PROGRAM 5

5. The Following Training Examples Map Descriptions Of Individuals Onto High, Medium And LowCredit-Worthiness.

medium skiing design single twenties no ->highRisk

high golf trading married forties yes ->lowRisk

low speedway transport married thirties yes ->medRisk

medium football banking single thirties yes ->lowRisk

high flying media married fifties yes ->highRisk

low football security single twenties no ->medRisk

medium golf media single thirties yes ->medRisk

medium golf transport married forties yes ->lowRisk

high skiing banking single thirties yes ->highRisk low

golf unemployed married forties yes ->highRisk

SOURCE CODE:

Input attributes are (from left to right) income, recreation, job, status, age-group, home-owner.Findtheunconditionalprobabilityof`golf'andtheconditionalprobabilityofsingle' given `medRisk' in thedataset?

```
totalRecords=10
numberGolfRecreation=4
probGolf=numberGolfRecreation/totalRecordsprint("Unconditio
nal probability of golf: ={}".format(probGolf)) #conditional
probability of `single' given`medRisk'
# bayes Formula
#p(single|medRisk)=p(medRisk|single)p(single)/p(medRisk)
#p(medRisk|single)=p(medRisk ∩single)/p(single)
```

```
# Therefore the result is:
numberMedRiskSingle=2
numberMedRisk=3
probMedRiskSingle=numberMedRiskSingle/totalRecordspr
obMedRisk=numberMedRisk/totalRecordsconditionalProba
bility=(probMedRiskSingle/probMedRisk)
print("Conditional probability of single given medRisk: = {}".format(conditionalProbability))
```

OUTPUT:

```
Unconditional probability of golf: = 0.4
Conditional probability of single given medRisk: = 0.6666666666666667
```

EXPERIMENT: DATE:

PROGRAM 6:

IMPLEMENT LINEAR REGRESSION USINGPYTHON.

AIM:To Implement linear regression using python.

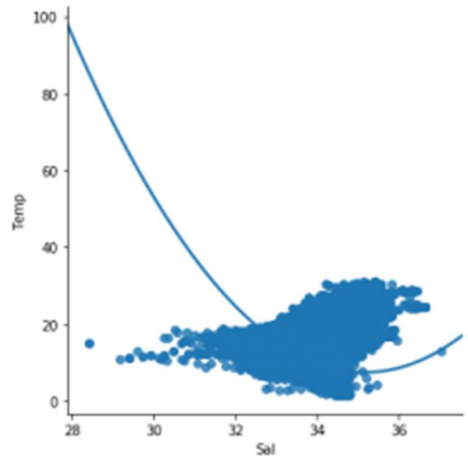
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
df = pd.read_csv('bottle.csv')
df_binary = df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()
```

	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45

```
#plotting the Scatter plot to check relationship between Sal and Temp
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order = 2, ci = None)
plt.show()
```

EXPERIMENT: DATE:




```
X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)

# Separating the data into independent and dependent variables
# Converting each dataframe into a numpy array
# since each dataframe contains only one column
df_binary.dropna(inplace = True)

# Dropping any rows with Nan values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

# Splitting the data into training and testing data
regr = LinearRegression()

regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

OUTPUT:

```
0.20780376990868232
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error

mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
#squared True returns MSE value, False returns RMSE value.
mse = mean_squared_error(y_true=y_test,y_pred=y_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,squared=False)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

EXPERIMENT: DATE:

OUTPUT:

```
MAE: 0.7927322046360309

MSE: 1.0251137190180517

RMSE: 1.0124789968281078
```

Roll no:

PROGRAM 7

IMPLEMENT NAÏVE BAYES THEOREM TO CLASSIFY THE ENGLISHTEXT

AIM:

To Implement naïve baye's theorem to classify the English text

SOURCE CODE:

```
import pandas as pd
fromsklearn.model_selection import train_test_split
fromsklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
fromsklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score

msg = pd.read_csv('document.csv', names=['message', 'label'])
print("Total Instances of Dataset: ", msg.shape[0])
msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})

X = msg.message
y = msg.labelnum

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)

count_v = CountVectorizer()
```

```
Xtrain_dm = count_v.fit_transform(Xtrain)

Xtest_dm = count_v.transform(Xtest)

df = pd.DataFrame(Xtrain_dm.toarray(),columns=count_v.get_feature_names())

clf = MultinomialNB()

clf.fit(Xtrain_dm, ytrain)

pred = clf.predict(Xtest_dm)

print('Accuracy Metrics:')

print('Accuracy: ', accuracy_score(ytest, pred)) print('Recall: ', recall_score(ytest, pred)) print('Precision: ',

precision_score(ytest, pred))

print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

document.csv:

```
I love this sandwich, pos

Thisis an amazing place,pos

I feel very good about these beers,pos

This is my best work,pos

What an awesome view,pos

I do not like this restaurant,neg

I am tired of this stuff,neg

I can't deal with this,neg He

is my sworn enemy, neg
```

My boss is horrible , neg

This is an awesome place,pos

I do not like the taste of this juice,neg

I love to dance,pos

I am sick and tired of this place,neg

What a great holiday,pos

That is a bad locality to stay,neg
We will have good fun tomorrow,pos I

went to my enemy's house today,neg

OUTPUT:

Total Instances of Dataset: 18

Accuracy Metrics:

Accuracy: 0.6

Recall: 0.6666666666666666

Precision: 0.6666666666666666

Confusion Matrix:

[[1 1]

[1 2]]

Experiment: **Date:**

PROGRAM 9

9. IMPLEMENT THE FINITE WORDS CLASSIFICATION SYSTEM USING BACK-PROPAGATIONALGORITHM

AIM:

To implement thefinite words classification system using Back-propagationalgorithm

SOURCE CODE:

```
import pandas as pd
from sklearn.model_selection
import train_test_split
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.neural_network import
MLPClassifierfromsklearn.metrics
import accuracy_score, confusion_matrix,
precision_score, recall_score

msg = pd.read_csv('document.csv',
names=['message', 'label']) print("Total Instances of
Dataset: ", msg.shape[0]) msg['labelnum'] =
msg.label.map({'pos': 1, 'neg': 0})
X =msg.message
```

```
y=msg.labelnum
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)

count_v = CountVectorizer()
Xtrain_dm =
count_v.fit_transform(Xtrain)
Xtest_dm =
count_v.transform(Xtest)

df = pd.DataFrame(Xtrain_dm.toarray(),columns=count_v.get_feature_names())

clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1)

clf.fit(Xtrain_dm, ytrain) pred = clf.predict(Xtest_dm)
print('Accuracy Metrics:')
   print('Accuracy: ', accuracy_score(ytest, pred)) print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

document.csv:

```
I love this
sandwich , pos
Thisis an
amazingplace ,pos
```

I feel very good about these

beers,pos This is my best

work,pos

What an awesome view,pos

I do not like this

restaurant,neg

I am

tired of this stuff,neg

I can't deal with

this,neg

He is

my sworn

enemy,neg

Myboss is

horrible,neg

This is an awesome place,pos

I do not like the taste of

this juice,neg

I love todance,pos

I am sick and tired of this

place,neg

What a great holiday,pos

That is a bad locality to stay,neg
We will have good fun

tomorrow,pos I went to my

enemy's house today,neg

OUTPUT:

Total Instances of

Dataset: 18

Accuracy Metrics:

Accuracy: 0.8

Recall: 1.0

Precision:

0.75

Confusi

on

Matrix:

[[1 1]

[0 3]