PROGRAM 5

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\_categorically\_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: %.21% % % (100-scores[1]\*100))

PROGRAM 6:

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):

weikerne point,xmat,k)

W (X.T (wei X)).I (X.T(wei\*ymat.T))

return W

def local Weight Regression(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();

PROGRAM 7:

#Importing required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import re

import nik

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer from sklearn.feature\_extraction.text import Count Vectorizer

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(T^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('cnglish)

review-join(review)

corpus.append(review)

Creating the Bag of Words model Cv Count Vectorizer(max features=1500)

X-ev.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=1 np.sum(y test)

zero\_count = total - one count

Im RandForly for y in sorted(ziply\_pred\_rf. y\_test), reverse=True)] x= np.arange(0. total + 1)

y RandFor= np.append([0], np.cumsum(Im\_RandFor)) plt.figure(figsize (10. 10))

plt.title('CAP Curve Analysis")

plt.plot([0, total]. 10. one\_count]. ck linestyle label='Random Model)

plt.plot(10. one count. total]. 10. one count, one\_count], c='grey'. linewidth=2, label="P

erfect Model')

DE

- 80

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

plt.legend()

PROGRAM 8:

import numpy as np import pandas as pd

import csv

from pgmpy.estimators import Maximuml.ikelihoodEstimator

from pgmpy.models import BayesianNetwork

from pgmpy.inference import VariableElimination

heart Disease = pd.read\_csv(/content/7-dataset.csv')

heart Disease heart Disease.replace("?,np.nan)

print('Sample instances from the dataset are given below) print(heart Disease.head())

model Bayesian Network([Cage', heart disease).("gender". heartdisease),('exang'.'heartdiseas e).('ep'. 'heartdisease).(heartdisease', 'restecg),("heart disease','chol)]) print("nLearning CPD using Maximum likelihood estimators') model.fit(heart Disease. estimator-Maximuml.ikelihoodEstimator)

print(An Inferencing with Bayesian Network:") Heart Diseasetest infer VariableElimination(model)

print('n 1. Probability of Heart Disease given evidence = restecg')

q1-Heart Diseasetest infer.query(variables["heartdisease"],evidence={'restecg':1})

print(q1)

print("n 2. Probability of Heart Disease given evidence=cp")

q2 Heart Diseasetest infer.query(variables=['heartdisease], evidence={'cp:2})

print(q2)

PROGRAM 9:

#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.reParams['axes.grid'] = True

plt.figure(figsize (6.3))

pl.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, KBins Discretizer. Function Transf

ormer

from sklearn preprocessing import LabelEncoder, OneHotEncoder, LabelBinarizer, Ordinal

from sklearn preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

Encoder

import statsmodels formula.api as smf

import statsmodels.tsa as tsa

from sklearn.linear model import Logistic Regression, Linear Regression, ElasticNet, Lasso,

Ridge

from sklearn.neighbors import KNeighbors Classifier, KNeighborsRegressor # from sklearn.tree import Decision Tree Classifier, Decision TreeRegressor, export\_graphvi

z, export

from sklearn ensemble import BaggingClassifier, BaggingRegressor, RandomForestClassifi

er,RandomForestRegressor

from sklearn ensemble import Gradient Boosting Classifier, Gradient Boosting Regressor. Ada BoostClassifier, AdaBoostRegressor

from sklearn.svm import LinearSVC, LinearSVR, SVC, SVR

from xgboost import XGBClassifier from sklearn, metrics import fl 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= ID fraud percentage=round(fraudulent/normal 100, 2)

print(ck 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 StandardScalert)

amount datal Amount values

data['Amount"] se.fit\_transform(amount.reshape(-1, 1)) data.drop(['Time'], axis=1. inplace=True)

data shape

data.drop duplicatest 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("FI score of the XGBoost model is (}'.format(fl\_score(y\_test, xgb\_yhat)))