PANIMALAR INSTITUTE OF TECHNOLOGY

JAISAKTHI EDUCATIONAL TRUST
(Affiliated to Anna University, Chennai)
Bangalore Trunk Road, Varadharajapuram
Poonamallee, Chennai – 600 123

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

AD8511 - MACHINE LEARNING LABORATORY

V SEMESTER - III YEAR

LAB MANUAL

ACADEMIC YEAR 2022-23

TABLE OF CONTENTS

Ex. No.	CONTENTS	PAGE No.
-	VISION AND MISSION OF THE INSTITUTE	iii
-	VISION AND MISSION OF THE DEPARMENT	iv
-	PROGRAM EDUCATIONAL OBJECTIVES	v
-	PROGRAM OUTCOMES	v
-	PROGRAM SPECIFIC OUTCOMES	vii
-	SYLLABUS	viii
LIST OF THE EXPERIMENTS		
1.	Implement the concept of decision trees with suitable data set from real world problem and classify the data set to produce new sample.	
2.	Detecting Spam mails using Support vector machine	
3.	Implement facial recognition application with artificial neural network	
4.	Study and implement amazon toolkit: Sagemaker	
5.	Implement character recognition using Multilayer Perceptron	
6.	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.	
7.	Implement sentiment analysis using random forest optimization algorithm	
8.	Write a program to construct a Bayesian 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.	
9.	Choose best machine learning algorithm to implement online fraud detection	
10.	Mini-project: students work in team on any socially relevant problem that needs a machine learning based solution, and evaluate the model performance	

VISION AND MISSION OF THE INSTITUTE

VISION

An Institution of Excellence by imparting quality education and serve as a perennial source of technical manpower with dynamic professionalism and entrepreneurship having social responsibility for the progress of the society and nation

MISSION

Panimalar Institute of Technology will strive to emerge as an Institution of Excellence in the country by

- Providing state-of-the-art infrastructure facilities for designing and developing solutions for engineering problems.
- Imparting quality education and training through qualified, experienced and committed members of the faculty**.**
- Inculcating high moral values in the minds of the Students and transforming them into a well-rounded personality**.**
- Establishing Industry Institute interaction to make students ready for the industrial environment.
- Promoting research based projects/activities in the emerging areas of Engineering & Technology.

VISION AND MISSION OF THE DEPARTMENT

VISION

To establish a unique standard of quality education by enriching the problem solving skills that adapt swiftly to the challenges of the society and industry. Producing professionals who shall be the leaders in technology employing Artificial Intelligence and Data Science along with core Computer Science.

MISSION

- To create an academic environment for higher learning, academic practices and research endeavours.
- To educate the students with latest technologies to update their knowledge in the field of AI and Data science.
- To empower students with knowledge through state-of-art infrastructure and curriculum.
- To produce successful professionals to serve the needs of Industry and society.
- To produce entrepreneurs in Artificial Intelligence and Data Science through excellence in education and research.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: To provide graduates with the proficiency to utilize the fundamental knowledge of basic sciences, mathematics, Artificial Intelligence, data science and statistics to build systems that require management and analysis of large volume of data.

PEO2: To enrich graduates with necessary technical skills to pursue pioneering research in the field of AI and Data Science and create disruptive and sustainable solutions for the welfare of ecosystems.

PEO3: To enable graduates to think logically, pursue lifelong learning and collaborate with an ethical attitude in a multidisciplinary team.

PEO4: To enable the graduates to design and model AI based solutions to critical problem domains in the real world.

PEO5: To enrich the innovative thoughts and creative ideas of the graduates for effective contribution towards economy building.

PROGRAM OUTCOMES OF THE DEPARTMENT

Engineering Graduates will be able to:

PO1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct Investigations of Complex Problems: Use research-based knowledge and research methods, including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling of complex engineering activities with an understanding of the limitations.

PO6. The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES OF THE DEPARTMENT

PSO1: Graduates should be able to evolve AI based efficient domain specific processes for effective decision making in several domains such as business and governance domains.

PSO2: Graduates should be able to arrive at actionable Fore sight, Insight , hind sight from data for solving business and engineering problems.

PSO3: Graduates should be able to create, select and apply the theoretical knowledge of AI and Data Analytics along with practical industrial tools and techniques to manage and solve wicked societal problems.

PSO4: Graduates should be capable of developing data analytics and data visualization skills, skills pertaining to knowledge acquisition, knowledge representation and knowledge engineering, and hence capable of coordinating complex projects

PSO5: Graduates should be able to carry out fundamental research to cater the critical needs of the society through cutting edge technologies of AI.

AD8511 MACHINE LEARNING LABORATORY L T P C
0 0 4 2

OBJECTIVES:

- To get practical knowledge on implementing machine learning algorithms in real time problem for getting solutions
- To implement supervised learning and their applications
- To understand unsupervised learning like clustering and EM algorithms
- To understand the theoretical and practical aspects of probabilistic graphical models.

LIST OF EXPERIMENTS :

1. Implement the concept of decision trees with suitable data set from real world problem and classify the data set to produce new sample.
2. Detecting Spam mails using Support vector machine
3. Implement facial recognition application with artificial neural network
4. Study and implement amazon toolkit: Sagemaker
5. Implement character recognition using Multilayer Perceptron
6. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.
7. Implement sentiment analysis using random forest optimization algorithm
8. Write a program to construct a Bayesian 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.
9. Choose best machine learning algorithm to implement online fraud detection
10. Mini-project: students work in team on any socially relevant problem that needs a machine learning based solution, and evaluate the model performance.

OUTCOMES:

Upon completion of the course, the students should be able to:

- Understand the implementation procedures for the machine learning algorithms.
- Design Java/Python programs for various Learning algorithms.
- Apply appropriate Machine Learning algorithms to data sets
- Identify and apply Machine Learning algorithms to solve real world problems.

REFERENCES

1. Sebastain Raschka, “Python Machine Learning”, Packt publishing (open source).
2. Ethem Alpaydin, “Introduction to Machine Learning”, MIT Press, Fourth Edition, 2020.
3. Stuart Russel and Peter Norvig, “Artificial Intelligence: A Modern Approach”, Fourth Edition, Pearson Education, 2020.

SOFTWARE:

- Python/Java with ML packages

TABLE OF CONTENTS

EX. NO.	NAME OF THE EXPERIMENT	PAGE NO.
1.	Implement the concept of decision trees with suitable data set from real world problem and classify the data set to produce new sample.	1
2.	Detecting Spam mails using Support vector machine	4
3.	Implement facial recognition application with artificial neural network	6
4.	Study and implement amazon toolkit: Sagemaker	9
5.	Implement character recognition using Multilayer Perceptron	12
6.	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.	14
7.	Implement sentiment analysis using random forest optimization algorithm	16
8.	Write a program to construct a Bayesian 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.	19
9.	Choose best machine learning algorithm to implement online fraud detection	21
10.	Mini-project: students work in team on any socially relevant problem that needs a machine learning based solution, and evaluate the model performance	24

EX.NO:1 IMPLEMENTATION OF DECISION TREES WITH SUITABLE DATA SET FROM REAL WORLD PROBLEM AND CLASSIFYING THE DATA SET TO PRODUCE NEW SAMPLE

AIM:

To implement decision trees with suitable data set from real world problem and to classify the data set to produce new sample.

ALGORITHM:

Step 1. Create the dataset
Step 2. Create empty data frame
Step3. Convert categorical variable into dummy/indicator variables or (binary variables) essentially 1's and 0's.
Step 4. Perform Decision tree classification
Step 5. Train the Decision Tree
Step 6. Print a decision tree in DOT format.
Step 7. Creation of Dot Data
Step 8. Creation of Graph from DOT data
Step 9. Show the graph

PROGRAM:

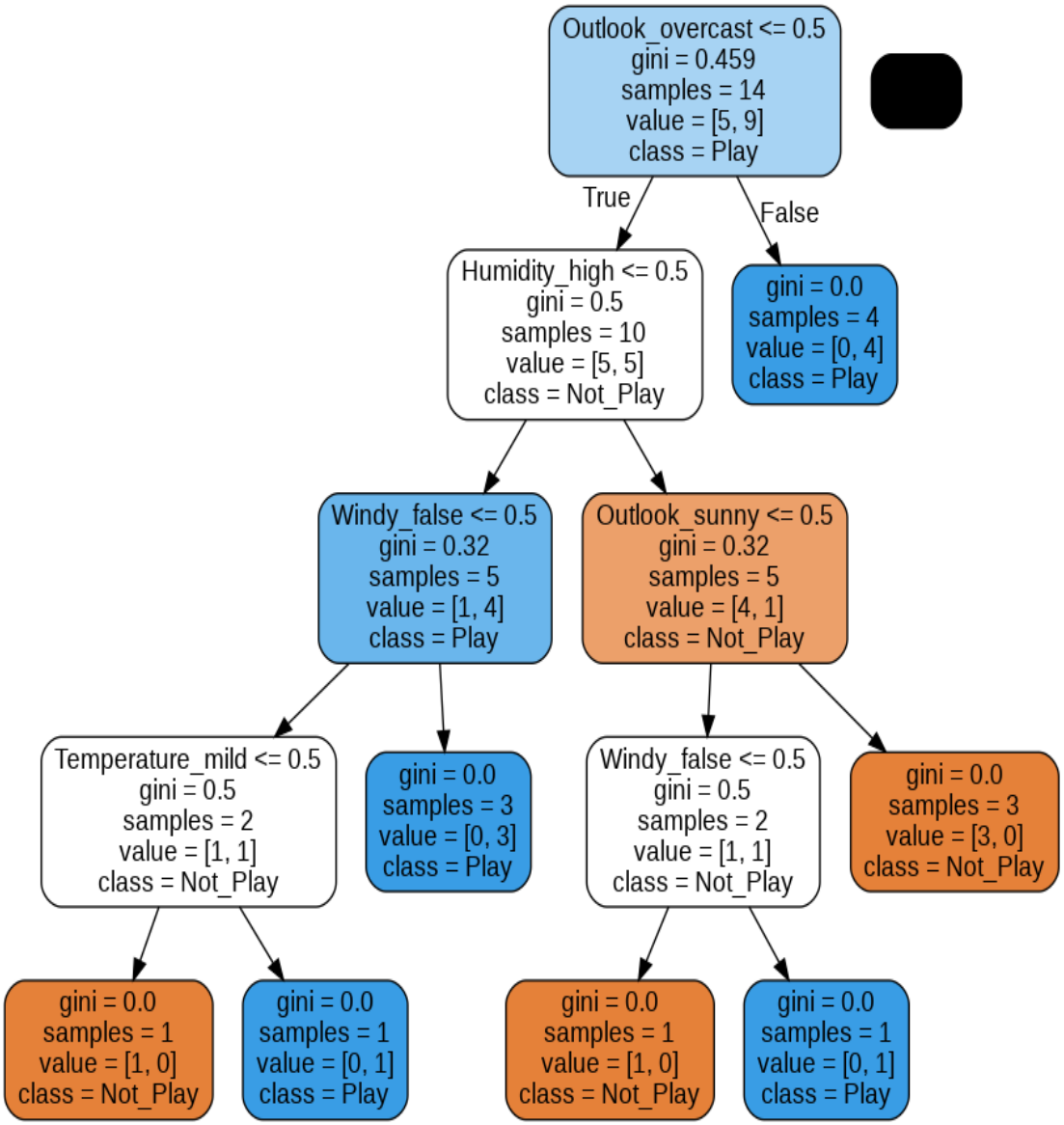
```
from sklearn import tree #For our Decision Tree
import pandas as pd # For our DataFrame
import pydotplus # To create our Decision Tree Graph
from IPython.display import Image  # To Display a image of our graph
#Create the dataset
#create empty data frame
golf_df = pd.DataFrame()
#add outlook
golf_df['Outlook'] = ['sunny', 'sunny', 'overcast', 'rainy', 'rainy', 'rainy',  'overcast', 'sunny',
 'sunny', 'rainy', 'sunny', 'overcast',  'overcast', 'rainy']
#add temperature
golf_df['Temperature'] = ['hot', 'hot', 'hot', 'mild', 'cool', 'cool', 'cool', 'mild',  'cool',  'mild',
 'mild', 'mild', 'hot', 'mild']
#add humidity
golf_df['Humidity'] = ['high', 'high', 'high', 'high', 'normal', 'normal', 'normal', 'high', 'normal'
,
  'normal', 'normal', 'high', 'normal', 'high']
#add windy
golf_df['Windy'] = ['false', 'true', 'false', 'false', 'false', 'true', 'true',    'false',     'false',     'false',
 'true',
 'true',  'false', 'true']
#finally add play
golf_df['Play'] = ['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes', 'yes',  'yes', 'yes', 'no']
#Print/show the new data
print(golf_df)
# Convert categorical variable into dummy/indicator variables or (binary vairbles) essential
y 1's and 0's
# I chose the variable name one_hot_data because in ML one-hot is a group of bits
among which the legal combinations of values are only those with a single high (1)
bit and all the others low (0)
```

```
one_hot_data = pd.get_dummies(golf_df[ ['Outlook', 'Temperature', 'Humidity', 'Windy'] ])
#print the new dummy data
one_hot_data
# The decision tree classifier.
clf = tree.DecisionTreeClassifier()
# Training the Decision Tree
clf_train = clf.fit(one_hot_data, golf_df['Play'])
# Export/Print a decision tree in DOT format.
print(tree.export_graphviz(clf_train, None))
#Create Dot Data
dot_data = tree.export_graphviz(clf_train, out_file=None, feature_names=list(one_hot_data
.
columns.values), class_names=['Not_Play', 'Play'], rounded=True, filled=True) #Gini decid
es
which attribute/feature should be placed at the root node, which features will act as internal
 nodes or leaf nodes
#Create Graph from DOT data
graph = pydotplus.graph_from_dot_data(dot_data)
# Show graph
Image(graph.create_png())
```

OUTPUT:

```
  Outlook Temperature Humidity  Windy Play
0      sunny         hot     high  false   no
1      sunny         hot     high   true   no
2   overcast         hot     high  false  yes
3      rainy        mild     high  false  yes
4      rainy        cool   normal  false  yes
5      rainy        cool   normal   true   no
6   overcast        cool   normal   true  yes
7      sunny        mild     high  false   no
8      sunny        cool   normal  false  yes
9      rainy        mild   normal  false  yes
10     sunny        mild   normal   true  yes
11  overcast        mild     high   true  yes
12  overcast         hot   normal  false  yes
13     rainy        mild     high   true   no
digraph Tree {
node [shape=box, fontname="helvetica"] ;
edge [fontname="helvetica"] ;
0 [label="X[0] <= 0.5\ngini = 0.459\nsamples = 14\nvalue = [5, 9]"] ;
1 [label="X[6] <= 0.5\ngini = 0.5\nsamples = 10\nvalue = [5, 5]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[8] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [1, 4]"] ;
1 -> 2 ;
3 [label="X[5] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]"] ;
2 -> 3 ;
4 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
3 -> 4 ;
5 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
3 -> 5 ;
6 [label="gini = 0.0\nsamples = 3\nvalue = [0, 3]"] ;
2 -> 6 ;
7 [label="X[2] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [4, 1]"] ;
1 -> 7 ;
8 [label="X[8] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]"] ;
```

```
7 -> 8 ;
9 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
8 -> 9 ;
10 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
8 -> 10 ;
11 [label="gini = 0.0\nsamples = 3\nvalue = [3, 0]"] ;
7 -> 11 ;
12 [label="gini = 0.0\nsamples = 4\nvalue = [0, 4]"] ;
0 -> 12 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```



RESULT:

Thus the decision trees and classifying the data set to produce new sample has been implemented an executed successfully.

EX.NO:2 DETECTING SPAM MAILS USING SVM

AIM:

To detect Spam mails using Support Vector Machine

ALGORITHM:

Step 1. Import Important Libraries

Step 2. Load the required Dataset

Step 3. Checking the information of the dataset

Step 4. Splitting our required data into X and Y

Step 5. Splitting our required data into training and testing

Step 6. Converting text into integer using CountVectorizer ()

Step 7. Applying SVM algorithm

Step 8. Checking the Accuracy

PROGRAM:

```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import GridSearchCV
from sklearn import svm
# Loading Dataset
data = pd.read_csv(Specific location Ex:
'https://raw.githubusercontent.com/Shreyakkk/Email-Spam-Detector/master/spam.csv')
# Checking the information of the dataset
data.info()
# Checking the information of the dataset
X = data['EmailText'].values
y = data['Label'].values
# Splitting our data into training and testing
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)
# Converting text into integer using CountVectorizer()
```

```
# Converting String to Integer
cv = CountVectorizer()
X_train = cv.fit_transform(X_train)
X_test = cv.transform(X_test)
# SVM algorithm
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
# Accuracy
print(classifier.score(X_test,y_test))
```

OUTPUT:

```
0.9766816143497757
```

RESULT:

Thus the Spam mails using Support Vector Machine was executed successfully.

EX.NO:3 IMPLEMENTATION OF FACIAL RECOGINITION APPLICATION WITH ARTIFICIAL NEURAL NETWORK

AIM:

To implement facial recognition application with artificial neural network.

ALGORITHM:

Step1. Load the required data.
Step 2. Split into training and testing set.
Step3. Compute a PCA
Step 4. Apply PCA Transformation
Step 5. Train a Neural Network
Step 6. Visualize

PROGRAM

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_lfw_people
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
# Load data
lfw_dataset = fetch_lfw_people(min_faces_per_person=100)
_, h, w = lfw_dataset.images.shape
X = lfw_dataset.data
y = lfw_dataset.target
target_names = lfw_dataset.target_names
# split into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
# Compute a PCA
n_components = 100
pca = PCA(n_components=n_components, whiten=True).fit(X_train)
# apply PCA transformation
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
# train a neural network
print("Fitting the classifier to the training set")
clf = MLPClassifier(hidden_layer_sizes=(1024,), batch_size=256, verbose=True, early_sto
pping=True).fit(X_train_pca, y_train)
y_pred = clf.predict(X_test_pca)
```

```
print(classification_report(y_test, y_pred, target_names=target_names))
# Visualization
def plot_gallery(images, titles, h, w, rows=3, cols=4):
    plt.figure()
    for i in range(rows * cols):
        plt.subplot(rows, cols, i + 1)
        plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)
        plt.title(titles[i])
        plt.xticks(())
        plt.yticks(())
 def titles(y_pred, y_test, target_names):
    for i in range(y_pred.shape[0]):
        pred_name = target_names[y_pred[i]].split(' ')[-1]
        true_name = target_names[y_test[i]].split(' ')[-1]
        yield 'predicted: {0}\ntrue: {1}'.format(pred_name, true_name)
 prediction_titles = list(titles(y_pred, y_test, target_names))
plot_gallery(X_test, prediction_titles, h, w)
plt.show()
```

OUTPUT:

```
Fitting the classifier to the training set
Iteration 1, loss = 1.50973308
Validation score: 0.387500
Iteration 2, loss = 1.11203960
Validation score: 0.475000
Iteration 3, loss = 0.87520535
Validation score: 0.525000
Iteration 4, loss = 0.67862159
Validation score: 0.600000
Iteration 5, loss = 0.52943889
Validation score: 0.675000
Iteration 6, loss = 0.42301443
Validation score: 0.725000
Iteration 7, loss = 0.33942270
Validation score: 0.750000
Iteration 8, loss = 0.27602812
Validation score: 0.750000
Iteration 9, loss = 0.22660596
Validation score: 0.750000
Iteration 10, loss = 0.18861426
Validation score: 0.775000
Iteration 11, loss = 0.15823997
Validation score: 0.775000
Iteration 12, loss = 0.13388429
Validation score: 0.775000
Iteration 13, loss = 0.11386469
Validation score: 0.775000
```

```
Iteration 14, loss = 0.09698632
Validation score: 0.787500
Iteration 15, loss = 0.08407431
Validation score: 0.800000
Iteration 16, loss = 0.07323753
Validation score: 0.825000
Iteration 17, loss = 0.06435131
Validation score: 0.825000
Iteration 18, loss = 0.05676487
Validation score: 0.825000
Iteration 19, loss = 0.05050037
Validation score: 0.825000
Iteration 20, loss = 0.04543220
Validation score: 0.812500
Iteration 21, loss = 0.04089021
Validation score: 0.812500
Iteration 22, loss = 0.03720510
Validation score: 0.812500
Iteration 23, loss = 0.03405820
Validation score: 0.812500
Iteration 24, loss = 0.03127900
Validation score: 0.812500
Iteration 25, loss = 0.02888221
Validation score: 0.812500
Iteration 26, loss = 0.02677985
Validation score: 0.812500
Iteration 27, loss = 0.02491267
Validation score: 0.812500
Validation score did not improve more than tol=0.000100 for 10
consecutive epochs. Stopping.
                   precision    recall  f1-score   support

     Colin Powell       0.81      0.90      0.85        61
  Donald Rumsfeld       0.84      0.71      0.77        51
    George W Bush       0.86      0.90      0.88       153
Gerhard Schroeder       0.85      0.69      0.76        32
       Tony Blair       0.78      0.78      0.78        45

         accuracy                           0.84       342
        macro avg       0.83      0.79      0.81       342
     weighted avg       0.84      0.84      0.83       342
```

RESULT:

Thus the facial recognition application with artificial neural network was executed.

EX.NO: 4 STUDY AND IMPLEMENT AMAZON TOOLKIT: SAGEMAKER

AIM:

To study and implement Amazon Toolkit: Sagemaker.

STUDY:

Amazon SageMaker is a fully managed machine learning service. With SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment. It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so there is no need to manage servers. It also provides common machine learning algorithms that are optimized to run efficiently against extremely large data in a distributed environment. With native support for bring-your-own-algorithms and frameworks, SageMaker offers flexible distributed training options that adjust to your specific workflows. Deploy a model into a secure and scalable environment by launching it with a few clicks from SageMaker Studio or the SageMaker console. Training and hosting are billed by minutes of usage, with no minimum fees and no upfront commitments.

Amazon-sagemaker-examples

SageMaker Automatic Model Tuning

These examples introduce SageMaker's hyperparameter tuning functionality which helps deliver the best possible predictions by running a large number of training jobs to determine which hyperparameter values are the most impactful.

XGBoost Tuning shows how to use SageMaker hyperparameter tuning to improve your model fit.

BlazingText Tuning shows how to use SageMaker hyperparameter tuning with the BlazingText built-in algorithm and 20_newsgroups dataset..

TensorFlow Tuning shows how to use SageMaker hyperparameter tuning with the pre-built TensorFlow container and MNIST dataset.

MXNet Tuning shows how to use SageMaker hyperparameter tuning with the pre-built MXNet container and MNIST dataset.

HuggingFace Tuning shows how to use SageMaker hyperparameter tuning with the pre-built HuggingFace container and 20_newsgroups dataset.

PROGRAM

Step 1: Create an Amazon SageMaker Notebook Instance

An Amazon SageMaker notebook instance is a fully managed machine learning Amazon Elastic Compute Cloud (Amazon EC2) compute instance that runs the Jupyter Notebook App. You use the notebook instance to create and manage Jupyter notebooks forpreprocessing data and to train and deploy machine learning models.

To create a SageMaker notebook instance

Open the Amazon SageMaker console at https://console.aws.amazon.com/sagemaker/

Choose Notebook instances, and then choose Create notebook instance.

On the Create notebook instance page, provide the following information (if a field is not mentioned, leave the default values):

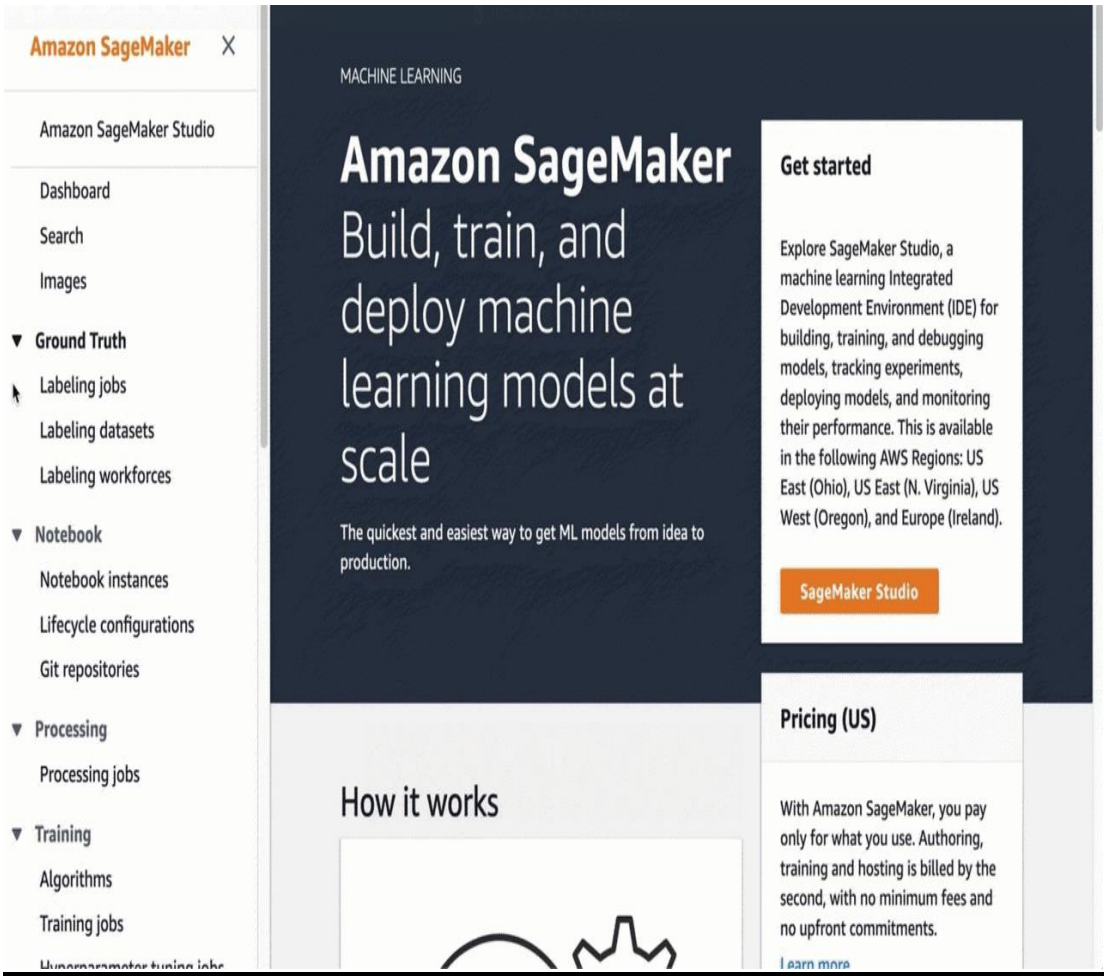
For Notebook instance name, type a name for your notebook instance.

For Notebook Instance type, choose ml.t2.medium. This is the least expensive instance type that notebook instances support, and it suffices for this exercise. If a ml.t2.medium instance type isn't available in your current AWS Region, choose ml.t3.medium.

For Platform Identifier, choose a platform type to create the notebook instance on. This platform type dictates the Operating System and the JupyterLab version that your notebook instance is created with.

For IAM role, choose Create a new role, and then choose Create role. This IAM role automatically gets permissions to access any S3 bucket that has sagemaker in the name. It gets these permissions through the AmazonSageMakerFullAccess policy, which SageMaker attaches to the role.

OUTPUT



RESULT:

Thus the Amazon Toolkit has been implemented and the study of Amazon sagemaker has been performed.

EX.NO:5 IMPLEMENTAION OF CHARACTER RECOGNITION USING MULTILAYER PERCEPTRON

Aim:

To implement character recognition using Multilayer Perceptron

ALGORITHM:

Step 1: Load data
Step 2: Flatten 28*28 images to a 784 vector for each image
Step 3: Normalize inputs from 0-255 to 0-1
Step 4: Define baseline model
Step 5: Create Model
Step 6: Compile Model
Step 7: Build the Model
Step 8: Fit the Model
Step 9: Evaluate the Model

PROGRAM

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# flatten 28*28 images to a 784 vector for each image
num_pixels = X_train.shape[1] * X_train.shape[2]
X_train = X_train.reshape((X_train.shape[0], num_pixels)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], num_pixels)).astype('float32')
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
# define baseline model
def baseline_model():
  # create model
  model = Sequential()
  model.add(Dense(num_pixels, input_dim=num_pixels, kernel_initializer='normal', activat
ion='relu'))
```

```
  model.add(Dense(num_classes, kernel_initializer='normal', activation='softmax'))
  # Compile model
  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
# build the model
model = baseline_model()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, ve
rbose=2)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Baseline Error: %.2f%%" % (100-scores[1]*100))
```

OUTPUT

```
Epoch 1/10 300/300 - 5s - loss: 0.2806 - accuracy: 0.9208 - val_loss: 0.1471 - val_accuracy: 0.9564 -
5s/epoch - 17ms/step
Epoch 2/10 300/300 - 6s - loss: 0.1107 - accuracy: 0.9677 - val_loss: 0.0983 - val_accuracy: 0.9694 -
6s/epoch - 21ms/step
Epoch 3/10 300/300 - 7s - loss: 0.0704 - accuracy: 0.9801 - val_loss: 0.0768 - val_accuracy: 0.9758 -
7s/epoch - 23ms/step
Epoch 4/10 300/300 - 4s - loss: 0.0491 - accuracy: 0.9863 - val_loss: 0.0716 - val_accuracy: 0.9778 -
4s/epoch - 14ms/step
Epoch 5/10 300/300 - 4s - loss: 0.0365 - accuracy: 0.9896 - val_loss: 0.0698 - val_accuracy: 0.9795 -
4s/epoch - 14ms/step
Epoch 6/10 300/300 - 4s - loss: 0.0264 - accuracy: 0.9929 - val_loss: 0.0692 - val_accuracy: 0.9788 -
4s/epoch - 14ms/step
Epoch 7/10 300/300 - 4s - loss: 0.0199 - accuracy: 0.9952 - val_loss: 0.0596 - val_accuracy: 0.9826 -
4s/epoch - 14ms/step
Epoch 8/10 300/300 - 4s - loss: 0.0137 - accuracy: 0.9969 - val_loss: 0.0606 - val_accuracy: 0.9812 -
4s/epoch - 14ms/step
Epoch 9/10 300/300 - 4s - loss: 0.0097 - accuracy: 0.9982 - val_loss: 0.0633 - val_accuracy: 0.9811 -
4s/epoch - 14ms/step
Epoch 10/10 300/300 - 4s - loss: 0.0075 - accuracy: 0.9989 - val_loss: 0.0614 - val_accuracy: 0.9820 -
4s/epoch - 14ms/step
Baseline Error: 1.80%
```

RESULT:

Thus the character recognition using Multilayer Perceptron is executed and Baseline error of 1.80% was obtained.

EX. NO:6 IMPLEMENTATION OF THE NON-PARAMETRIC LOCALLY WEIGHTED REGRESSION ALGORITHM TO FIT DATA POINTS

AIM:

To implement the non-parametric Locally Weighted Regression algorithm in order to fit data points and to select appropriate data set and to draw graphs

ALGORITHM:

Step1: Read the given data Sample to X and the curve (linear or non linear) to Y
Step2: Set the value for Smoothening parameter or Free parameter say τ
Step 3. Set the bias /Point of interest set x0 which is a subset of X
Step 4. Determine the weight matrix using:

$$w(x, x_o) = e^{-\frac{(x-x_o)^2}{2\tau^2}}$$

Step 5. Determine the value of model term parameter β using:

$$\hat{\beta}(x_o) = (X^T W X)^{-1} X^T W y$$

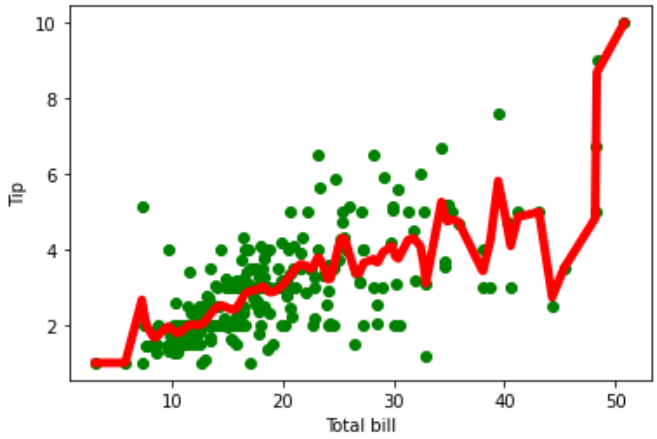
Step 6. Prediction = x0*β

PROGRAM:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
def kernel(point, xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m)))
    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
# load data points
```

```
data = pd.read_csv('/10-dataset.csv')
bill = np.array(data.total_bill)
tip = np.array(data.tip)
 #preparing and add 1 in bill
mbill = np.mat(bill)
mtip = np.mat(tip)
m= np.shape(mbill)[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T))
#set k here
ypred = localWeightRegression(X,mtip,0.5)
SortIndex = X[:,1].argsort(0)
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();
```

OUTPUT:



RESULT:

Thus the non-parametric Locally Weighted Regression algorithm in order to fit data points has been executed and the graph has been obtained.

EX.NO:7 IMPLEMENTATION OF SENTIMENT ANALYSIS USING RANDOM FOREST OPTIMIZATION ALGORITHM

AIM:

To implement Sentiment Analysis using Random Forest Optimization Algorithm.

ALGORITHM:

Step1: Importing required Libraries

Step2. Reading the dataset

Step3. Preprocessing

Step4. Creating the Bag of Words model

Step5. Splitting the dataset into the Training set and Test set

Step 6. Apply Random Forest Classifier

Step 7. Perform CAP Analysis

PROGRAM

```
#Importing required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
import seaborn as sn
# Reading the dataset
dataset = pd.read_csv('/content/Restaurant_Reviews.tsv', delimiter = '\t', quoting = 3)
# Preprocessing
nltk.download('stopwords')
corpus = [ ]
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
```

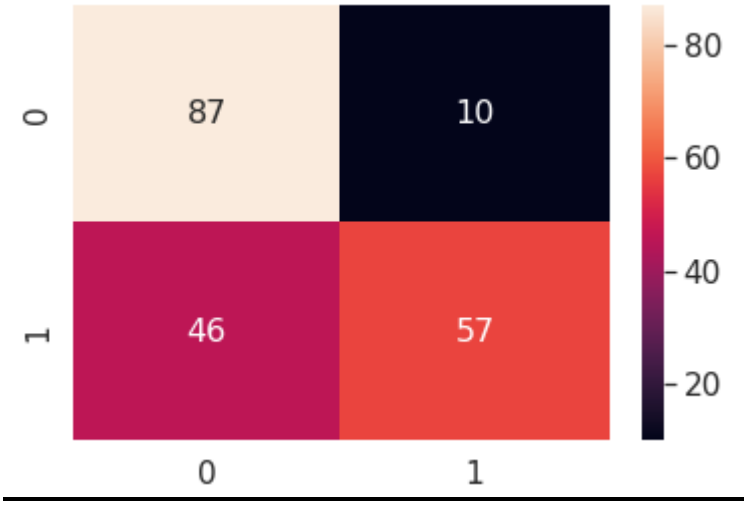
```
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english')
)]
    review = ' '.join(review)
    corpus.append(review)
# Creating the Bag of Words model
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
# Random Forest
rf_classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_sta
te = 0)
rf_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
cm_RandFor = confusion_matrix(y_test, y_pred_rf)
df_cm = pd.DataFrame(cm_RandFor, range(2), range(2))
# plt.figure(figsize=(10,7))
sn.set(font_scale=1.4) # for label size
sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size
plt.show()
#CAP Analysis
total = len(y_test)
one_count = np.sum(y_test)
zero_count = total - one_count
lm_RandFor = [y for _, y in sorted(zip(y_pred_rf, y_test), reverse = True)]
x = np.arange(0, total + 1)
y_RandFor = np.append([0], np.cumsum(lm_RandFor))
plt.figure(figsize = (10, 10))
plt.title('CAP Curve Analysis')
plt.plot([0, total], [0, one_count], c = 'k', linestyle = '--', label = 'Random Model')
plt.plot([0, one_count, total], [0, one_count, one_count], c = 'grey', linewidth = 2, label = 'P
```

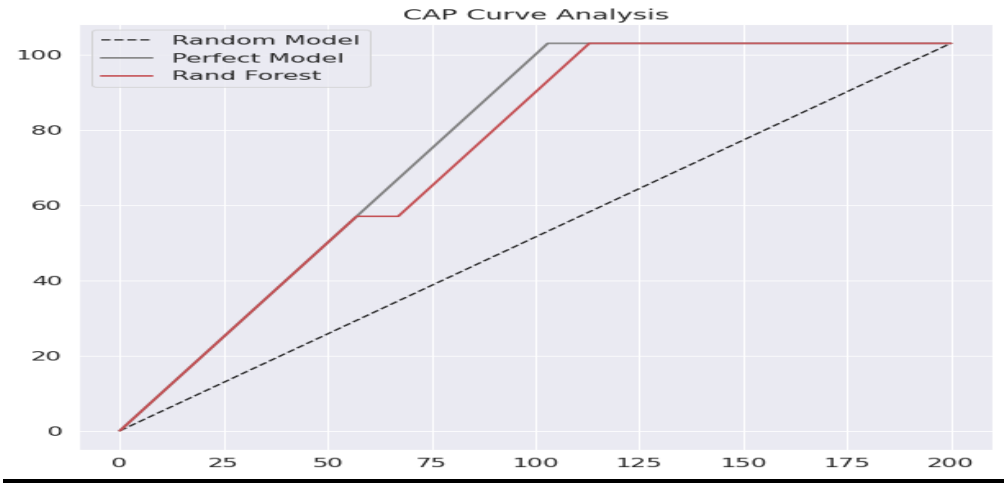
erfect Model')

plt.plot(x, y_RandFor, c = 'r', label = 'Rand Forest', linewidth = 2)

plt.legend()

OUTPUT





RESULT:

Thus the Sentiment Analysis using Random Forest Optimization Algorithm has been implemented and executed.

EX. NO:8 CONSTRUCTION OF BAYESIAN NETWORK

AIM:

To write a program to construct a Bayesian network considering medical data. Using this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

ALGORITHM:

Step1: Read the Heart Disease data from the respective location
Step2: Display the sample data
Step3: Model Bayesian Network.
Step4: Learning CPDs using Maximum Likelihood Estimators
Step5: Inferencing with Bayesian Network
Step6: Computing the Probability of HeartDisease given Age
Step7: Computing the Probability of HeartDisease given cholesterol

PROGRAM:

```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianNetwork
from pgmpy.inference import VariableElimination
heartDisease = pd.read_csv('/content/7-dataset.csv')
heartDisease = heartDisease.replace('?',np.nan)
print('Sample instances from the dataset are given below')
print(heartDisease.head())
model= BayesianNetwork([('age','heartdisease'),('gender','heartdisease'),('exang','heartdiseas
e'),('cp','heartdisease'),('heartdisease','restecg'),('heartdisease','chol')])
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)
print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)
print('\n 1. Probability of HeartDisease given evidence= restecg')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)
print('\n 2. Probability of HeartDisease given evidence= cp')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

OUTPUT

```
 Sample instances from the dataset are given below
      Age gender  cp  trestbps  chol  fbs  restecg  thalach  exang
0     63   1       1      145   233    1       2      150      0
1     67   1       4      160   286    0        2      108      1
2     67   1       4      120   229    0        2      129      1
3     37   1       3      130   250    0        0      187      0
4     41   0       2      130   204    0        2      172      0

      oldpeak  slope   ca    thal  heartdisease
0      2.3       3     0      6          0
1      1.5       2     3      3          2
2      2.6       2     2      7          1
3      3.5       3     0      3          0
4      1.4       1     0      3          0

Inferencing with Bayesian Network:

 1. Probability of HeartDisease given evidence= restecg
Finding Elimination Order: : 100% 4/4 [00:00<00:00, 28.19it/s]
Eliminating: cp: 100% 4/4 [00:00<00:00, 52.25it/s]
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.1012 |
+-----------------+---------------------+
| heartdisease(1) |              0.0000 |
+-----------------+---------------------+
| heartdisease(2) |              0.2392 |
+-----------------+---------------------+
| heartdisease(3) |              0.2015 |
+-----------------+---------------------+
| heartdisease(4) |              0.4581 |
+-----------------+---------------------+


2. Probability of HeartDisease given evidence= cp
Finding Elimination Order: : 0% 0/3 [00:00<?, ?it/s]
Eliminating: exang: 100% 3/3 [00:00<00:00, 52.18it/s]
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.3610 |
+-----------------+---------------------+
| heartdisease(1) |              0.2159 |
+-----------------+---------------------+
| heartdisease(2) |              0.1373 |
+-----------------+---------------------+
| heartdisease(3) |              0.1537 |
+-----------------+---------------------+
| heartdisease(4) |              0.1321 |
+-----------------+---------------------+
```

RESULT:

Thus by using Bayesian network the diagnosis of heart patients using standard Heart Disease Data Set has been demonstrated.

EX. NO:9 ONLINE FRAUD DETECTION USING XGBoost

AIM:

To implement online fraud detection using XGBoost.

ALGORITHM:

Step1. Import the required packages.
Step 2. Loading the required dataset from the Specific Location.
Step 3. Use the XGBoost model.
Step 4. Identify the total number of transactions and the fraudulent transaction using the Model.
Step 5. Identify the Accuracy score using the XGBoost model

PROGRAM:

```
#Packages related to general operating system & warnings
import os
import warnings
warnings.filterwarnings('ignore')
#Packages related to data importing, manipulation, exploratory data #analysis, data underst
anding
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
from termcolor import colored as cl # text customization
#Packages related to data visualizaiton
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
#Setting plot sizes and type of plot
plt.rc("font", size=14)
plt.rcParams['axes.grid'] = True
plt.figure(figsize=(6,3))
plt.gray()
from matplotlib.backends.backend_pdf import PdfPages
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import metrics
from sklearn.impute import MissingIndicator, SimpleImputer
from sklearn.preprocessing import  PolynomialFeatures, KBinsDiscretizer, FunctionTransf
ormer
```

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, LabelBinarizer, Ordinal
Encoder
import statsmodels.formula.api as smf
import statsmodels.tsa as tsa
from sklearn.linear_model import LogisticRegression, LinearRegression, ElasticNet, Lasso,
 Ridge
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
# from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphvi
z, export
from sklearn.ensemble import BaggingClassifier, BaggingRegressor,RandomForestClassifi
er,RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier,GradientBoostingRegressor, Ada
BoostClassifier, AdaBoostRegressor
from sklearn.svm import LinearSVC, LinearSVR, SVC, SVR
from xgboost import XGBClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
data=pd.read_csv("/content/creditcard.csv")
Total_transactions = len(data)
normal = len(data[data.Class == 0])
fraudulent = len(data[data.Class == 1])
fraud_percentage = round(fraudulent/normal*100, 2)
print(cl('Total number of Transactions are {}'.format(Total_transactions), attrs = ['bold']))
print(cl('Number of Normal Transactions are {}'.format(normal), attrs = ['bold']))
print(cl('Number of fraudulent Transactions are {}'.format(fraudulent), attrs = ['bold']))
print(cl('Percentage of fraud Transactions is {}'.format(fraud_percentage), attrs = ['bold']))
sc = StandardScaler()
amount = data['Amount'].values
data['Amount'] = sc.fit_transform(amount.reshape(-1, 1))
data.drop(['Time'], axis=1, inplace=True)
data.shape
data.drop_duplicates(inplace=True)
```

```
data.shape
X = data.drop('Class', axis = 1).values
y = data['Class'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 1)
# XGBoost
xgb = XGBClassifier(max_depth = 4)
xgb.fit(X_train, y_train)
xgb_yhat = xgb.predict(X_test)
print('Accuracy score of the XGBoost model is {}'.format(accuracy_score(y_test, xgb_yhat
)))
print('F1 score of the XGBoost model is {}'.format(f1_score(y_test, xgb_yhat)))
```

OUTPUT:

```
Total number of Transactions are 15936
Number of Normal Transactions are 15862
Number of fraudulent Transactions are 73
Percentage of fraud Transactions is 0.46
Accuracy score of the XGBoost model is 0.998963998963999
F1 score of the XGBoost model is 0.9130434782608695
```

RESULT:

Thus the online fraud detection using XGBoost algorithm has been implemented and executed successfully.

EX. NO:10 MINI PROJECT

Mini-project: students work in team on any socially relevant problem that needs a machine learning based solution, and evaluate the model performance

Documentation

Problem Statement: Requirement of the selected problem aim.

Software Requirements: Specification of the software requirement for the mini project.

Algorithms & Dataset used: Algorithm and dataset description.

Implementation: Sample snippets

Results & Discussions: Results of the proposed work. Model performance and comparisons are required

Future Enhancement: Future work has to be explained.