



SCHOOL OF COMPUTER SCIENCE

MACHINE LEARNING

BACHELOR OF COMPUTER SCIENCE ENGINEERING WITH
SPECIALIZATION IN ARTIFICIAL INTELLIGENCE AND MACHINE
LEARNING

BATCH: B3 (Hons.)

SEMESTER: IV

SUBMITTED TO: PROF. GOPAL S PHARTIYAL

SUBMITTED BY: HARSHIT RAHEJA

SAP ID: 500086226

ROLL NO: R2142201556

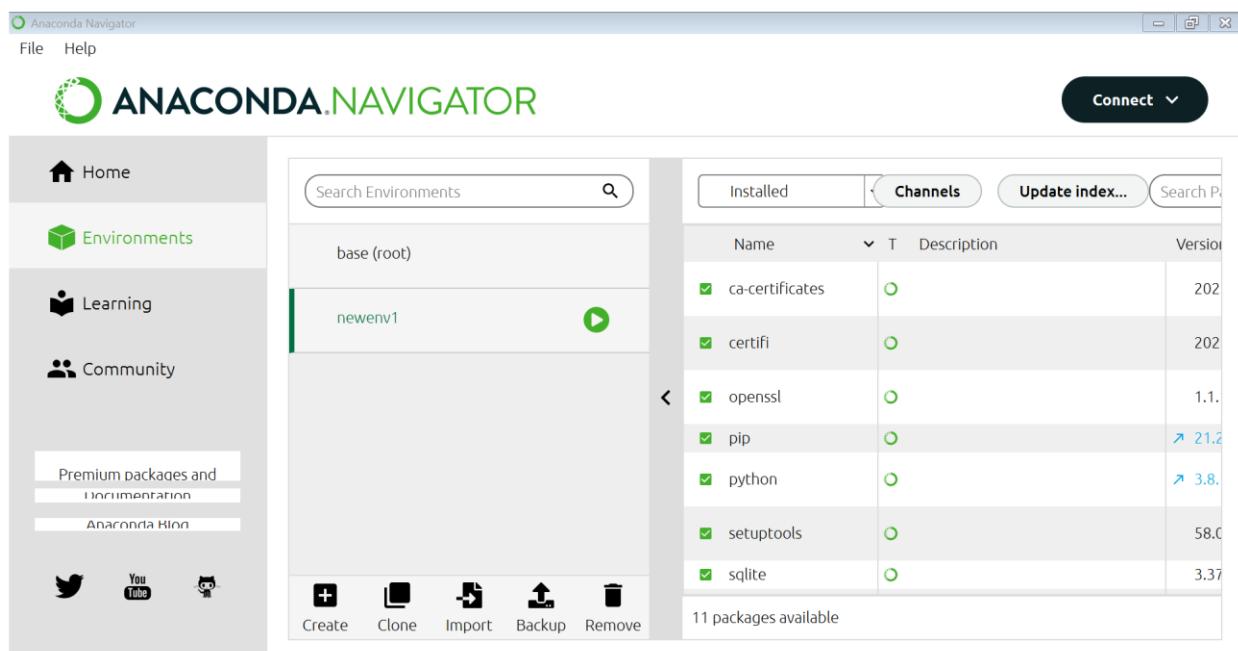
Index

| S.no | Page no. | Experiment | Remarks |
|------|----------|---|---------|
| 1 | 3 | Introduction to Scikit learn library | |
| 2 | 3-7 | Linear regression | |
| 3 | 8-9 | Linear regression with x and y as vector | |
| 4 | 10-17 | Coefficient of Determination, Split Train, and Test data, LR and R model | |
| 5 | 17-19 | Classification | |
| 6 | 20-22 | Multiclass Classification | |
| 7 | 22-26 | Decision Tree | |
| 8 | 26-37 | Random forest and Adaboost | |
| 9 | 38-44 | Neural Network | |
| 10 | 45-47 | Support Vector Machine | |

Experiment 1: Introduction to Scikit learn library

Installation Steps

1. Install Anaconda
2. Open the Anaconda Navigator
3. Go to environment and create a new environment.
4. Go to environments in the left panel and check the installed libraries.
5. Install numpy
6. Install scipy
7. Install sklearn
8. Installation is completed.



Experiment 2: Linear regression

Aim: Linear regression with input as vector

Code:

```
from sklearn.datasets import make_regression
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np

x,t= make_regression(100,1, shuffle= True, bias =0.0, noise=10, random_state=4 )

print("The maximum value of x is :",max(x))

print("The maximum value of t is :",max(t))

print("The minimum value of x is :",min(x))

print("The minimum value of t is :",min(t))

mean_x=np.mean(x)

print("The mean of X is :",mean_x)

mean_t=np.mean(t)

print("The meand of t is :",mean_t)

Sd_x=np.std(x)

print("The standard deviation of x is :",Sd_x)

Sd_t=np.std(t)

print("The standard deviation of t is:",Sd_t)

plt.scatter(x, t)

plt.show()

I=x-np.mean(x)

J=t-np.mean(t)
```

```
b_1 = np.sum(I*J)/np.sum(I*I)
```

```
b_0 = mean_t - b_1*mean_x
```

```
print(b_0,"\\t",b_1)
```

Output:

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The main area displays the code in a script named 'ML_exp2.py'.

```
1  #%%%
2  from sklearn.datasets import make_regression
3  import matplotlib.pyplot as plt
4  import numpy as np
5  x,t= make_regression(100,1, shuffle= True, bias =0.0, noise=10, random_state=
6  print("The maximum value of x is : ",max(x))
7  print("The maximum value of t is : ",max(t))
8  print("The minimum value of x is : ",min(x))
9  print("The minimum value of t is : ",min(t))
10 mean_x=np.mean(x)
11 print("The mean of X is : ",mean_x)
12 mean_t=np.mean(t)
13 print("The meand of t is : ",mean_t)
14 Sd_x=np.std(x)
15 print("The standard deviation of x is : ",Sd_x)
16 Sd_t=np.std(t)
17 print("The standard deviation of t is : ",Sd_t)
18 plt.scatter(x, t)
19 plt.show()
20 I=x-np.mean(x)
21 J=t-np.mean(t)
22 b_1 = np.sum(I*J)/np.sum(I*I)
23 b_0 = mean_t - b_1*mean_x
24 print(b_0,"\\t",b_1)
25
```

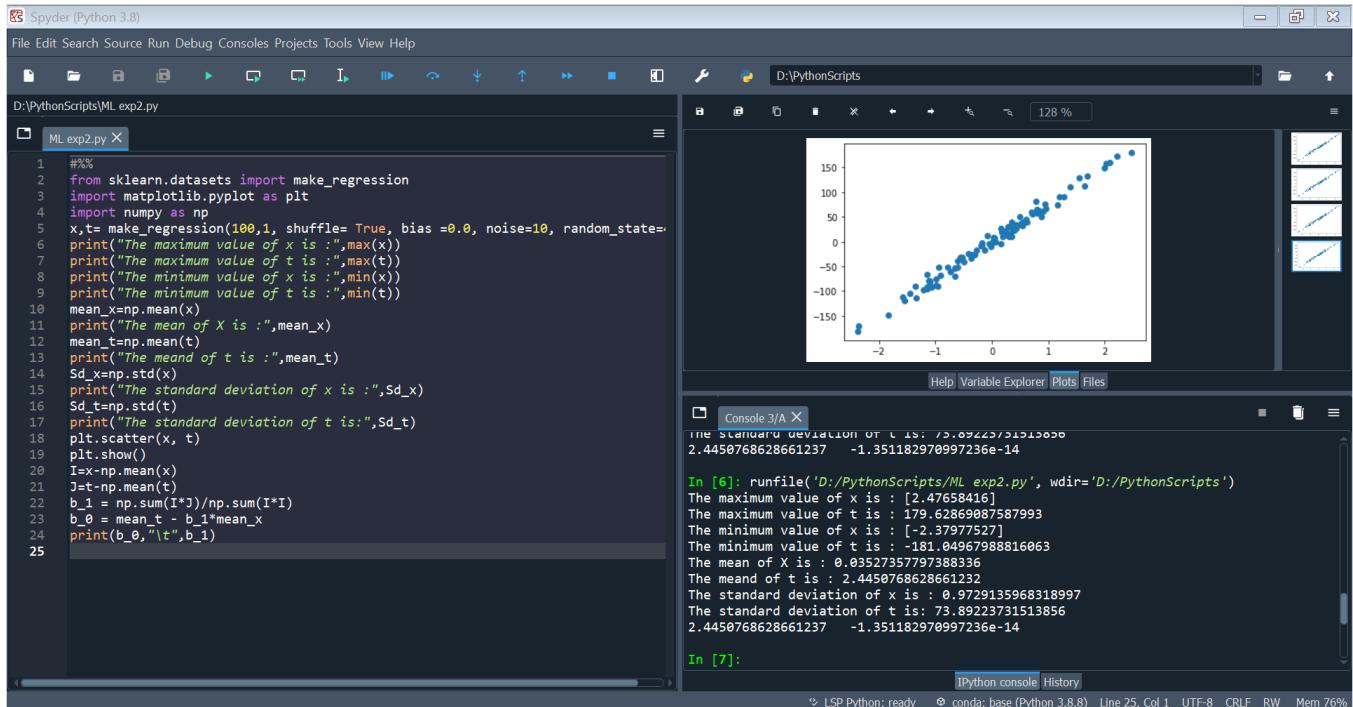
The Variable Explorer panel on the right shows the following data:

| Name | Type | Size | Value |
|--------|------------------|----------|--|
| b_0 | float64 | 1 | 2.4450768628661237 |
| b_1 | float64 | 1 | -1.351182970997236e-14 |
| I | Array of float64 | (100, 1) | [[-1.38273387], [0.31706982], [-116.74148308], ..., [157.34291264], [-116.74148308], ..., [157.34291264]] |
| J | Array of float64 | (100,) | 36.40006836 65.38832845 ... |
| J | Array of float64 | (100,) | 36.40006836 65.38832845 ... |
| mean_t | float64 | 1 | 2.4450768628661232 |
| mean_x | float64 | 1 | 0.03527357797388336 |

The IPython console at the bottom shows the execution results:

```
In [6]: runfile('D:/PythonScripts/ML_exp2.py', wdir='D:/PythonScripts')
The maximum value of x is : 2.47658416
The maximum value of t is : 179.62869087587993
The minimum value of x is : [-2.37977527]
The minimum value of t is : -181.04967988816063
The mean of X is : 0.03527357797388336
The meand of t is : 2.4450768628661232
The standard deviation of x is : 0.9729135968318997
The standard deviation of t is : 73.89223731513856
2.4450768628661237 -1.351182970997236e-14
```

At the bottom, status information includes LSP Python: ready, conda: base (Python 3.8.8), Line 25, Col 1, UTF-8, CRLF, RW, Mem 76%.



Experiment 2.1: Linear regression with input as vector

Aim: Linear regression with input as vector

Code:

```
from sklearn.datasets import make_regression
```

```
import numpy as np
```

```
x,t= make_regression(n_samples=100,n_features=5,shuffle=True, bias =0.0, noise=10,
```

```
random_state=1)
```

```
o=np.ones(100)
```

```
x=np.insert(x,0,o,axis=1)
```

```

print("Dimensions of x :",np.shape(x))

X=np.transpose(x)

y=(np.linalg.inv(np.dot(X,x)))

w=np.dot(y,np.dot(X,t))

print("Values of w's are :",w)

#%%
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(x,t)

print(regressor.intercept_)

```

Output:

The screenshot shows the Spyder Python 3.8 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The main window has two tabs: 'ML_exp2.py' and 'ML_Exp3.py*'. The code in 'ML_Exp3.py*' is as follows:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Jan 29 11:27:56 2022
4
5 @author: Lenovo
6 """
7 #%%
8 from sklearn.datasets import make_regression
9 import numpy as np
10 x,t= make_regression(n_samples=100,n_features=5,shuffle=True, bias =0.0, noise=1.0)
11 o=np.ones(100)
12 x=np.insert(x,0,o, axis=1)
13 print("Dimensions of x : ",np.shape(x))
14 X=np.transpose(x)
15 y=(np.linalg.inv(np.dot(X,x)))
16 w=np.dot(y,np.dot(X,t))
17 print("Values of w's are : ",w)
18
19 #%%
20 from sklearn.linear_model import LinearRegression
21 regressor = LinearRegression()
22 regressor.fit(x,t)
23 print(regressor.intercept_)

```

The Variable Explorer pane on the right shows the following variables:

| Name | Type | Size | Value |
|-----------|-------------------------------------|----------|--|
| o | Array of float64 | (100,) | [1. 1. 1. ... 1. 1. 1.] |
| regressor | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| t | Array of float64 | (100,) | [32.23471426 -56.99996383 109.50432...] |
| w | Array of float64 | (6,) | [0.97198982 8.23155766 51.62669014 35.68489779 26.46372731 22.493009 ...] |
| x | Array of float64 | (100, 6) | [[1. 1. 1. 1. 1. 1.], [1. 1. 1. 1. 1. 1.], ..., [1. 1. 1. 1. 1. 1.]] |
| X | Array of float64 | (6, 100) | [[1. 1. 1. 1. 1. 1.], [1. 1. 1. 1. 1. 1.], ..., [1. 1. 1. 1. 1. 1.]] |
| y | Array of float64 | (6, 6) | [[0.01062854 -0.00016699 -0.00039181 0.0013594 -0.00224718 0.00099 ...] |

The IPython console at the bottom shows the execution results:

```

In [37]: runfile('E:/upes/Semester 4/Machine Learning/Lab/ML_Exp3.py', wdir='E:/upes/Semester 4/Machine Learning/Lab')
Dimensions of x : (100, 6)
Values of w's are : [ 0.97198982 8.23155766 51.62669014 35.68489779 26.46372731 22.49300908]

In [38]:

```

Experiment 3: Linear regression with x and y as vector

Aim: Linear regression with x and y as vector

Code:

```
#% %
```

```
from sklearn.datasets import make_regression
```

```
import numpy as np
```

```
x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = 10, random_state = 1, n_targets = 5)
```

```
x0 = np.ones((100,1))
```

```
newX = np.concatenate((x0, x), axis =1)
```

```
Xt = np.transpose(newX)
```

```
Winv = np.dot(Xt,newX)
```

```
W1 = np.linalg.inv(Winv)
```

```
W2 = np.dot(W1, Xt)
```

```
W = np.dot(W2,t)
```

```
print("The values of W are: ",W)
```

```
#% %
```

```
y=np.dot(np.transpose(W),Xt)
```

```
#%%

from sklearn.linear_model import LinearRegression

model = LinearRegression()

w = model.fit(x,t)

print(model.intercept_)

print(np.transpose(model.coef_))
```

Output:

The screenshot shows the Spyder Python 3.8 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The left sidebar shows three open files: ML_exp2.py, ML_Exp3.py, and ML_exp2.2.py*. The main area displays the code for ML_exp2.2.py*, which imports necessary libraries and performs linear regression calculations. The Variable Explorer on the right shows the state of variables: model (LinearRegression object), newX (Array of float64), t (Array of float64), w (Array of float64), W (linear_model._base.LinearRegression object), W1 (Array of float64), and W2 (Array of float64). The IPython console at the bottom shows the execution of the code, including the output of the print statements.

```

# -*- coding: utf-8 -*-
# Created on Sat Feb  5 11:16:29 2022
# @author: Lenovo
#%%
from sklearn.datasets import make_regression
import numpy as np
x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = 10, random_state = 1, n_targets = 5)
newX = np.concatenate((x,1), axis=1)
Xt = np.transpose(newX)
Winv = np.dot(Xt,newX)
W1 = np.linalg.inv(Winv)
W = np.dot(W1,Xt)
W = np.transpose(W)
print("The values of W are: ",W)
#%%
y=np.dot(np.transpose(W),Xt)
#%%
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model = model.fit(x,t)
print(model.intercept_)
print(np.transpose(model.coef_))

In [44]: runcell(t, 'D:/PythonScripts/ML_exp2.2.py')
The values of W are: [[ 1.22418408  0.01029957  0.17032319  0.96591982 -0.68129122]
[75.91893571 10.48074354 49.38871509 21.05350606 16.65191228]
[19.91349993 6.6818684 49.38861427 26.3944197 34.71119638]
[55.9978671 4.12834177 79.26516364 10.793226 53.68491929]
[10.78176086 79.0753797 9.9533553 98.9907775 17.21825076]
[71.30963429 17.87590479 86.89179137 1.46268676 85.56859189]]
```

Experiment 4:

Part 1: Coefficient of Determination

Aim: Coefficient of Determination

Code:

```
from sklearn.datasets import make_regression  
  
from sklearn.linear_model import LinearRegression  
  
from sklearn.metrics import r2_score  
  
n=[0,10,20,40,80]  
  
q=[]  
  
for i in n:  
  
    x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = i, random_state=1)  
  
    model = LinearRegression()  
  
    w = model.fit(x,t)  
  
    y=model.predict(x)  
  
    score = r2_score(t,y)  
  
    q.append(score)
```

```

print("R^2 score for perfect model is when noise is",i,: ",score)

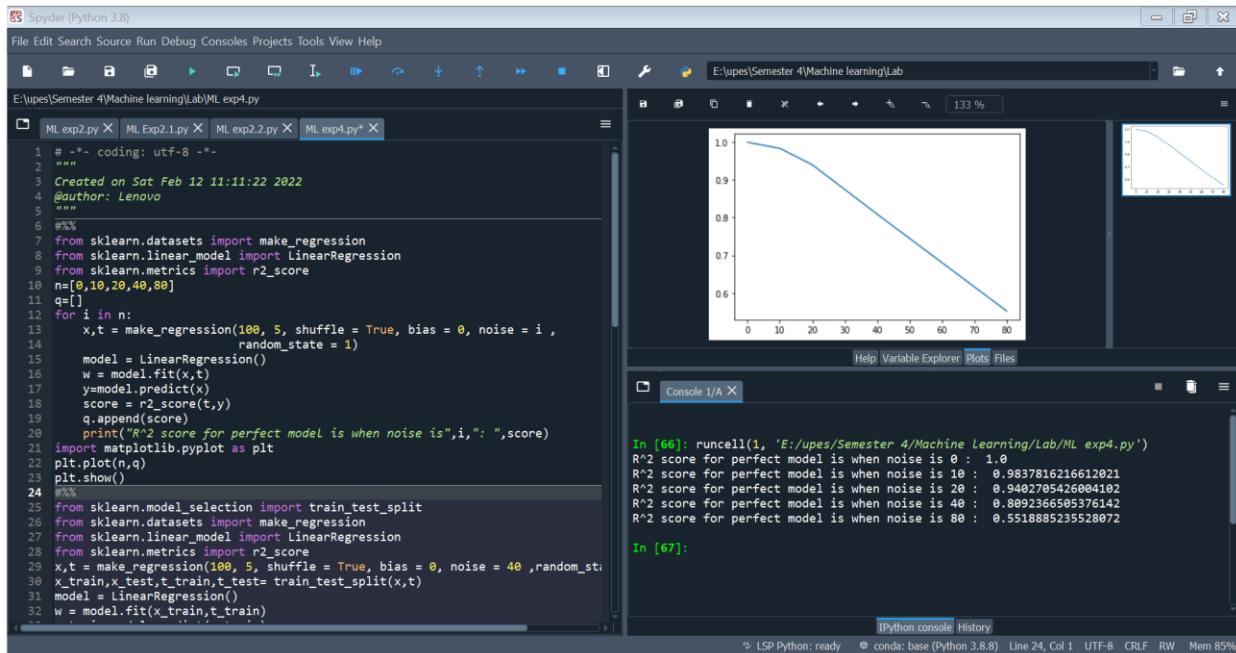
import matplotlib.pyplot as plt

plt.plot(n,q)

plt.show()

```

Output:



Part 2: Split Train and Test data

Part 2.1: Coefficient of determination of Train and Test data

Aim: Coefficient of determination of Train and Test data

Code:

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.datasets import make_regression

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = 40 ,random_state = 1)

x_train,x_test,t_train,t_test= train_test_split(x,t)

model = LinearRegression()

w = model.fit(x_train,t_train)

y_train=model.predict(x_train)

y_test=model.predict(x_test)

score = r2_score(t_train,y_train)

score2=r2_score(t_test,y_test)

print("R2 for train data: ",score)

print("R2 for test data:",score2)
```

Output:

The screenshot shows the Spyder IDE interface with the following details:

- Code Editor:** Displays the Python script `ML exp4.py` containing code for linear regression with variable noise.
- Variable Explorer:** Shows the state of variables in memory, including:
 - `i`: int, Value: 80
 - `model`: linear_model._base.LinearRegression, Value: LinearRegression object of sklearn.linear_model._base module
 - `model_LR`: linear_model._base.LinearRegression, Value: LinearRegression object of sklearn.linear_model._base module
 - `model_R`: linear_model._ridge.Ridge, Value: Ridge object of sklearn.linear_model._ridge module
 - `n`: list, Value: [0, 10, 20, 40, 80]
 - `q`: list, Value: [1.0, 0.985356930020785, 0.9216496...]
 - `q1`: list, Value: [1.0, 0.9718279519956938, 0.9649248...]
- Console:** Displays the execution of the script and its output:
 - In [66]: `runcell(1, 'E:/upes/Semester 4/Machine Learning/Lab/ML exp4.py')`
 - R² score for perfect model is when noise is 0 : 1.0
 - R² score for perfect model is when noise is 10 : 0.9837816216612021
 - R² score for perfect model is when noise is 20 : 0.9402705426004102
 - R² score for perfect model is when noise is 40 : 0.8092366505376142
 - R² score for perfect model is when noise is 80 : 0.5518885235528072- In [67]: `runcell(2, 'E:/upes/Semester 4/Machine Learning/Lab/ML exp4.py')`
- R² for train data: 0.8148664794757692
- R² for test data: 0.7513495533852201
- In [68]:

Part 2.2: Coefficient of determination of Train and Test data with variable noise

Aim: Coefficient of determination of Train and Test data with variable noise

Code:

```
n=[0,10,20,40,80]
```

```
q=[]
```

```
q1=[]
```

```
for i in n:
```

```
x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = i ,random_state = 1)
```

```
x_train,x_test,t_train,t_test= train_test_split(x,t)
```

```
model = LinearRegression()
```

```
w = model.fit(x_train,t_train)
```

```
y_train=model.predict(x_train)

y_test=model.predict(x_test)

score = r2_score(t_train,y_train)

score2 = r2_score(t_test,y_test)

q.append(score)

q1.append(score2)

#print("R^2 score for train data when noise is ",i,": ",score2)

import matplotlib.pyplot as plt

plt.plot(n,q,"b")

plt.plot(n,q1,"r")

plt.show()
```

Output:

The screenshot shows the Spyder Python 3.8 IDE interface. On the left, there are four tabs: ML_exp2.py, ML_Exp2.1.py, ML_exp2.2.py, and ML_exp4.py*. The ML_exp4.py* tab is active, displaying Python code for generating multiple linear regression models with different noise levels and calculating their R-squared scores. The code uses `make_regression` from `sklearn.datasets` and `LinearRegression` from `sklearn.linear_model`. It also imports `matplotlib.pyplot` for plotting. The right side of the interface shows two plots: a main plot with a red line and a blue line, and an inset plot showing a similar trend. Below the plots is a 'Plots' tab. At the bottom, the 'Console I/A' tab shows the execution of code cells, with outputs for R-squared values for train and test data. The status bar at the bottom indicates the Python version (3.8.8), conda base environment, and memory usage (82%).

```
34 y_test= model.predict(x_test)
35 score = r2_score(y_train,y_train)
36 score2=r2_score(t_train,y_train)
37 print("R2 for train data: ",score)
38 print("R2 for test data: ",score2)
39 #%%%
40 n=[0,10,20,40,80]
41 q=[]
42 q1=[]
43 for i in n:
44     x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = i ,random_state=1)
45     x_train,x_test,t_train,t_test= train_test_split(x,t)
46     model = LinearRegression()
47     model.fit(x_train,y_train)
48     y_trainmodel.predict(x_train)
49     y_testmodel.predict(x_test)
50     score = r2_score(t_train,y_train)
51     score2 = r2_score(t_test,y_test)
52     q.append(score)
53     q1.append(score2)
54     #print("R^2 score for train data when noise is ",i," : ",score2)
55     import matplotlib.pyplot as plt
56     plt.plot(n,q,'b')
57     plt.plot(n,q1,'r')
58     plt.show()
59 #%%%
60 from sklearn.model_selection import train_test_split
61 from sklearn.datasets import make_regression
62 from sklearn.linear_model import LinearRegression
63 from sklearn.linear_model import Ridge
64 from sklearn.metrics import r2_score
65 x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = 80 ,random_state=1)
```

```
In [67]: runcell(3, 'E:/upes/Semester 4/Machine Learning/Lab/ML_exp4.py')
In [68]: runcell(2, 'E:/upes/Semester 4/Machine Learning/Lab/ML_exp4.py')
R2 for train data: 0.8148664794757692
R2 for test data: 0.7513495533852201
In [69]: runcell(3, 'E:/upes/Semester 4/Machine Learning/Lab/ML_exp4.py')
In [70]: runcell(3, 'E:/upes/Semester 4/Machine Learning/Lab/ML_exp4.py')
In [71]: runcell(3, 'E:/upes/Semester 4/Machine Learning/Lab/ML_exp4.py')
In [72]:
```

Part 3: LR and R model

Aim: LR and R model

Code:

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.datasets import make_regression
```

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.linear_model import Ridge
```

```
from sklearn.metrics import r2_score
```

```
x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = 80 ,random_state = 1)
```

```
x_train,x_test,t_train,t_test= train_test_split(x,t)
```

```
model_LR= LinearRegression()
```

```
model_R=Ridge()
```

```
w = model_LR.fit(x_train,t_train)

w1=model_R.fit(x_train,t_train)

y_train=model_LR.predict(x_train)

y_test=model_LR.predict(x_test)

y1_train=model_R.predict(x_train)

y1_test=model_R.predict(x_test)

score = r2_score(t_train,y_train)

score2=r2_score(t_test,y_test)

score3 = r2_score(t_train,y1_train)

score4=r2_score(t_test,y1_test)

print("R2 for train data in LR: ",score)

print("R2 for test data in LR: ",score2)

print("R2 for train data in R: ",score3)

print("R2 for test data in R: ",score4)
```

Output:

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named 'ML exp4.py' with the following content:

```
53 q1.append(score2)
54 #print("R^2 score for train data when noise is ",i,: "score2")
55 import matplotlib.pyplot as plt
56 plt.plot(n,q,"b")
57 plt.plot(n,q1,"r")
58 plt.show()
59 #%%
60 from sklearn.model_selection import train_test_split
61 from sklearn.datasets import make_regression
62 from sklearn.linear_model import LinearRegression
63 from sklearn.linear_model import Ridge
64 from sklearn.metrics import r2_score
65 x,t = make_regression(100, 5, shuffle = True, bias = 0, noise = 80 ,random_state=42)
66 x_train,x_test,t_train,t_test= train_test_split(x,t)
67 model_LR= LinearRegression()
68 model_R=Ridge()
69 w = model_LR.fit(x_train,t_train)
70 w1=model_R.fit(x_train,t_train)
71 y_train=model_LR.predict(x_train)
72 y_test=model_LR.predict(x_test)
73 y1_train=model_R.predict(x_train)
74 y1_test=model_R.predict(x_test)
75 score = r2_score(t_train,y_train)
76 score2=r2_score(t_test,y_test)
77 score3 = r2_score(t_train,y1_train)
78 score4=r2_score(t_test,y1_test)
79 print("R^2 for train data in LR: ",score)
80 print("R^2 for test data in LR: ",score2)
81 print("R^2 for train data in R: ",score3)
82 print("R^2 for test data in R: ",score4)
83
84
```

On the right, the Variable Explorer shows the following data:

| Name | Type | Size | Value |
|----------|-------------------------------------|------|--|
| i | int | 1 | 80 |
| model | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| model_LR | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| model_R | linear_model._ridge.Ridge | 1 | Ridge object of sklearn.linear_model._ridge module |
| n | list | 5 | [0, 10, 20, 40, 80] |
| q | list | 5 | [1.0, 0.9886666700826499, 0.9444308... |
| q1 | list | 5 | [1.0, 0.9654481645301916, 0.9179213... |

The IPython console shows the following output:

```
In [71]: runcell(3, 'E:/upes/Semester 4/Machine Learning/Lab/ML exp4.py')
In [72]: r2 for train data in LR:  0.540077201825693
R2 for test data in LR:  0.5333999968618625
R2 for train data in R:  0.5399286738887556
R2 for test data in R:  0.5377083696967986
In [73]:
```

At the bottom, the status bar indicates: LSP Python: ready, conda: base (Python 3.8.8), Line 73, Col 34, UTF-8, CRLF, RW, Mem 82%.

Experiment 5: Classification

Aim: To understand the concept of Classification

Code:

```
#% %
```

```
from sklearn.datasets import make_classification
```

```
x,t=make_classification(100,5,n_classes=2)
```

```
#% %
```

```
from sklearn.model_selection import train_test_split
```

```
import numpy as np
```

```
x0 = np.ones((100,1))
```

```
newX = np.concatenate((x0, x), axis =1)

x_train,x_test,t_train,t_test=train_test_split(newX,t,shuffle=True)

Xt = np.transpose(x_train)

Winv = np.dot(Xt,x_train)

W1 = np.linalg.inv(Winv)

W2 = np.dot(W1, Xt)

W = np.dot(W2,t_train)

#y=np.dot(np.transpose(W),x_test)

y=np.dot(x_test,W)

print(y)

#% %

y0=[]

for i in y:

    if i>=0:

        y0.append(1)

    else:

        y0.append(0)

print("y is:",y0)
```

```
#%%
```

```
count=0
```

```
for i in y0:
```

```
    if(i==1):
```

```
        count+=1
```

```
accuracy=count/len(y0)
```

```
print(accuracy)
```

Output:

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The left sidebar lists several open files: ML exp2.py, ML Exp2.1.py, ML exp2.2.py, ML exp4.py, and ML exp5.py. The main code editor window displays the following Python script:

```
7
8 #%%
9 from sklearn.datasets import make_classification
10 x,t=make_classification(100,5,n_classes=2)
11 #%%
12 from sklearn.model_selection import train_test_split
13 import numpy as np
14 x0 = np.ones((100,1))
15 newX = np.concatenate((x0, x), axis =1)
16 x_train,x_test,t_train,t_test=train_test_split(newX,t,shuffle=True)
17 Xt = np.transpose(x_train)
18 Winv = np.dot(Xt,x_train)
19 W1 = np.linalg.inv(Winv)
20 W2 = np.dot(W2,t_train)
21 W = np.dot(W2,np.transpose(W))
22 y=np.dot(np.transpose(W),x_test)
23 y=np.dot(x_test,W)
24 print(y)
25 #%%
26 y0=[]
27 for i in y:
28     if i>0:
29         y0.append(1)
30     else:
31         y0.append(0)
32 print("y is:",y0)
33 #%%
34 count=0
35 for i in y0:
36     if(i==1):
37         count+=1
38 accuracy=count/len(y0)
39 print(accuracy)
40
```

To the right of the code editor is the Variable Explorer window, which lists the following variables:

| Name | Type | Size | Value |
|---------|------------------|----------|---|
| i | int | 1 | 1 |
| newX | Array of float64 | (100, 6) | [[1. 0.19118814 0.06955743 -0.2165221 0.50700523 -0.20059 ...] |
| t | Array of int32 | (100,) | [0 0 0 ... 1 0 0] |
| t_test | Array of int32 | (25,) | [0 0 1 ... 0 1 1] |
| t_train | Array of int32 | (75,) | [1 0 0 ... 1 1 1] |
| W | Array of float64 | (6,) | [0.54014479 -0.26472737 0.22720156 -0.13042584 0.0024571 0.258225 ...] |
| W1 | Array of float64 | (6, 6) | [[1.33713337e-02 1.56555502e-03 -1.66274883e-03 -3.1... 4 ... [0.01285934 0.01378549 0.01268845 ... 0.01266871 ...] |

Below the Variable Explorer is the IPython console, which shows the execution of cells 39 through 42. Cell 39 runs `runcell(2, 'E:/upes/Semester 4/Machine Learning/Lab/ML exp5.py')`. Cell 40 runs `runcell(3, 'E:/upes/Semester 4/Machine Learning/Lab/ML exp5.py')`, displaying the value of `y` as a list of 1s and 0s. Cell 41 runs `runcell(4, 'E:/upes/Semester 4/Machine Learning/Lab/ML exp5.py')`, displaying the value of `0.8`. Cell 42 is currently executing.

Experiment 6: Multiclass Classification

Aim: Multiclass Classification

Code:

```
#% %  
  
from sklearn.datasets import make_classification  
  
import numpy as np  
  
from sklearn.model_selection import train_test_split  
  
classes = 3  
  
x, t = make_classification(100, 5, n_classes = classes, random_state= 20, n_informative = 2,  
n_clusters_per_class = 1)  
  
res = np.zeros((t.shape[0], classes), dtype=int)  
  
res[np.arange(t.shape[0]), t] = 1  
  
x0=np.ones((100,1), dtype=int);  
  
new_x = np.concatenate((x0,x), axis = 1);  
  
x_train,x_test,t_train,t_test=train_test_split(new_x ,res, shuffle = True );  
  
tr=x_train.transpose();  
  
m3 = np. dot(tr,x_train);  
  
i = np.linalg.inv(m3);  
  
m4 = np. dot(i,tr);
```

```
m5=np.dot(m4,t_train);

transp=m5.transpose();

print("the value of W is:-");

print(m5);

y = np. dot(x_test,m5);

print("the value of y is", y );

k=np.argmax(y , axis = 1);

print("the value of k is:- ",k)

#% %

count=0

for i in k:

    if(i==1 or i==2):

        count+=1

accuracy=count/len(k)

print(accuracy)
```

Output:

The screenshot shows the Spyder Python IDE interface. The code in the editor window is for generating a dataset and performing a decision tree classification. The variable explorer shows arrays m4, m5, new_x, res, t, t_test, t_train, and t_transpose. The IPython console displays the output of the code, including the value of k (2) and the accuracy (0.48). The status bar at the bottom indicates the environment is LSP Python: ready, conda: base (Python 3.8.8), Line 29, Col 32, UTF-8, CRLF, RW, Mem 81%.

```
6 """
7 #%%
8 from sklearn.datasets import make_classification
9 import numpy as np
10 from sklearn.model_selection import train_test_split
11 classes = 3
12 X, t = make_classification(100, 5, n_classes = classes, random_state= 20, n_informative= 2, n_clusters_per_class= 1)
13 res = np.zeros((t.shape[0], classes), dtype=int)
14 res[np.arange(t.shape[0]), t] = 1
15 X0=np.ones((100,1), dtype=int)
16 new_X = np.concatenate((X0,X), axis = 1);
17 X_train,X_test,t_train,t_test=train_test_split(new_X ,res, shuffle = True );
18 tr=X_train.transpose();
19 m3 = np.dot(tr,X_train);
20 i = np.linalg.inv(m3);
21 m4 = np.dot(i,tr);
22 m5=np.dot(m4,t_train);
23 transpm5= m5.transpose();
24 print("the value of W is:-");
25 print(m5);
26 y = np. dot(X_test,m5);
27 print("the value of y is", y );
28 k=np.argmax(y , axis = 1);
29 print("the value of k is:- ",k)
30 #%%
31 count=0
32 for i in k:
33     if(i==1 or i==2):
34         count+=1
35 accuracy=count/len(k)
36 print(accuracy)
```

In [2]: runcell(2, 'E:/upes/Semester 4/Machine Learning/Lab/untitled0.py')
0.48

In [3]: runcell(2, 'E:/upes/Semester 4/Machine Learning/Lab/untitled0.py')
0.24

Experiment 7: Decision Trees

Aim: To understand the concept of Decision trees using classification and regression

Code:

```
from sklearn.datasets import make_classification
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import tree
```

```
X, t = make_classification(100, 5, n_classes = 3, random_state= 70, n_informative = 2,  
n_clusters_per_class = 1)
```

```
X_train, X_test, t_train, t_test = train_test_split(X,t)
```

```
#% %
```

```
model = tree.DecisionTreeClassifier()
```

```
model.fit(X_train, t_train)

tree.plot_tree(model)

#% %

from sklearn.datasets import make_regression

X, t = make_regression(100, 5, random_state= 70,noise=10,bias=True)

X_train, X_test, t_train, t_test = train_test_split(X,t)

#% %

from sklearn import tree

model = tree.DecisionTreeRegressor(criterion="gini")

model.fit(X_train, t_train)

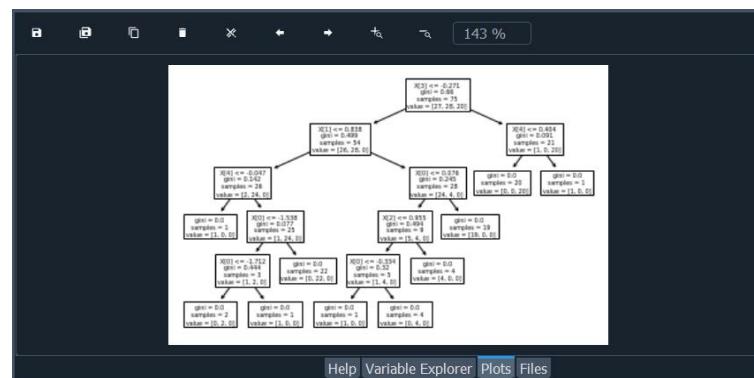
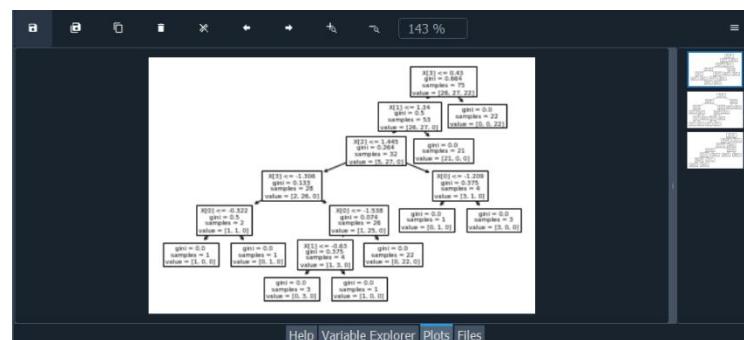
tree.plot_tree(model)
```

Output:

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named 'Desicion tree.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar 26 11:24:23 2022
4
5 @author: Lenovo
6 """
7
8 from sklearn.datasets import make_classification
9 from sklearn.model_selection import train_test_split
10 from sklearn import tree
11 X, t = make_classification(100, 5, n_classes = 3, random_state= 70, n_informative= 2, n_redundant= 0, n_clusters_per_class= 1, weights=[0.4, 0.3, 0.3], class_sep= 0.5)
12 X_train, X_test, t_train, t_test = train_test_split(X,t)
13 #%%%
14 model = tree.DecisionTreeClassifier()
15 model.fit(X_train, t_train)
16 tree.plot_tree(model)
17
18
```

On the right, the IPython console window shows the command 'runfile('E:/upes/Semester 4/Machine Learning/Lab/Desicion tree.py', wdir='E:/upes/Semester 4/Machine Learning/Lab')' and the resulting output, which is a visual representation of the decision tree.



Code of Decision tree using regression:

```
import numpy as np

from sklearn.datasets import make_classification

from sklearn.tree import DecisionTreeRegressor

from sklearn import tree

x, t = make_classification(100, 5, n_classes = 3, random_state= 40, n_informative = 2,
n_clusters_per_class = 1)

from sklearn.model_selection import train_test_split

x_train,x_test,t_train,t_test=train_test_split(x,t)

#% %

model=tree.DecisionTreeRegressor()

model.fit(x_train,t_train)

tree.plot_tree(model)
```

Output:

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named 'untitled5.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Apr 12 12:20:39 2022
4
5 @author: Lenovo
6 """
7
8 import numpy as np
9 from sklearn.datasets import make_classification
10 from sklearn.tree import DecisionTreeRegressor
11 from sklearn import tree
12 X, y = make_classification(100, 5, n_classes = 3, random_state= 40, n_informative= 2, n_redundant= 0, n_clusters_per_class= 1, weights= [0.4, 0.3, 0.3], class_sep= 0.5)
13 from sklearn.model_selection import train_test_split
14 X_train,X_test,y_train,y_test=train_test_split(X,y)
15 #
16 model=tree.DecisionTreeRegressor()
17 model.fit(X_train,y_train)
18 tree.plot_tree(model)
```

The right side of the interface features a 'Plots' tab where a decision tree diagram is visualized. Below the plots is a 'Console' tab showing the command run: 'In [17]: runfile('E:/upes/Semester 4/Machine Learning/Lab/untitled5.py', wdir='E:/upes/Semester 4/Machine Learning/Lab')'. The status bar at the bottom indicates the environment: 'LSP Python: ready'.

Experiment 8: Random Forest AdaBoost

Aim: To understand the concept of Random Forest and Adaboost using classification and regression

Code of Random Forest Classifier:

```
import pandas
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
data = {'CGPA':['g9','l9','g9','l9','g9'],
```

```
'Inter':['Y','N','N','N','Y'],  
  
'PK':['++','==','==','==','=='],  
  
'CS':['G','G','A','A','G'],  
  
'Job':['Y','Y','N','N','Y']}  
  
table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])  
  
table.where(table["CGPA"]=="g9").count()  
  
encoder=LabelEncoder()  
  
for i in table:  
  
    table[i]=encoder.fit_transform(table[i])  
  
X=table.iloc[:,0:4].values  
  
t=table.iloc[:,4].values  
  
X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=2)  
  
model = RandomForestClassifier(n_estimators=3)  
  
model.fit(X_train,t_train)  
  
if model.predict([[0,1,1,1]])==1:  
  
    print("Got JOB")  
  
else:  
  
    print("Didn't get JOB")
```

```

import matplotlib.pyplot as plt

from sklearn import tree

plt.figure(figsize=(15,10))

tree.plot_tree(model.estimators_[2],filled=True)

```

Output:

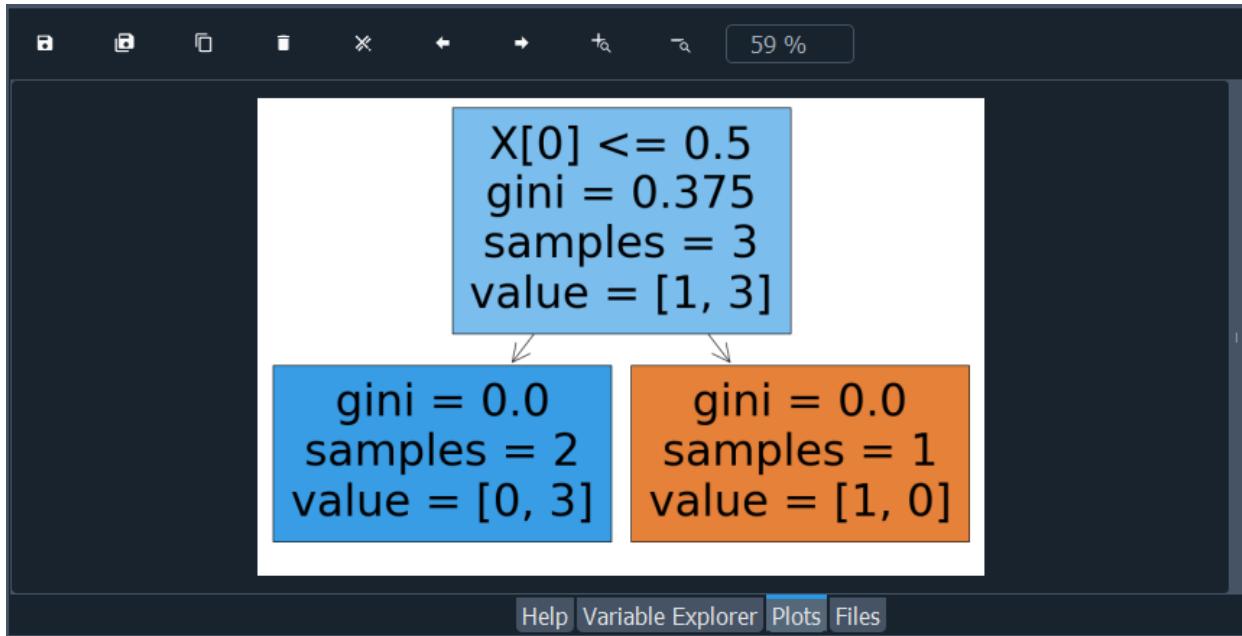
The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named `Random_Forest.py` containing the provided Python code for plotting a decision tree. On the right, the IPython console window shows the resulting decision tree structure. The root node has the condition $X[0] \leq 0.5$, a gini value of 0.375, and 3 samples. It branches into two nodes: one with a gini value of 0.0 and 2 samples (value [0, 3]), and another with a gini value of 0.0 and 1 sample (value [1, 0]). Below the tree plot, the IPython console shows the command runfile and the output "Got JOB".

```

@uthor: gopal.singh
"""

import pandas
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
data = {'CGPA':['g9','l9','g9','l9','g9'],
        'Inter':['Y','N','N','N','Y'],
        'PK':[==,'=','==','=','='],
        'CS':['G','G','A','A','G'],
        'Job':['Y','Y','N','N','Y']}
table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
table.where(table["CGPA"]=="g9").count()
encoder=LabelEncoder()
for i in table:
    table[i]=encoder.fit_transform(table[i])
X=table.iloc[:,0:4].values
X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=42)
model = RandomForestClassifier(n_estimators=3)
model.fit(X_train,t_train)
if model.predict([[0,1,1,1]])==1:
    print("Got JOB")
else:
    print("Didn't get JOB")
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(model.estimators_[2],filled=True)

```



Code of AdaBoost Classifier:

```
import pandas

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.ensemble import AdaBoostClassifier

data = {'CGPA':['g9','l9','g9','l9','g9'],
        'Inter':['Y','N','N','N','Y'],
        'PK':[ '+','==','==','==','=='],
        'CS':['G','G','A','A','G'],
        'Job':['Y','Y','N','N','Y']}

table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
```

```
table.where(table["CGPA"]=="g9").count()

encoder=LabelEncoder()

for i in table:

    table[i]=encoder.fit_transform(table[i])

X=table.iloc[:,0:4].values

t=table.iloc[:,4].values

X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=2)

model = AdaBoostClassifier(n_estimators=3)

model.fit(X_train,t_train)

if model.predict([[0,1,1,1]])==1:

    print("Got JOB")

else:

    print("Didn't get JOB")
```

Output:

The screenshot shows the Spyder Python IDE interface. On the left, there are several tabs for different files: Random_Forest.py, SVM.py, untitled0.py, AdaBoost Regression.py, and AdaBoost Classifier.py. The AdaBoost Classifier.py tab is active, displaying the following Python code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Apr 12 11:10:19 2022
4
5 @author: Lenovo
6 """
7
8 import pandas
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.model_selection import train_test_split
11 from sklearn.ensemble import AdaBoostClassifier
12 data = {'CGPA': ['g9', 'l9', 'g9', 'l9', 'g9'],
13         'Inter': ['Y', 'N', 'N', 'N', 'Y'],
14         'PK': ['++', '==', '==', '==', '=='],
15         'CS': ['G', 'G', 'A', 'A', 'G'],
16         'Job': ['Y', 'Y', 'N', 'N', 'Y']}
17 table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
18 table.where(table["CGPA"]=="g9").count()
19 encoder=LabelEncoder()
20 for i in table:
21     table[i]=encoder.fit_transform(table[i])
22 X=table.iloc[:,0:4].values
23 t=table.iloc[:,4].values
24 X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=2)
25 model = AdaBoostClassifier(n_estimators=3)
26 model.fit(X_train,t_train)
27 if model.predict([[0,1,1,1]])==1:
28     print("Got JOB")
29 else:
30     print("Didn't get JOB")

```

The right side of the interface shows the Variable Explorer and IPython console. The Variable Explorer displays variables like 'a' (float64), 'data' (dict), 'encoder' (preprocessing._label.LabelEncoder), 'h0' (list), 'i' (str), 'model' (ensemble._weight_boosting.AdaBoostClassifier), and 'score' (float64). The IPython console shows the command runfile('E:/upes/Semester 4/Machine Learning/Lab/AdaBoost Classifier.py', wdir='E:/upes/Semester 4/Machine Learning/Lab') followed by the output 'Got JOB'.

Code of Random Forest using Regression:

```

import pandas

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor

data = {'CGPA': ['g9', 'l9', 'g9', 'l9', 'g9'],

        'Inter': ['Y', 'N', 'N', 'N', 'Y'],

        'PK': ['++', '==', '==', '==', '=='],

        'CS': ['G', 'G', 'A', 'A', 'G'],

        'Job': ['Y', 'Y', 'N', 'N', 'Y']}}

table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])

```

```
table
```

```
table.where(table["CGPA"]=="g9").count()

encoder=LabelEncoder()

for i in table:

    table[i]=encoder.fit_transform(table[i])

X=table.iloc[:,0:4].values

t=table.iloc[:,4].values

X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=2)

model = RandomForestRegressor(n_estimators=3)

model.fit(X_train,t_train)

if model.predict([[0,1,1,1]])==1:

    print("Got JOB")

else:

    print("Didn't get JOB")

import matplotlib.pyplot as plt

from sklearn import tree

plt.figure(figsize=(15,10))

tree.plot_tree(model.estimators_[2],filled=True)
```

Output:

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named 'untitled3.py' containing Python code for AdaBoost regression. On the right, the IPython console shows the execution of the script, including the output of a decision tree plot. The plot shows two nodes: one for X[1] <= 0.5 with mse = 0.188, samples = 3, value = 0.75, and another for mse = 0.0, samples = 1, value = 0.0. Below the plot, the IPython console shows the command 'runfile' being run, followed by several 'In []:' prompts showing the progress of the job.

```

4
5 #author: Lenovo
6 """
7
8 import pandas
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.model_selection import train_test_split
11 from sklearn.ensemble import RandomForestRegressor
12 data = {'CGPA':['g9','l9','g9','l9','g9'],
13         'Inter':['Y','N','N','N','Y'],
14         'PK':[ '+','==','==','==','=='],
15         'CS':['G','G','A','A','G'],
16         'Job':['Y','Y','N','N','Y']}
17 table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
18 table.where(table["CGPA"]=="g9").count()
19 encoder=LabelEncoder()
20 for i in table:
21     table[i]=encoder.fit_transform(table[i])
22 X=table.iloc[:,0:4].values
23 t=table.iloc[:,4].values
24 X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=42)
25 model = RandomForestRegressor(n_estimators=3)
26 model.fit(X_train,t_train)
27 if model.predict([[0,1,1,1]])==1:
28     print("Got JOB")
29 else:
30     print("Didn't get JOB")
31
32 import matplotlib.pyplot as plt
33 from sklearn import tree
34 plt.figure(figsize=(15,10))
35 tree.plot_tree(model.estimators_[2],filled=True)

```

Code of AdaBoost using Regression:

```
import pandas
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import AdaBoostRegressor
```

```
data = {'CGPA':['g9','l9','g9','l9','g9'],
```

```
'Inter':['Y','N','N','N','Y'],
```

```
'PK':[ '+','==','==','==','=='],
```

```
'CS':['G','G','A','A','G'],
```

```
'Job':['Y','Y','N','N','Y']}
```

```
table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
```

```
table.where(table["CGPA"]=="g9").count()

encoder=LabelEncoder()

for i in table:

    table[i]=encoder.fit_transform(table[i])

X=table.iloc[:,0:4].values

t=table.iloc[:,4].values

X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=2)

model = AdaBoostRegressor(n_estimators=3)

model.fit(X_train,t_train)

if model.predict([[0,1,1,1]])==1:

    print("Got JOB")

else:

    print("Didn't get JOB")
```

Output:

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays 'Adabost Regression.py' with the following content:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Apr 12 11:08:00 2022
4
5 @author: Lenovo
6 """
7
8 import pandas
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.model_selection import train_test_split
11 from sklearn.ensemble import AdaBoostRegressor
12 data = {'CGPA': ['g9', 'l9', 'g9', 'l9', 'g9'],
13         'Inter': ['Y', 'N', 'N', 'N', 'Y'],
14         'PK': ['++', '==', '==', '==', '=='],
15         'CS': ['G', 'G', 'A', 'A', 'G'],
16         'Job': ['Y', 'Y', 'N', 'N', 'Y']}
17 table=pandas.DataFrame(data,columns=['CGPA','Inter','PK','CS','Job'])
18 table.where(table['CGPA']=='g9').count()
19 encoder=LabelEncoder()
20 for i in table:
21     table[i]=encoder.fit_transform(table[i])
22 X=table.iloc[:,0:4].values
23 t=table.iloc[:,4].values
24 X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.2,random_state=2)
25 model = AdaBoostRegressor(n_estimators=3)
26 model.fit(X_train,t_train)
27 if model.predict([[0,1,1,1]])==1:
28     print("Got JOB")
29 else:
30     print("Didn't get JOB")
31

```

The Variable Explorer on the right shows the following variables:

| Name | Type | Size | Value |
|---------|---|------|----------------------|
| a | float64 | 1 | 1.514083276823... |
| data | dict | 5 | {'CGPA': ['g9', ...] |
| encoder | preprocessing._label.LabelEncoder | 1 | LabelEncoder(...) |
| h0 | list | 1 | [Line2D] |
| i | str | 3 | Job |
| model | ensemble._weight_boosting.AdaBoostRegressor | 1 | AdaBoostRegres... |
| score | float64 | 1 | 0.933333333333... |

The IPython console at the bottom shows the execution of the script:

```

In [10]: runfile('E:/upes/Semester 4/Machine Learning/Lab/Adabost Regression.py',
      wdir='E:/upes/Semester 4/Machine Learning/Lab')
Got JOB

In [11]:

```

Code of Decision Stump:

```

from sklearn.metrics import accuracy_score

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

import pandas

data = {'CGPA':['g9','l9','g9','l9','g9'],

        'Inter':['Y','N','N','N','Y'],

        'PK':['++','==','==','==','=='],

        'CS':['G','G','A','A','G'],

        'Job': ['Y','Y','N','N','Y']}

```

```
'CS':['G','G','A','A','G'],
'Job':['Y','Y','N','N','Y']}}

table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])

encoder=LabelEncoder()

for i in table:

    table[i]=encoder.fit_transform(table[i])

X=table.iloc[:,0:4].values

t=table.iloc[:,4].values

X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.1,random_state=2)

print("\nDecision Tree Classifier")

model = tree.DecisionTreeClassifier()

model = model.fit(X_train, t_train)

predicted_value = model.predict(X_test)

print(predicted_value)

tree.plot_tree(model)
```

```

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(t_test, predicted_value)

print("Accuracy: ",accuracy)

```

Output:

```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\upes\Semester 4\Machine learning\Lab\Desicion stump.py
3 Created on Tue Apr 12 12:10:13 2022
4
5 #author: Lenovo
6 """
7
8 from sklearn.metrics import accuracy_score
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.model_selection import train_test_split
11 from sklearn.tree import DecisionTreeClassifier
12 from sklearn import tree
13 import pandas
14 data = {'CGPA':['g9','l9','g9','l9','g9'],
15         'Inter':['Y','N','N','W','V'],
16         'PK':[ '+', '=', '==', '!=', '=='],
17         'CS': ['G','G','A','A','G'],
18         'Job':['Y','Y','N','N','Y']}
19 table=pandas.DataFrame(data,columns=['CGPA","Inter","PK","CS","Job"])
20 encoder=LabelEncoder()
21 for i in table:
22     table[i]=encoder.fit_transform(table[i])
23 X=table.iloc[:,0:4].values
24 t=table.iloc[:,4].values
25 X_train,X_test,t_train,t_test=train_test_split(X,t,test_size=0.1,random_state=42)
26 print("\nDecision Tree Classifier")
27 model = tree.DecisionTreeClassifier()
28 model = model.fit(X_train, t_train)
29 predicted_value = model.predict(X_test)
30 print(predicted_value)
31 tree.plot_tree(model)
32 from sklearn.metrics import accuracy_score
33 accuracy = accuracy_score(t_test, predicted_value)
34 print("Accuracy: ",accuracy)

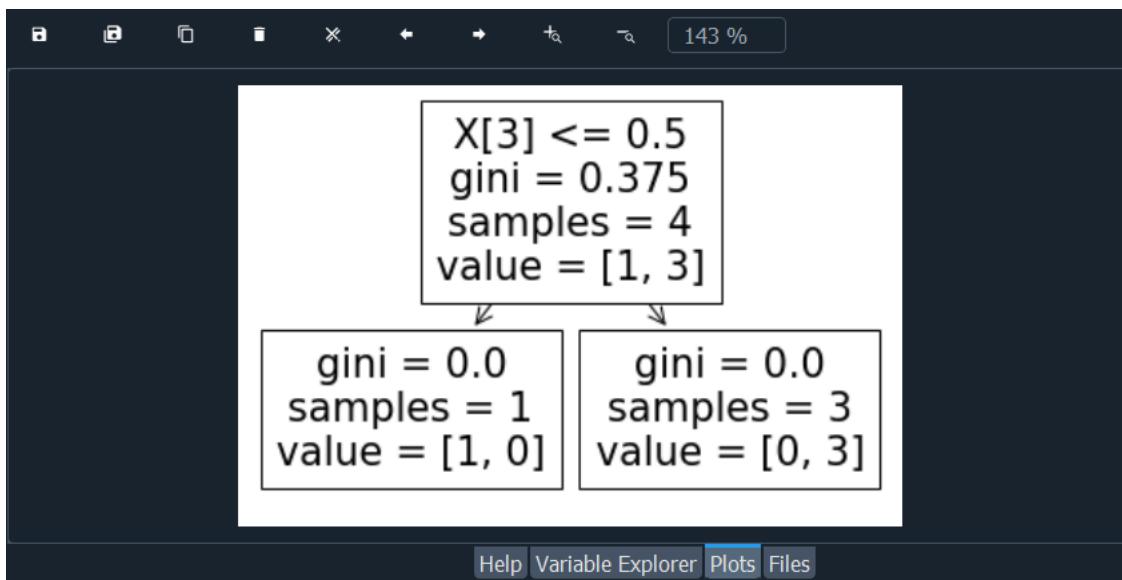
```

In [12]: runfile('E:/upes/Semester 4/Machine Learning/Lab/untitled3.py', wdir='E:/upes/Semester 4/Machine Learning/Lab')
Got JOB

In [13]: runfile('E:/upes/Semester 4/Machine Learning/Lab/Desicion stump.py', wdir='E:/upes/Semester 4/Machine Learning/Lab')
Decision Tree Classifier
[0]
Accuracy: 1.0

In [14]:

LSP Python: ready conda: base (Python 3.8.8) Line 14, Col 43 UTF-8 CRLF RW Mem 87%



Experiment 9: Neural Network

Aim: To understand the concept of Neural Network

Code:

```
import numpy as np

def abs(x):

    return x if x>0 else -x

def sigmoid (x):

    return 1/(1 + np.exp(-x))

def sigmoid_derivative(x):

    return x * (1 - x)

def checkError(predicted_output):

    expected_output = [[0],[1],[1],[0]]

    for i,j in zip(expected_output , predicted_output):

        if abs(i[0]-j[0]) > 0.001:

            return True

    return False

inputs = np.array([[0,0],[0,1],[1,0],[1,1]])

expected_output = np.array([[0],[1],[1],[0]])

epoch = 0
```

```
lr = 0.1

inputLayerNeurons = int(input("enter no of inputLayer"))

hiddenLayerNeurons = int(input("enter no of hiddenlayer "))

outputLayerNeurons = int(input("enter no of outputlayer "))

hidden_weights = []

for i in range(1,inputLayerNeurons+1):

    hidden_weights_ind = []

    for j in range(inputLayerNeurons+1,inputLayerNeurons+hiddenLayerNeurons+1):

        hidden_weights_ind.append(float(input('w'+str(i)+str(j)))))

    hidden_weights.append(hidden_weights_ind)

output_weights = []

for i in range(inputLayerNeurons+1,inputLayerNeurons+hiddenLayerNeurons+1):

    output_weights_ind = []

    for j in range(inputLayerNeurons+hiddenLayerNeurons+1,inputLayerNeurons+hiddenLayerNeurons+outputLayerNeurons+1):

        output_weights_ind.append(float(input('w'+str(i)+str(j)))))

    output_weights.append(output_weights_ind)
```

```
hidden_bias = []

output_bias = []

for i in range(inputLayerNeurons+1,inputLayerNeurons+hiddenLayerNeurons+outputLayerNeurons+1):

    if i > inputLayerNeurons+hiddenLayerNeurons:

        output_bias.append(float(input("o"+str(i)))))

    else:

        hidden_bias.append(float(input("o"+str(i)))))

hidden_weights = np.asarray(hidden_weights)

hidden_bias = np.asarray([hidden_bias])

output_weights = np.asarray(output_weights)

output_bias = np.asarray([output_bias])

print("Initial hidden weights: ",end="")

print(*hidden_weights)

print("Initial hidden biases: ",end="")

print(*hidden_bias)

print("Initial output weights: ",end="")

print(*output_weights)

print("Initial output biases: ",end="")
```

```
print(*output_bias)

predicted_output = [[0],[0],[0],[0]]

while checkError(predicted_output):

    epoch += 1

    #Forward Propagation

    hidden_layer_activation = np.dot(inputs,hidden_weights)

    hidden_layer_activation += hidden_bias

    hidden_layer_output = sigmoid(hidden_layer_activation)

    output_layer_activation = np.dot(hidden_layer_output,output_weights)

    output_layer_activation += output_bias

    predicted_output = sigmoid(output_layer_activation)

    error = expected_output - predicted_output

    d_predicted_output = error * sigmoid_derivative(predicted_output)

    error_hidden_layer = d_predicted_output.dot(output_weights.T)

    d_hidden_layer = error_hidden_layer * sigmoid_derivative(hidden_layer_output)

    output_weights += hidden_layer_output.T.dot(d_predicted_output) * lr

    output_bias += np.sum(d_predicted_output, axis=0, keepdims=True) * lr

    hidden_weights += inputs.T.dot(d_hidden_layer) * lr
```

```
hidden_bias += np.sum(d_hidden_layer, axis=0, keepdims=True) * lr

print("Final hidden weights: ", end="")

print(*hidden_weights)

print("Final hidden bias: ", end="")

print(*hidden_bias)

print("Final output weights: ", end="")

print(*output_weights)

print("Final output bias: ", end="")

print(*output_bias)

print("\nOutput from neural network: ", end="")

print(*predicted_output)

print("\nNo of epochs")

print(epoch)
```

Output:

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Lenovo\Downloads\NN_1 (1).py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 22 12:18:38 2022
4
5 @author: gopal.singh
6 """
7 import numpy as np
8 def abs(x):
9     return x if x>0 else -x
10 def sigmoid(x):
11     return 1/(1 + np.exp(-x))
12 def sigmoid_derivative(x):
13     return x * (1 - x)
14 def checkError(predicted_output):
15     expected_output = [[0],[1],[1],[0]]
16     for i,j in zip(expected_output , predicted_output):
17         if abs(i[0]-j[0]) > 0.001:
18             return True
19     return False
20 inputs = np.array([[0,0],[0,1],[1,0],[1,1]])
21 expected_output = np.array([[0],[1],[1],[0]])
22 epoch = 0
23 lr = 0.1
24 inputLayerNeurons = int(input("enter no of inputLayer"))
25 hiddenLayerNeurons = int(input("enter no of hiddenlayer "))
26 outputLayerNeurons = int(input("enter no of outputlayer "))
27 hidden_weights = []
28 for i in range(1,inputLayerNeurons+1):
29     hidden_weights.append([])
30     for j in range(hiddenLayerNeurons+1):
31         hidden_weights[-1].append(float(input('w'+str(i)+str(j))))
```

Console 1/A X

```
In [15]: runfile('C:/Users/Lenovo/Downloads/NN_1 (1).py', wdir='C:/Users/Lenovo/Downloads')

enter no of inputLayer 2
enter no of hiddenlayer 3
enter no of outputlayer 1
w13 0.2
w14 0.2
w15 0.3
w23 0.4
w24 0.5
w25 0.5
w36 0.3
w46 0.4
w56 0.3
o3 0.5
o4 0.5
o5 0.5
```

LSP Python: ready conda: base (Python 3.8.8) Line 31, Col 1 UTF-8 CRLF RW Mem 86%

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Lenovo\Downloads\NN_1 (1).py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 22 12:18:38 2022
4
5 @author: gopal.singh
6 """
7 import numpy as np
8 def abs(x):
9     return x if x>0 else -x
10 def sigmoid(x):
11     return 1/(1 + np.exp(-x))
12 def sigmoid_derivative(x):
13     return x * (1 - x)
14 def checkError(predicted_output):
15     expected_output = [[0],[1],[1],[0]]
16     for i,j in zip(expected_output , predicted_output):
17         if abs(i[0]-j[0]) > 0.001:
18             return True
19     return False
20 inputs = np.array([[0,0],[0,1],[1,0],[1,1]])
21 expected_output = np.array([[0],[1],[1],[0]])
22 epoch = 0
23 lr = 0.1
24 inputLayerNeurons = int(input("enter no of inputLayer"))
25 hiddenLayerNeurons = int(input("enter no of hiddenlayer "))
26 outputLayerNeurons = int(input("enter no of outputlayer "))
27 hidden_weights = []
28 for i in range(1,inputLayerNeurons+1):
29     hidden_weights.append([])
30     for j in range(hiddenLayerNeurons+1):
31         hidden_weights[-1].append(float(input('w'+str(i)+str(j))))
```

Console 1/A X

```
w36 0.3
w46 0.4
w56 0.3
o3 0.5
o4 0.5
o5 0.5
o6 0.5
Initial hidden weights: [0.2 0.2 0.3] [0.4 0.5 0.5]
Initial hidden biases: [0.5 0.5 0.5]
Initial output weights: [0.3] [0.4] [0.3]
Initial output biases: [0.5]
```

LSP Python: ready conda: base (Python 3.8.8) Line 31, Col 1 UTF-8 CRLF RW Mem 87%

```
In [15]: runfile('C:/Users/Lenovo/Downloads/NN_1 (1).py', wdir='C:/Users/Lenovo/Downloads')

enter no of inputLayer 2

enter no of hiddenlayer 3

enter no of outputlayer 1

w13 0.2

w14 0.2

w15 0.3

w23 0.4

w24 0.5

w25 0.5

w36 0.3

w46 0.4

w56 0.3

o3 0.5

o4 0.5

o5 0.5
```

Console 1/A X

```
w25 0.5

w36 0.3

w46 0.4

w56 0.3

o3 0.5

o4 0.5

o5 0.5

o6 0.5
Initial hidden weights: [0.2 0.2 0.3] [0.4 0.5 0.5]
Initial hidden biases: [0.5 0.5 0.5]
Initial output weights: [0.3] [0.4] [0.3]
Initial output biases: [0.5]
```

Experiment 10: Support vector machine

Aim: To understand the concept of support vector machine.

Code:

```
import numpy as np

from sklearn.datasets import make_classification

x, t = make_classification(100, 5, n_classes = 3, random_state= 50, n_informative = 3,
n_clusters_per_class = 1)

from sklearn.model_selection import train_test_split

x_train,x_test, y_train , y_test = train_test_split(x,t,test_size = 0.3 )

from sklearn import svm

model = svm.SVC(kernel='linear')

model.fit(x_train,y_train)

y_pred = model.predict(x_test)

from sklearn import metrics

score = metrics.accuracy_score(y_test,y_pred)

print(score)

#% %

import matplotlib.pyplot as plt

w = model.coef_[0]
```

```

a = -w[0]/w[1]

xx = np.linspace(-3, 3)

yy = a *xx - model.intercept_[0]/w[1]

h0 = plt.plot(xx,yy,'k-',label="non weighted div" )

plt.scatter(x_train[:,0],x_train[:,1],c=y_train)

plt.legend()

plt.show()

```

Output:

The screenshot shows the Spyder Python 3.8 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The left sidebar lists files: Random_Forest.py, SVM.py (current file), untitled0.py*, Adaboost Regression.py, and AdaBoost Classifier.py. The main workspace displays the code for SVM.py. The code imports numpy, sklearn.datasets, sklearn.model_selection, and sklearn.svm. It creates a dataset with 100 samples, 3 classes, and 50 informative features. It splits the data into training and testing sets. A linear SVC model is fitted to the training data. Predictions are made on the test set, and the accuracy is printed. Finally, it plots the decision boundary and data points. The right side of the interface shows the plot pane with a scatter plot of data points colored by class (yellow, purple, green) and a black line representing the non-weighted decision boundary. Below the plot is a warning message about inline plotting. The bottom status bar indicates the environment is LSP Python: ready, conda: base (Python 3.8.8), Line 9, Col 50, UTF-8, CRLF, RW, Mem 90%.

```

E:\upes\Semester 4\Machine learning\Lab\SVM.py
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\upes\Semester 4\Machine learning\Lab
133 %
In [1]: runfile('E:/upes/Semester 4/Machine Learning/Lab/SVM.py', wdir='E:/upes/Semester 4/Machine Learning/Lab')
0.9333333333333333
Warning
Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

IPython console History
LSP Python: ready conda: base (Python 3.8.8) Line 9, Col 50 UTF-8 CRLF RW Mem 90%

```

