, target_vector):
        pass

    def run(self, input_vector):
        input_vector = np.array(input_vector, ndmin=2).T
        print("Input = ", input_vector)

        input_hidden = activation_function(self.weight_in_hidden @ input_vector)
```

```
        print("Hidden = ", input_hidden)

        output_vector = activation_function(self.weight_in_hidden_out @ input_hidden)
        print("Output = ", output_vector)
        return output_vector

simple_network = Nnetwork(no_of_in_nodes=2, no_of_out_nodes=2, no_of_hidden_nodes=4,
learning_rate=0.6)

#run simple network for arrays, lists and tuples with shape (2):

y = simple_network.run([2,3])
print("Y = ", y)
```

OUTPUT":

```
weights_in_hidden =  [[-0.68798443  0.29428266]
 [ 0.57363879 -0.64646032]
 [-0.38809421  0.07104818]
 [-0.23288421  0.26427463]]
weights_in_hidden_out =  [[ 0.12718945 -0.15067287 -0.36574728  0.3725497 ]
 [-0.09102931 -0.22077172  0.40025881 -0.32163589]]
Input =  [[2]
 [3]]
Hidden =  [[0.37915865]
 [0.31171721]
 [0.36284346]
 [0.58104275]]
Output =  [[0.52124119]
 [0.46381691]]
Y =  [[0.52124119]
 [0.46381691]]
```

Name:- Nilesh Vijay Patil
Roll No:- 140
Practical No:- 13.2
Practical Name: Classification Of Iris Dataset By Applying Artificial Neural Network With Back-Propogation Algorithm

---

```
# classification of iris data set by aplying artificial neural network using Back-propogation algorithm
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# load dataset
data = load_iris()

# Get features and target
x = data.data
y = data.target
print("Y=", y)

y = pd.get_dummies(y).values
print(y[:3])

# split data into train and test data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=20, random_state=4)

# initialize variable
learning_rate = 0.1
iteration = 6000
N = y_train.size

# number of input features
input_size = 4

# number of hidden layers neurons
hidden_size = 2

# mo. of neurons at output layers
output_size = 3
results = pd.DataFrame(columns=["mse", "accuracy"])

# initialize weights
np.random.seed(10)
# initialiizing weight for the hidden layers
W1 = np.random.normal(scale=0.5, size=(input_size, hidden_size))
print("weight 1", W1)

# initializing weight for the output layers
```

```
W2 = np.random.normal(scale=0.5, size=(hidden_size, output_size))
print("weight 2", W2)


def sigmoid(x):
    return 1/(1 + np.exp(-x))


def mean_squared_error(y_pred, y_true):
    return (((y_pred - y_true) ** 2).sum()) / (2 * y_pred.size)


def accuracy(y_pred, y_true):
    acc = y_pred.argmax(axis=1) == y_true.argmax(axis=1)
    return acc.mean()


for itr in range(iteration):

    # feedforward propagation
    # on hidden layer
    Z1 = np.dot(x_train, W1)
    A1 = sigmoid(Z1)

# on output layer
    Z2 = np.dot(A1, W2)
    A2 = sigmoid(Z2)

# calculating error
    mse = mean_squared_error(A2, y_train)
    acc = accuracy(A2, y_train)
    results = results._append({"mse": mse, "accuracy": acc}, ignore_index=True)

# backpropagation
    E1 = A2 - y_train
    dw1 = E1 * A2 * (1 - A2)

    E2 = np.dot(dw1, W2.T)
    dw2 = E2 * A1 * (1 - A1)

# weight updates
    W2_update = np.dot(A1.T, dw1) / N
    W1_update = np.dot(x_train.T, dw2) / N

    W2 = W2 - learning_rate * W2_update
    W1 = W1 - learning_rate * W1_update

results.mse.plot(title="Mean squared Error")

results.accuracy.plot(title="Accuracy")
```

```
# feedforward
Z1 = np.dot(x_test, W1)
A1 = sigmoid(Z1)

Z2 = np.dot(A1, W2)
A2 = sigmoid(Z2)

acc = accuracy(A2, y_test)
print("Accuracy: {}".format(acc))
```

OUTPUT:

```
C:\Users\nilesh\MCA-I_ML\Scripts\python.exe C:/Users/nilesh/PycharmProjects/MCA-I_ML/nural_network_Backpropa_algo.py
Y= [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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
[[ True False False]
 [ True False False]
 [ True False False]]
weight 1 [[ 0.66579325  0.35763949]
 [-0.77270015 -0.00419192]
 [ 0.31066799 -0.36004278]
 [ 0.13275579  0.05427426]]
weight 2 [[ 0.00214572 -0.08730011  0.21651309]
 [ 0.60151869 -0.48253284  0.51413704]]
```