POORNIMA INSTITUTE OF ENGINEERING & TECHNOLOGY, JAIPUR

Department of Computer Engineering

Lab Manual

Machine Learning Lab

<6CS4-22>

Branch	**CS**	**Name of Lab**	**ML Lab**
Session	**2019-20**	**Subject Code**	**6CS4-22**
Year	**3rd Year**	**Faculty**	**Pooja Sharma**
Semester	**6th Semester**	**Lab Assistant**	

Document No.	PIET/CS/2019/IV/LAB/ML/ 6CS4-22	Created By	
Version		Verified By	
Authorized By (HOD)			

LAB RULES

DO'S	DON'TS
Be regular to the lab.	Do not come late to the lab.
Follow proper dress code.	Do not throw the connecting wires on the floor.
Maintain Silence.	Do not operate µp/IC trainer kits unnecessarily.
Know the theory behind the experiment before coming to the lab.	Don't bring any external material inside the LAB.
Arrange the chairs/stools and equipment properly before leaving thelab.	Do not panic if you don't get the output.
Avoid unnecessary talking while doing the experiment.	Don't carry any LAB equipment outside the lab.
Keep the Table clean.	Do not try to repair or tamper lab equipment.

TABLE OF CONTENTS

INSTRUCTIONS

Before entering in the lab

All the students are supposed to prepare the theory regarding the next experiment.

Students are supposed to bring the practical file and the lab copy.

Previous programs should be written in the practical file.

All the students must follow the instructions, failing which he/she may not be allowed in the lab.

While working in the lab

Adhere to experimental schedule as instructed by the lab in-charge.

Get the previously executed program signed by the instructor.

Get the output of the current program checked by the instructor in the lab copy.

Each student should work on his/her assigned computer at each turn of the lab.

Take responsibility of valuable accessories.

Concentrate on the assigned practical and do not play games

If anyone caught red handed carrying any equipment of the lab, then he/she will have to face serious consequences.

Syllabus

SN List of Experiments

1 Implement and demonstrate the FIND-Salgorithm for finding the most specific ypothesis based on a given set of training data samples. Read the training data from a .CSV file.

2 For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithmto output a description of the set of all hypotheses consistent with the training examples.

3 Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge toclassify a new sample

4 Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets

5 Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

6 Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.

7 Write a program to construct aBayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

8 Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

9 Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

10 Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

MARKS SCHEME

RTU Marks Scheme

Maximum Marks Allocation		
Sessional	End-Term	Total
45	30	75

Marks Division

Mid Term I & II		
Practical	Viva	Total
22	8	30
Attendance & Performance		
Performance	Attendance	Total
22	8	30
End-Term Practical		
Practical	Viva	Total
22	8	30

Internal Assessment System

Total Marks – 10

Attendance	Discipline	Performance	Record	Viva	Total
2	2	2	2	2	10

LAB PLAN

Total number of experiment: - 10

Experiment Number	Turns	Scheduled Day
EXP-1	1	Week -1
EXP-2	1	Week -2
EXP-3	1	Week -3
EXP-4	1	Week -4
EXP-5	1	Week -5
EXP-6	1	Week -6
EXP-7	1	Week -7
EXP-8	1	Week -8
EXP-9	1	Week -9
EXP-10	1	Week -10

Distribution of lab hours

Attendance	05 minutes
Explanation of concept	15 minutes
Explanation of experiment	30 minutes
Performance of experiment	60 minutes
Viva / Quiz / Queries / Evaluation of Records	10 minutes
Total	**120 minutes**
Software required	Anaconda Python Distribution Navigator

6CS4-22 Machine Learning Lab Plan

No.	Contents	Experiments	Lab Turn
1	**Get the data**	• Getting the dataset • Importing libraries • Importing datasets • Understand Your Data With Descriptive Statistics • Understand Your Data With Visualization	Turn-01
2	**Prepare Your Data For Machine Learning**	• Data Cleaning • Encoding Text & Categorical Data • Rescale data. • Standardize data. • Normalize data. • Binarize data. • Splitting dataset into training and test set • Feature scaling	
4	**Train model using Supervised Learning-Regression Analysis**	• Linear Regression • Multiple Linear Regression • Non-Parametric Locally Weighted Regression Algorithm	Turn-02
5	**Train model using Supervised Learning-Classification**	• Logistic Regression	Turn-03
6		• Decision Tree	Turn-04
7		• Naïve Bayes classifier	Turn-05
8		• K nearest neighbor	Turn-06
9		• Support Vector Machine	Turn-07
10		• Random forest algorithm	Turn-08
11		• Artificial Neural Network by implementing the Back propagation	Turn-09
12	**Train model Unsupervised Learning**	K-Means & EM	Turn-10
13		Hierarchical Clustering & A priori Algorithm	Turn-11
10	**Selecting hypotheses**	FIND-S algorithm Candidate-Elimination algorithm	Turn-12
11	**Compare & Evaluate Machine Learning Algorithms and Project implementation**	Assignment of End –End Machine Learning Project	Turn-13

EXPERIMENT 1

Lab Objective:

1. Get the data
2. Data Preprocessing
3. Prepare Your Data For Machine Learning

Data Preprocessing in Machine learning

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

```
# In[4]: Importing Libraries
import pandas as pd
import numpy as np

# In[5]:

data=pd.read_csv('Data.csv')

# In[6]:

print(data.shape)

# In[14]:

data.head(20)

# In[15]:

data.info()

# In[16]:


types = data.dtypes
```

```
print(types)

# In[7]: Extracting dependent and independent variables:

x= data.iloc[:,:-1].values

# In[8]:

print(x)

# In[9]:

y=data.iloc[:,3].values
print(y)

# In[10]:

y= data.iloc[:,3].values

# In[102]:

set_option('display.width', 100)
set_option('precision', 3)
description = data.describe()
print(description)


# In[27]:


class_counts = data.groupby('Purchased').size()
print(class_counts)

# In[28]:

set_option('display.width', 100)
set_option('precision', 3)
correlations = data.corr(method='pearson')
print(correlations)

# In[29]:

skew = data.skew()
print(skew)


# In[30]:

import matplotlib.pyplot as plt
```

```
# In[31]:

from matplotlib import pyplot

# In[32]:
data.hist()
pyplot.show()


# In[33]:


data.plot(kind='density', subplots=True, layout=(3,3), sharex=False)
pyplot.show()

# In[34]:

data.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
pyplot.show()

# In[18]:
#handling missing data (Replacing missing data with the mean value)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy= 'mean', fill_value=None, verbose=0,
copy=True)
#Fitting imputer object to the independent variables x.
imputer= imputer.fit(x[:, 1:3])
#Replacing missing data with the calculated mean value
x[:, 1:3]= imputer.transform(x[:, 1:3])

# In[12]:
print(x)
# In[22]:

#Catgorical data
#for Country Variable
from sklearn.preprocessing import LabelEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
print(x)


# In[24]:
#for Country Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
#Encoding for dummy variables
onehot_encoder= OneHotEncoder(categorical_features= [0])
x= onehot_encoder.fit_transform(x).toarray()
```

```
print(x)


# In[26]:
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)
print(y)


# In[30]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)


# In[28]:
from sklearn.preprocessing import StandardScaler


# In[33]:
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
print(x_train)


# In[34]:
x_test= st_x.transform(x_test)
print(x_test)
```

EXPERIMENT 2

Lab Objective:

- Linear Regression
- Multiple Linear Regression

Problem Statement –

Read Data set of Salary from salary.csv Apply Linear Regression and make prediction.

```
# In[1]:

import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd


# In[2]:
data_set= pd.read_csv('Salary_Data.csv')


# In[4]:
print(data_set.head(10))


# In[5]:
x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values


# In[10]:
print(x)


# In[6]:
print(y)


# In[7]:
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_state=0)


# In[8]:
#Fitting the Simple Linear Regression model to the training dataset
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
```

```
regressor.fit(x_train, y_train)


# In[12]:
print(regressor.intercept_)


# In[13]:
print(regressor.coef_)


# In[9]:
#Prediction of Test and Training set result
y_pred= regressor.predict(x_test)
x_pred= regressor.predict(x_train)


# In[10]:
mtp.scatter(x_train, y_train, color="green")
mtp.plot(x_train, x_pred, color="red")
mtp.title("Salary vs Experience (Training Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()


# In[11]:
#visualizing the Test set results
mtp.scatter(x_test, y_test, color="blue")
mtp.plot(x_train, x_pred, color="red")
mtp.title("Salary vs Experience (Test Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()


# In[14]:
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Problem Statement –

Read Data set of multiple input variable Apply Multiple Linear Regression and make prediction.

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

```
#importing datasets
data_set= pd.read_csv('50_Startups.csv')
#Extracting dependent and independent Variables:
x=data_set.iloc[:, :-1].values
y=data_set.iloc[:, 4].values
#Encoding Dummy Variables:
#Catgorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_x= LabelEncoder()
x[:, 3]= labelencoder_x.fit_transform(x[:,3])
onehotencoder= OneHotEncoder(categorical_features= [3])
x= onehotencoder.fit_transform(x).toarray()

#avoiding the dummy variable trap:
x = x[:, 1:]
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)

#Fitting the MLR model to the training set:
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)

#Predicting the Test set result;
y_pred= regressor.predict(x_test)
print('Train Score: ', regressor.score(x_train, y_train))
print('Test Score: ', regressor.score(x_test, y_test) )
```

EXPERIMENT 3

Lab Objective:

- **Train model using Supervised Learning-Classification**
- Decision Tree

Problem Statement-Take Diabetes Dataset apply preprocessing train classifier using decision tree and make prediction and calculate accuracy.

```
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model_selection import train_test_split # Import train_test_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

```
pima = pd.read_csv("diabetes.csv")
pima.head()
X= pima.iloc[:, :-1].values
y= pima.iloc[:,8].values
print(X)
print(y)
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

```
graph.write_png('diabetes.png')
Image(graph.create_png())
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes1.png')
Image(graph.create_png())
```

EXPERIMENT 4

Lab Objective:

- **Train model using Supervised Learning-Classification**
- Naïve Bayes classifier

Problem Statement-Take Social Network Ads Dataset apply preprocessing , train classifier using naïve bayes, make prediction and calculate accuracy.

```
# Naive Bayes


# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values


print(X)
print(y)


# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)


# Predicting the Test set results
y_pred = classifier.predict(X_test)


# Making the Confusion Matrix
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))


print("Accuracy:",metrics.accuracy_score(y_test, y_pred))




# Visualising the Training set results
from matplotlib.colors import ListedColormap
```

```
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

EXPERIMENT 5

Lab Objective:

- **Train model using Supervised Learning-Classification**
- K nearest neighbor & Logistic Regression

Problem Statement- Apply KNN & Logistic Regression on dataset & compare the accuracy of both Models.

```
# K-Nearest Neighbors (K-NN)
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import metrics
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values


# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Making the Confusion Matrix
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
```

```
plt.title('K-NN (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()
```

Logistic Regression

# Logistic Regression

```
# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('Social_Network_Ads.csv')


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

from sklearn.preprocessing import LabelEncoder

label_encoder_X= LabelEncoder()

X[:, 0]= label_encoder_X.fit_transform(X[:, 0])

print(X)


#X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from sklearn.svm import SVC


#Create a svm Classifier
clf = SVC(kernel='linear') # Linear Kernel


#Train the model using the training sets
clf.fit(X_train, y_train)


#Predict the response for test dataset
y_pred = clf.predict(X_test)


from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
from sklearn.svm import SVR
#Create a svm Classifier
clf = SVR(kernel='rbf') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```
# Making the Confusion Matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)


# Visualising the Training set results

from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),

                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

             alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):

    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],

                c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Logistic Regression (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()


# Visualising the Test set results

from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test
```

```
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

EXPERIMENT 6

Lab Objective:

- **Train model using Supervised Learning-Classification & Regression**
- SVM & Random Forest

Problem Statement- Apply SVM & Random Forest on dataset & compare the accuracy of both Models also apply regressor.

```
import pandas as pd

data = pd.read_csv('Social_Network_Ads.csv')

print(data.head(10))

data.shape

x=data.iloc[:,1:4].values

print(x)

y=data.iloc[:,4].values

print(y)


from sklearn.preprocessing import LabelEncoder

labelencoder_x= LabelEncoder()

x[:, 0]= labelencoder_x.fit_transform(x[:,0])

print(x)


from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.30,random_state =0)

print(x_test)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
print(x_train)
print(x_test)

from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train,y_train)# Train the data
y_pred = clf.predict(x_test)
print(y_pred)

from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
from sklearn import metrics
print('accuracy:',metrics.accuracy_score(y_test,y_pred))
from sklearn.svm import SVR
clf = SVR(kernel='linear')
clf.fit(x_train,y_train)# Train the data
y_pred = clf.predict(x_test)

print('meanAbsoluteError',metrics.mean_absolute_error(y_test,y_pred))
print('meanSquaredError',metrics.mean_squared_error(y_test,y_pred))
```

Random Forest

```
import pandas as pd

data = pd.read_csv('Social_Network_Ads.csv')

print(data.head(10))

data.shape

x=data.iloc[:,1:4].values

print(x)

y=data.iloc[:,4].values

print(y)


from sklearn.preprocessing import LabelEncoder

labelencoder_x= LabelEncoder()

x[:, 0]= labelencoder_x.fit_transform(x[:,0])

print(x)


from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.30,random_state =0)

print(x_test)


from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)

print(x_train)

print(x_test)


from sklearn.ensemble import RandomForestClassifier
```

```
clf = RandomForestClassifier()
clf.fit(x_train,y_train)# Train the data
y_pred = clf.predict(x_test)
print(y_pred)

from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
from sklearn import metrics
print('accuracy:',metrics.accuracy_score(y_test,y_pred))

from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor()
clf.fit(x_train,y_train)# Train the data
y_pred = clf.predict(x_test)
print(y_pred)

print('meanAbsoluteError',metrics.mean_absolute_error(y_test,y_pred))
print('meanSquaredError',metrics.mean_squared_error(y_test,y_pred))
```