B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

Machine Learning

Submitted in partial fulfillment for the 6th Semester Laboratory

Bachelor of Technology
in
Computer Science and Engineering

Submitted by:

AKSHAY S BHARADWAJ

1BM18CS011

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Mar-June 2021

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Big Data Analytics (20CS6PEBDA) laboratory has been carried out by AKSHAY S BHARADWAJ (1BM18CS011) during the 6th Semester Mar-June-2021.

Signature of the Faculty Incharge:

NAME OF THE FACULTY:

Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

PROGRAM – 1

Question:

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples.

Program:

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/akshaysb2k/find-s.csv
/kaggle/input/finds/data.csv
```

In [2]:

```
data = pd.read_csv("/kaggle/input/finds/data.csv")
print("The entered data is \n")
```

```
print(data,"\n")
d = np.array(data)[:,:-1]
print("\n The attributes are: \n",d)
target = np.array(data)[:,-1]
print("\n The target is: ",target)
def training(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
print("\n The final hypothesis is:",training(d,target))
```

Output:

```
The entered data is

  Weather Temperature Humidity Goes
0  Sunny        Warm     Mild  Yes
1   Rainy        Cold     Mild   No
2  Sunny     Moderate    Nomal  Yes
3  Sunny        Cold     High   Yes


 The attributes are:
 [['Sunny ' 'Warm ' 'Mild']
 ['Rainy' 'Cold' 'Mild']
 ['Sunny ' 'Moderate' 'Nomal']
 ['Sunny ' 'Cold' 'High ']]

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

 The final hypothesis is: ['Sunny ' '?' '?']
```

Fig 1. Entered data for Find-S with final hypothesis output

PROGRAM – 2

Question:

For a given set of training data examples stored in a .csv file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all the hypothesis consistent with the training examples.

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
/kaggle/input/candidate-elimination/week2_data.csv

In [2]:
data = pd.read_csv("../input/candidate-elimination/week2_data.csv")
```

```
print("The entered data is: \n")
print(data,"\n")
atts = np.array(data.iloc[:,:-1])
target = np.array(data.iloc[:,-1])
print("\nThe attributes are: ")
print(atts)
print("\nThe target is: ")
print(target)
```

In [3]:

```
def learn(atts, target):
    s = atts[0].copy()
    print("\nSpecific Hypothesis: \n",s)
    g = [["?" for i in range(len(s))] for i in range(len(s))]
    print("\nGeneral Hypothesis: \n",g)
    for i, h in enumerate(atts):
        if target[i] == "Yes":
            print("\nPositive Example")
            for x in range(len(s)):
                if h[x] != s[x]:
                    s[x] = '?'
                    g[x][x] = '?'
        if target[i] == "No":
            print("\nNegative Example")
            for x in range(len(s)):
                if h[x] != s[x]:
                    g[x][x] = s[x]
                else:
                    g[x][x] = '?'
        print("Step: ", i+1)
        print("Specific Hypothesis: ")
        print(s)
        print("General Hypothesis : ")
        print(g)
    indices = [i for i, val in enumerate(g) if val == ['?', '?', '?', '?', '?
', '?']]
    for i in indices:
        g.remove(['?', '?', '?', '?', '?', '?'])
    return s, g
```

In [4]:

```
s_final,g_final = learn(atts, target)
print("\nFinal Specific Hypothesis:", s_final, sep="\n")
print("\nFinal General Hypothesis:", g_final, sep="\n")
```

Output:

```
The entered data is:

     Sky Temperature   Humid    Wind Water  Forest Output
0  Sunny        Warm  Normal  Strong  Warm    Same    Yes
1  Sunny        Warm    High  Strong  Warm    Same    Yes
2  Rainy        Cold    High  Strong  Warm  Change     No
3  Sunny        Warm    High  Strong  Cool  Change    Yes


The attributes are:
[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]

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

Fig 2. Input data for candidate elimination with attributes and target

```
Specific Hypothesis:
 ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

General Hypothesis:
 [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Positive Example
Step:  1
Specific Hypothesis:
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
General Hypothesis :
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Positive Example
Step:  2
Specific Hypothesis:
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
General Hypothesis :
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Negative Example
Step:  3
Specific Hypothesis:
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
General Hypothesis :
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]

Positive Example
Step:  4
Specific Hypothesis:
['Sunny' 'Warm' '?' 'Strong' '?' '?']
General Hypothesis :
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific Hypothesis:
['Sunny' 'Warm' '?' 'Strong' '?' '?']

Final General Hypothesis:
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
```

Fig 3. Final Specific and General Hypothesis from candidate elimination algorithm

PROGRAM – 3

Question:

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 to classify a new sample.

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
/kaggle/input/id3-algorithm/week3_data.csv

In [2]:
data = pd.read_csv("../input/id3-algorithm/week3_data.csv")
features = [feat for feat in data]
```

```
features.remove("answer")
class Node:
    def __init__(self):
        self.children = []
        self.value = ""
        self.isLeaf = False
        self.pred = ""
def entropy(examples):
    pos = 0.0
    neg = 0.0
    for _, row in examples.iterrows():
        if row["answer"] == "yes":
            pos += 1
        else:
            neg += 1
    if pos == 0.0 or neg == 0.0:
        return 0.0
    else:
        p = pos / (pos + neg)
        n = neg / (pos + neg)
        return -(p * math.log(p, 2) + n * math.log(n, 2))
def info_gain(examples, attr):
    uniq = np.unique(examples[attr])
    #print ("\n",uniq)
    gain = entropy(examples)
    #print ("\n",gain)
    for u in uniq:
        subdata = examples[examples[attr] == u]
        #print ("\n",subdata)
        sub_e = entropy(subdata)
        gain -= (float(len(subdata)) / float(len(examples))) * sub_e
        #print ("\n",gain)
    return gain
def ID3(examples, attrs):
    root = Node()
    max_gain = 0
    max_feat = ""
    for feature in attrs:
        #print ("\n",examples)
        gain = info_gain(examples, feature)
        if gain > max_gain:
            max_gain = gain
            max_feat = feature
    root.value = max_feat
    #print ("\nMax feature attr",max_feat)
    uniq = np.unique(examples[max_feat])
    #print ("\n",uniq)
    for u in uniq:
        #print ("\n",u)
        subdata = examples[examples[max_feat] == u]
        #print ("\n",subdata)
        if entropy(subdata) == 0.0:
```

```
            newNode = Node()
            newNode.isLeaf = True
            newNode.value = u
            newNode.pred = np.unique(subdata["answer"])
            root.children.append(newNode)
        else:
            dummyNode = Node()
            dummyNode.value = u
            new_attrs = attrs.copy()
            new_attrs.remove(max_feat)
            child = ID3(subdata, new_attrs)
            dummyNode.children.append(child)
            root.children.append(dummyNode)
    return root
def printTree(root: Node, depth=0):
    for i in range(depth):
        print("\t", end="")
    print(root.value, end="")
    if root.isLeaf:
        print(" -> ", root.pred)
    print()
    for child in root.children:
        printTree(child, depth + 1)

In [3]:
root = ID3(data, features)
printTree(root)
```

Output:

outlook	temperature	humidity	wind	answer
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no

Fig 4. Input dataset for decision tree ID3 algorithm

```
outlook
        overcast ->  ['yes']

        rain
                wind
                        strong ->  ['no']

                        weak ->  ['yes']

        sunny
                humidity
                        high ->  ['no']

                        normal ->  ['yes']
```

Fig 5. Final output Decision Tree

PROGRAM – 4

Question:

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.

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li
st all files under the input directory
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
ets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be save
d outside of the current session
```

```
/kaggle/input/naive-bayes-classifier/week4_data_play_tennis.csv
```

```
In [2]:
data = pd.read_csv('../input/naive-bayes-classifier/week4_data_play_tennis.cs
v')
data.head()
```

```
In [3]:
y = list(data['PlayTennis'].values)
X = data.iloc[:,1:].values
print(f'Target Values: {y}')
print(f'Features: \n{X}')
```

```
In [4]:
y_train = y[:8]
y_val = y[8:]
X_train = X[:8]
X_val = X[8:]
print(f"Number of instances in training set: {len(X_train)}")
print(f"Number of instances in testing set: {len(X_val)}")
```

```
Number of instances in training set: 8
Number of instances in testing set: 6
```

```
In [5]:
class NaiveBayesClassifier:
    def __init__(self, X, y):
        self.X, self.y = X, y
        self.N = len(self.X)
        self.dim = len(self.X[0])
        self.attrs = [[] for _ in range(self.dim)]
        self.output_dom = {}
        self.data = []
        for i in range(len(self.X)):
            for j in range(self.dim):
```

```
                if not self.X[i][j] in self.attrs[j]:
                    self.attrs[j].append(self.X[i][j])
            if not self.y[i] in self.output_dom.keys():
                self.output_dom[self.y[i]] = 1
            else:
                self.output_dom[self.y[i]] += 1
            self.data.append([self.X[i], self.y[i]])
    def classify(self, entry):
        solve = None
        max_arg = -1
        for y in self.output_dom.keys():
            prob = self.output_dom[y]/self.N
            for i in range(self.dim):
                cases = [x for x in self.data if x[0][i] == entry[i] and x[1]
== y]
                n = len(cases)
                prob *= n/self.N
            if prob > max_arg:
                max_arg = prob
                solve = y
        return solve
```

In [6]:

```
nbc = NaiveBayesClassifier(X_train, y_train)
total_cases = len(y_val)
good = 0
bad = 0
predictions = []
for i in range(total_cases):
    predict = nbc.classify(X_val[i])
    predictions.append(predict)
    if y_val[i] == predict:
        good += 1
    else:
        bad += 1
print('Predicted values:', predictions)
print('Actual values:', y_val)
print()
print('Total number of testing instances in the dataset:', total_cases)
print('Number of correct predictions:', good)
print('Number of wrong predictions:', bad)
print()
print('Accuracy of Bayes Classifier:', good/total_cases)
```

Output:

Out[2]:

	PlayTennis	Outlook	Temperature	Humidity	Wind
0	No	Sunny	Hot	High	Weak
1	No	Sunny	Hot	High	Strong
2	Yes	Overcast	Hot	High	Weak
3	Yes	Rain	Mild	High	Weak
4	Yes	Rain	Cool	Normal	Weak

Fig 6. First five records of input dataset for Naïve Bayes Classifier

	PlayTennis	Outlook	Temperature	Humidity	Wind
2	No	Sunny	Hot	High	Weak
3	No	Sunny	Hot	High	Strong
4	Yes	Overcast	Hot	High	Weak
5	Yes	Rain	Mild	High	Weak
6	Yes	Rain	Cool	Normal	Weak
7	No	Rain	Cool	Normal	Strong
8	Yes	Overcast	Cool	Normal	Strong
9	No	Sunny	Mild	High	Weak
10	Yes	Sunny	Cool	Normal	Weak
11	Yes	Rain	Mild	Normal	Weak
12	Yes	Sunny	Mild	Normal	Strong
13	Yes	Overcast	Mild	High	Strong
14	Yes	Overcast	Hot	Normal	Weak
15	No	Rain	Mild	High	Strong

Fig 7. Full Input dataset for Naïve Bayes Classifier

```
Target Values: ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
Features:
[['Sunny' 'Hot' 'High' 'Weak']
 ['Sunny' 'Hot' 'High' 'Strong']
 ['Overcast' 'Hot' 'High' 'Weak']
 ['Rain' 'Mild' 'High' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Strong']
 ['Overcast' 'Cool' 'Normal' 'Strong']
 ['Sunny' 'Mild' 'High' 'Weak']
 ['Sunny' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Mild' 'Normal' 'Weak']
 ['Sunny' 'Mild' 'Normal' 'Strong']
 ['Overcast' 'Mild' 'High' 'Strong']
 ['Overcast' 'Hot' 'Normal' 'Weak']
 ['Rain' 'Mild' 'High' 'Strong']]
```

Fig 8. Target values and features of data

```
Predicted values: ['No', 'Yes', 'No', 'Yes', 'Yes', 'No']
Actual values: ['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

Total number of testing instances in the dataset: 6
Number of correct predictions: 4
Number of wrong predictions: 2

Accuracy of Bayes Classifier: 0.6666666666666666
```

Fig 9. Final output of naïve bayes classifier with accuracy

PROGRAM – 5

Question:

Write a program to construct a Bayesian network considering training data. Use this model to make predictions.

Program:

```
In [1]:
!pip install pgmpy
```

```
In [2]:
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

# Input data files are available in the read-only "../input/" directory
```

```
# For example, running this (by clicking run or pressing Shift+Enter) will li
st all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
ets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be save
d outside of the current session
```

In [3]:
```
trainingData = pd.read_csv('/content/week5_data_heart.csv')
trainingData = trainingData.replace('?',np.nan)
print('The sample instances from the dataset are:')
print(trainingData.head())
print('\n Attributes and datatypes: ')
print(trainingData.dtypes)
```

In [4]:
```
model = BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang'
,'heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdise
ase','chol')])
print('\n Learning CPD using Maximum likelihood estimators')
model.fit(trainingData,estimator=MaximumLikelihoodEstimator)
print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)
```

```
 Learning CPD using Maximum likelihood estimators

 Inferencing with Bayesian Network:
```

In [5]:
```
print('\n 1.Probability of HeartDisease given evidence = restecg (Rest ECG):
1')
q1 = HeartDiseasetest_infer.query(variables = ['heartdisease'], evidence={'re
stecg':1})
print(q1)
```

In [6]:
```
print('\n 2.Probability of HeartDisease given evidence = chol (Cholestorol):
100 ')
```

```
q2 = HeartDiseasetest_infer.query(variables = ['heartdisease'], evidence={'ch
ol':100})
print(q2)
```

Output:

```
Requirement already satisfied: pgmpy in /usr/local/lib/python3.7/dist-packages (0.1.14)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from pgmpy) (0.22.2.post1)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packages (from pgmpy) (2.4.7)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from pgmpy) (1.19.5)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (from pgmpy) (0.10.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from pgmpy) (1.1.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from pgmpy) (1.4.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from pgmpy) (4.41.1)
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (from pgmpy) (1.8.1+cu101)
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages (from pgmpy) (2.5.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from pgmpy) (1.0.1)
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from statsmodels->pgmpy) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->pgmpy) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->pgmpy) (2018.9)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch->pgmpy) (3.7.4.3)
Requirement already satisfied: decorator<5,>=4.3 in /usr/local/lib/python3.7/dist-packages (from networkx->pgmpy) (4.4.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy>=0.4.0->statsmodels->pgmpy) (1.15.0)
```

Fig 10. Installation of pgmpy library

```
The sample instances from the dataset are:
   age  sex  cp  trestbps  chol  ...  oldpeak  slope  ca  thal  heartdisease
0   63    1   1       145   233  ...      2.3      3   0     6             0
1   67    1   4       160   286  ...      1.5      2   3     3             2
2   67    1   4       120   229  ...      2.6      2   2     7             1
3   37    1   3       130   250  ...      3.5      3   0     3             0
4   41    0   2       130   204  ...      1.4      1   0     3             0

[5 rows x 14 columns]

 Attributes and datatypes:
age             int64
sex             int64
cp              int64
trestbps        int64
chol            int64
fbs             int64
restecg         int64
thalach         int64
exang           int64
oldpeak       float64
slope           int64
ca             object
thal           object
heartdisease    int64
dtype: object
```

Fig 11. Sample instances from heart disease dataset with attributes and their datatypes

```
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 1001.27it/s]
Eliminating: sex: 100%|██████████| 5/5 [00:00<00:00, 79.63it/s]
 1.Probability of HeartDisease given evidence = restecg (Rest ECG): 1
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.1012 |
+-----------------+---------------------+
| heartdisease(1) |              0.0000 |
+-----------------+---------------------+
| heartdisease(2) |              0.2392 |
+-----------------+---------------------+
| heartdisease(3) |              0.2015 |
+-----------------+---------------------+
| heartdisease(4) |              0.4581 |
+-----------------+---------------------+
```

Fig 12. Probability of heart disease given rest ECG using Bayesian network

```
 2.Probability of HeartDisease given evidence = chol (Cholestorol): 100
/usr/local/lib/python3.7/dist-packages/pgmpy/factors/discrete/DiscreteFactor.py:519: UserWarning: Found unknown state name. Trying to swi
tch to using all state names as state numbers
  "Found unknown state name. Trying to switch to using all state names as state numbers"
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 832.10it/s]
Eliminating: sex: 100%|██████████| 5/5 [00:00<00:00, 62.73it/s]
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              1.0000 |
+-----------------+---------------------+
| heartdisease(1) |              0.0000 |
+-----------------+---------------------+
| heartdisease(2) |              0.0000 |
+-----------------+---------------------+
| heartdisease(3) |              0.0000 |
+-----------------+---------------------+
| heartdisease(4) |              0.0000 |
+-----------------+---------------------+
```

Fig 13. Probability of heart disease given cholesterol

PROGRAM – 6

Question:

Apply k-Means algorithm to cluster a set of data stored in a .CSV file.

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
docker-python
```

```
# For example, here's several helpful packages to load

import math;
import sys;
import pandas as pd
import numpy as np
from random import choice
from matplotlib import pyplot
from random import shuffle, uniform;

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li
st all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
ets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be save
d outside of the current session
```

```
/kaggle/input/k-means/iris.csv
```

```
In [2]:
def ReadData(fileName):
    f = open(fileName,'r')
    lines = f.read().splitlines()
    f.close()

    items = []

    for i in range(1,len(lines)):
        line = lines[i].split(',')
        itemFeatures = []

        for j in range(len(line)-1):
            v = float(line[j])
            itemFeatures.append(v)
        items.append(itemFeatures)

    shuffle(items)

    return items
```

```
def FindColMinMax(items):
    n = len(items[0])
    minima = [float('inf') for i in range(n)]
    maxima = [float('-inf') -1 for i in range(n)]

    for item in items:
        for f in range(len(item)):
            if(item[f] < minima[f]):
                minima[f] = item[f]

            if(item[f] > maxima[f]):
                maxima[f] = item[f]

    return minima,maxima

def EuclideanDistance(x,y):
    S = 0
    for i in range(len(x)):
        S += math.pow(x[i]-y[i],2)

    return math.sqrt(S)

def InitializeMeans(items,k,cMin,cMax):
    f = len(items[0])
    means = [[0 for i in range(f)] for j in range(k)]

    for mean in means:
        for i in range(len(mean)):
            mean[i] = uniform(cMin[i]+1,cMax[i]-1)

    return means

def UpdateMean(n,mean,item):
    for i in range(len(mean)):
        m = mean[i]
        m = (m*(n-1)+item[i])/float(n)
        mean[i] = round(m,3)

    return mean


def FindClusters(means,items):
    clusters = [[] for i in range(len(means))]

    for item in items:
        index = Classify(means,item)
        clusters[index].append(item)

    return clusters

def Classify(means,item):

    minimum = float('inf');
    index = -1
```

```
    for i in range(len(means)):
        dis = EuclideanDistance(item,means[i])

        if(dis < minimum):
            minimum = dis
            index = i

    return index

def CalculateMeans(k,items,maxIterations=100000):
    cMin, cMax = FindColMinMax(items)

    means = InitializeMeans(items,k,cMin,cMax)

    clusterSizes = [0 for i in range(len(means))]

    belongsTo = [0 for i in range(len(items))]

    for e in range(maxIterations):
        noChange = True;
        for i in range(len(items)):
            item = items[i];
            index = Classify(means,item)
            clusterSizes[index] += 1
            cSize = clusterSizes[index]
            means[index] = UpdateMean(cSize,means[index],item)

            if(index != belongsTo[i]):
                noChange = False
            belongsTo[i] = index

        if (noChange):
            break

    return means

def CutToTwoFeatures(items,indexA,indexB):
    n = len(items)
    X = []
    for i in range(n):
        item = items[i]
        newItem = [item[indexA],item[indexB]]
        X.append(newItem)

    return X

def PlotClusters(clusters):
    n = len(clusters)
    X = [[] for i in range(n)]

    for i in range(n):
        cluster = clusters[i]
        for item in cluster:
            X[i].append(item)
```

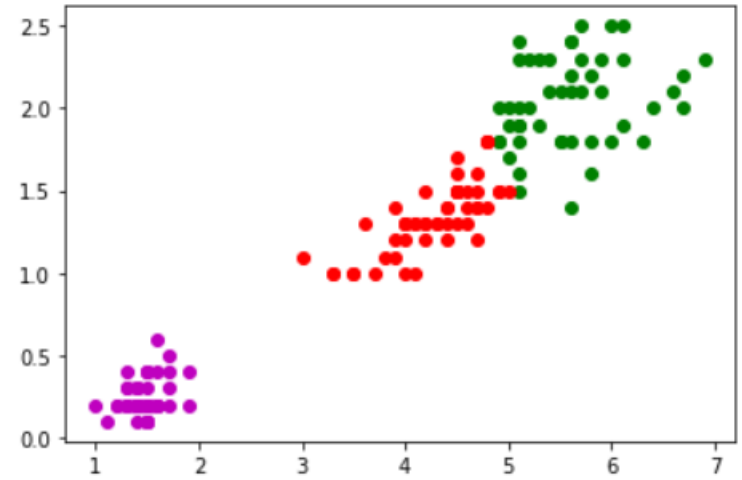
```
    colors = ['r','b','g','c','m','y']

    for x in X:
        c = choice(colors)
        colors.remove(c)
        Xa = []
        Xb = []

        for item in x:
            Xa.append(item[0])
            Xb.append(item[1])

        pyplot.plot(Xa,Xb,'o',color=c)

    pyplot.show()


In [3]:
def main():
    items = ReadData('../input/k-means/iris.csv')
    k = 3
    items = CutToTwoFeatures(items,2,3)
    print(items)
    means = CalculateMeans(k,items)
    print("\nMeans = ", means)

    clusters = FindClusters(means,items)

    PlotClusters(clusters)
    newItem = [1.5,0.2]
    print(Classify(means,newItem))

if __name__ == "__main__":
    main()
```

Output:

```
[[1.2, 0.2], [1.4, 0.2], [4.3, 1.3], [1.5, 0.4], [1.5, 0.3], [1.4, 0.2], [4.5, 1.5], [1.4, 0.3], [5.9, 2.1], [4.6, 1.5], [1.6, 0.6], [5.
4, 2.1], [5.1, 1.8], [3.5, 1.0], [1.9, 0.2], [4.0, 1.2], [6.1, 2.3], [3.9, 1.2], [5.2, 2.0], [5.0, 2.0], [4.4, 1.2], [1.6, 0.2], [4.7, 1.
4], [4.5, 1.6], [4.8, 1.8], [5.8, 1.6], [5.7, 2.3], [5.2, 2.3], [1.0, 0.2], [1.4, 0.3], [4.1, 1.0], [5.1, 1.6], [4.8, 1.4], [6.1, 2.5],
[6.7, 2.2], [6.4, 2.0], [1.4, 0.2], [4.2, 1.3], [5.6, 1.8], [1.4, 0.2], [4.8, 1.8], [3.0, 1.1], [1.4, 0.2], [5.6, 2.4], [1.5, 0.1], [5.6,
2.1], [3.9, 1.4], [4.2, 1.3], [4.8, 1.8], [5.6, 2.2], [4.5, 1.7], [4.9, 1.8], [4.7, 1.2], [5.5, 2.1], [1.3, 0.4], [5.5, 1.8], [3.3, 1.0],
[4.5, 1.5], [1.4, 0.2], [5.1, 2.3], [5.6, 1.4], [1.5, 0.2], [4.5, 1.3], [5.1, 2.4], [6.3, 1.8], [1.3, 0.2], [1.5, 0.4], [1.4, 0.2], [6.7,
2.0], [5.0, 1.5], [4.9, 1.8], [1.3, 0.2], [3.8, 1.1], [1.5, 0.2], [4.5, 1.5], [4.0, 1.3], [1.7, 0.4], [3.5, 1.0], [4.6, 1.3], [3.6, 1.3],
[6.6, 2.1], [3.9, 1.1], [5.1, 2.0], [5.5, 1.8], [4.3, 1.3], [4.2, 1.2], [4.5, 1.5], [1.3, 0.2], [1.5, 0.4], [1.5, 0.2], [6.0, 1.8], [1.2,
0.2], [4.7, 1.5], [4.7, 1.4], [4.4, 1.3], [5.1, 1.9], [5.9, 2.3], [5.4, 2.3], [1.1, 0.1], [4.6, 1.4], [1.4, 0.1], [5.0, 1.9], [1.5, 0.2],
[4.0, 1.3], [5.1, 1.9], [4.0, 1.0], [5.8, 1.8], [3.7, 1.0], [4.1, 1.3], [5.0, 1.7], [1.7, 0.3], [4.9, 2.0], [1.6, 0.2], [1.7, 0.5], [1.6,
0.2], [4.4, 1.4], [1.3, 0.3], [1.5, 0.1], [1.5, 0.2], [1.6, 0.2], [1.3, 0.2], [5.6, 2.4], [1.9, 0.4], [5.8, 2.2], [1.7, 0.2], [6.9, 2.3],
[5.1, 1.5], [1.6, 0.4], [4.0, 1.3], [5.3, 1.9], [6.0, 2.5], [6.1, 1.9], [4.2, 1.5], [4.5, 1.5], [4.9, 1.5], [5.7, 2.1], [1.5, 0.1], [3.3,
1.0], [1.6, 0.2], [4.9, 1.5], [1.3, 0.3], [1.4, 0.3], [5.7, 2.5], [1.5, 0.2], [1.5, 0.1], [4.7, 1.6], [5.3, 2.3], [4.4, 1.4], [4.1, 1.3]]
```

Fig 14. Items in the dataset iris.csv

```
Means =  [[1.462, 0.254], [5.589, 2.038], [4.266, 1.345]]
```




```
0
```

Fig 15. Output graph showing k = 3 clusters with means

PROGRAM – 7

Question:

Apply EM algorithm to cluster a set of data stored in a .CSV file. Compare the results of k-means algorithm and EM algorithm.

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
docker-python
# For example, here's several helpful packages to load

from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import sklearn.metrics as metrics
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li
st all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
ets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be save
d outside of the current session
```

```
/kaggle/input/emalgorithm/dataset.csv
```

In [2]:

```
names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width', 'Class']

dataset = pd.read_csv("../input/emalgorithm/dataset.csv", names=names)

X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

y = [label[c] for c in dataset.iloc[:, -1]]

plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])



plt.subplot(1,3,1)
plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])
model=KMeans(n_clusters=3, random_state=0).fit(X)
plt.subplot(1,3,2)
plt.title('KMeans')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[model.labels_])

print('The accuracy score of K-Mean: ',metrics.accuracy_score(y, model.lab
els_))
print('The Confusion matrixof K-Mean:\n',metrics.confusion_matrix(y, model
.labels_))


gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
plt.subplot(1,3,3)
```

```
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])

print('The accuracy score of EM: ',metrics.accuracy_score(y, y_cluster_gmm
))
print('The Confusion matrix of EM:\n ',metrics.confusion_matrix(y, y_clust
er_gmm))
```

Output:

```
The accuracy score of K-Mean:  0.24
The Confusion matrixof K-Mean:
 [[ 0 50  0]
 [48  0  2]
 [14  0 36]]
The accuracy score of EM:  0.36666666666666664
The Confusion matrix of EM:
  [[50  0  0]
 [ 0  5 45]
 [ 0 50  0]]
```

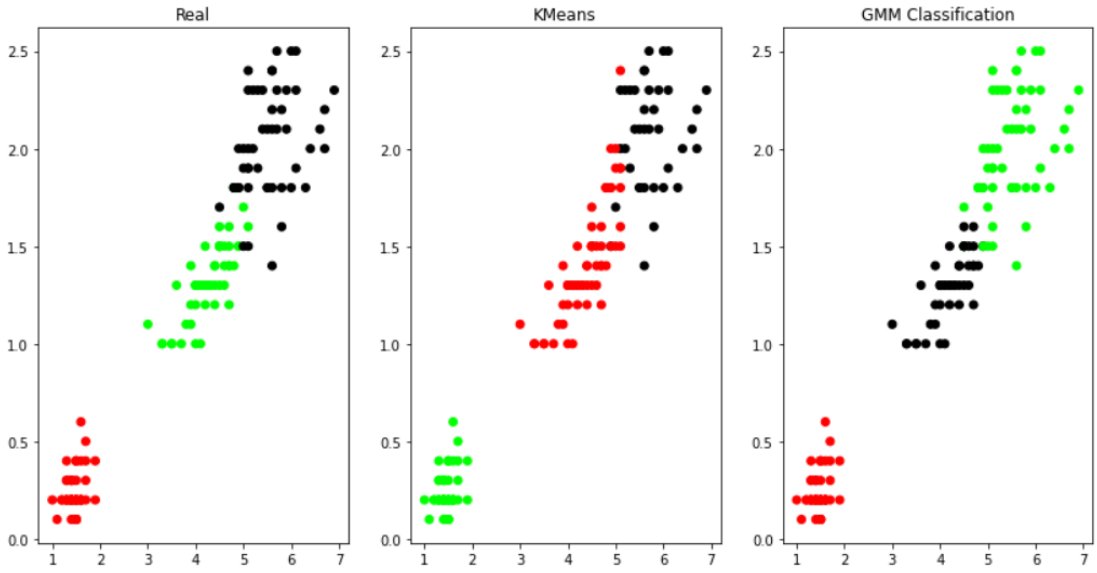


Fig 16. Output showing accuracy comparison of k-means and EM algorithms on the given data and the graphical representations of the algorithm clusters along with the confusion matrices

1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa

Fig 17. Few rows of the iris dataset used

PROGRAM – 8

Question:

Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
docker-python
# For example, here's several helpful packages to load

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import datasets

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li
st all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
ets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be save
d outside of the current session

In [2]:
iris = datasets.load_iris()
X = iris.data
```

```
Y = iris.target

print('sepal-length','sepal-width','petal-length','petal-width')
print(X)
print('target')
print(Y)
```

```
In [3]:
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, ran
dom_state=42)

#Training the model with Nearest nighbors K=3
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
```

Out[3]:

```
KNeighborsClassifier(n_neighbors=3)
```

```
In [4]:
from sklearn.metrics import accuracy_score

y_pred=knn.predict(X_test)
matrix =confusion_matrix(y_test,y_pred)
print(" Confusion matrix:\n",matrix)
print(" Correct predicition",accuracy_score(y_test,y_pred))
print(" Wrong predicition",(1-accuracy_score(y_test,y_pred)))
print(' Accuracy Metrics')
print(classification_report(y_test,y_pred))
```

Output:

```
target
[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]
```

Fig 18. Target values of iris dataset

```
sepal-length sepal-width petal-length petal-width
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
```

Fig 19. Iris dataset values

```
Confusion matrix:
[[19  0  0]
 [ 0 15  0]
 [ 0  1 15]]
Correct predicition 0.98
Wrong predicition 0.020000000000000018
Accuracy Metrics
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.94      1.00      0.97        15
           2       1.00      0.94      0.97        16

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50
```

Fig 20. Output of KNN classifier with correct and wrong predictions and accuracy metrics

PROGRAM – 9

Question:

Implement the Linear Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Program:

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

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

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/linear-reg/salary.csv
```

In [2]:

```
dataset = pd.read_csv('../input/linear-reg/salary.csv')
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, 1].values
```

In [3]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, rand
om_state=0)
```

In [4]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[4]:

```
LinearRegression()
```

In [5]:

```
y_pred = regressor.predict(X_test)
```

In [6]:

```
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()
```

In [7]:

```
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
```

Output:

	YearsExperience	Salary
2	1.1	39343
3	1.3	46205
4	1.5	37731
5	2.0	43525
6	2.2	39891

Fig 21. Sample rows of salary dataset

Fig 22. Output graph showing salary vs experience years for training set of data

Fig 23. Output graph showing salary vs experience for test data

PROGRAM – 10

Question:

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

Program:

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
docker-python
# For example, here's several helpful packages to load

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

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li
st all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
ets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be save
d outside of the current session
/kaggle/input/local-weight/tips_data.csv

In [2]:
def kernel(point,xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m))) # eye - identity matrix
```

```
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
    return weights

def localWeight(point,xmat,ymat,k):
    wei = kernel(point,xmat,k)
    W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
    return W

def localWeightRegression(xmat,ymat,k):
    m,n = np.shape(xmat)
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred


def graphPlot(X,ypred):
    sortindex = X[:,1].argsort(0) #argsort - index of the smallest
    xsort = X[sortindex][:,0]
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.scatter(bill,tip, color='green')
    ax.plot(xsort[:,1],ypred[sortindex], color = 'red', linewidth=5)
    plt.xlabel('Total bill')
    plt.ylabel('Tip')
    plt.show();

In [3]:
data = pd.read_csv('../input/local-weight/tips_data.csv')
bill = np.array(data.total_bill) # We use only Bill amount and Tips data
tip = np.array(data.tip)

In [4]:
mbill = np.mat(bill) # .mat will convert nd array is converted in 2D array
mtip = np.mat(tip)
m= np.shape(mbill)[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T))

In [5]:
ypred = localWeightRegression(X,mtip,3)
graphPlot(X,ypred)
```

Output:

	total_bill	tip	sex	smoker	day	time	size
2	16.99	1.01	Female	No	Sun	Dinner	2
3	10.34	1.66	Male	No	Sun	Dinner	3
4	21.01	3.5	Male	No	Sun	Dinner	3
5	23.68	3.31	Male	No	Sun	Dinner	2
6	24.59	3.61	Female	No	Sun	Dinner	4
7	25.29	4.71	Male	No	Sun	Dinner	4
8	8.77	2.0	Male	No	Sun	Dinner	2
9	26.88	3.12	Male	No	Sun	Dinner	4
10	15.04	1.96	Male	No	Sun	Dinner	2
11	14.78	3.23	Male	No	Sun	Dinner	2
12	10.27	1.71	Male	No	Sun	Dinner	2
13	35.26	5.0	Female	No	Sun	Dinner	4
14	15.42	1.57	Male	No	Sun	Dinner	2
15	18.43	3.0	Male	No	Sun	Dinner	4

Fig 24. Sample rows of the tips dataset

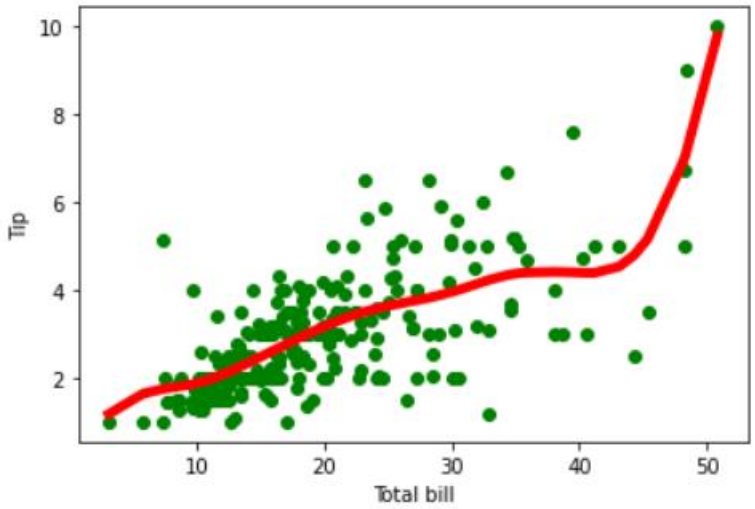


Fig 25. Output graph of the locally weighted regression algorithm showing tips vs total bill