AD8511 MACHINE LEARNING LABORATORY

Name of the Student : .....

Register Number : .....

Year / Semester / Section :.....

Branch : .....

Department of Artificial Intelligence and Data Science

CHENNAI INSTITUTE OF TECHNOLOGY
(Autonomous)

CHENNAI INSTITUTE OF TECHNOLOGY
(Autonomous)

Register No: | | | | | | | | | | | | |

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of work done by Mr./Ms..... of ..... semester B.E.....in the ..... Laboratory during the academic year .....

Staff in-Charge **Head of the Department**

Submitted for the University Practical Examination held on .....

Internal Examiner **External Examiner**

Date: ..... **Date: .....**

LIST OF PROGRAMS

EX.NO : 1	DECISION TREE ALGORITHM
DATE:	

AIM:

To write a program to implement the decision tree model.

PROCEDURE:

- Import the necessary packages.
- Load the dataset into a data frame.
- Split the data for training and testing.
- Train decision tree model on the given training data.
- Predict the class for each data in training set.
- Print the accuracy and other evaluation metrics.

PROGRAM:

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np

# Load the Iris dataset to a variable
iris = load_iris()
# Load the data into 2 different pandas dataframes.
# X for the features
X = pd.DataFrame(iris.data, columns=iris.feature_names)
# y for the targets
y = pd.Categorical.from_codes(iris.target, iris.target_names)

X.head()

print(y)

# Convert categorical data into dummy/indicator variables.
y = pd.get_dummies(y)

y.head()

# import the function
```

```
from sklearn.model_selection import train_test_split

# Split our data into the training sets and the testing sets
# Each set has a pair of X and y
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

## Import and create the model
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()

# Train our model with our training data.
dt.fit(X_train, y_train)

# Call the .predict() function of the model and feed in the testing data
y_pred = dt.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score
species = np.array(y_test).argmax(axis=1)
predictions = np.array(y_pred).argmax(axis=1)
print("Confusion Matrix: \n", confusion_matrix(species, predictions))
print("Accuracy: \n", accuracy_score(species, predictions))
```

OUTPUT:

```
['setosa', 'setosa', 'setosa', 'setosa', 'setosa', ..., 'virginica', 'virginica', 'virginica', 'virginica',
'virginica']
Length: 150
Categories (3, object): ['setosa', 'versicolor', 'virginica']
Confusion Matrix:
 [[13 0  0]
 [ 0 15  1]
 [ 0  0  9]]
Accuracy:
 0.9736842105263158
```

INFERENCE:

The decision tree model is trained with the iris dataset having an accuracy of 97%.

RESULT:

The program to implement decision tree algorithm has been executed successfully.

EX.NO : 2	DETECTING SPAM MAILS USING SUPPORT VECTOR MACHINE
DATE:	

AIM:

To write a program to detect spam mails using Support Vector machine.

PROCEDURE:

- Import the necessary packages.
- Load the email dataset.
- Split the dataset into training and test data
- Create a Support Vector machine model, train the model using the training data.
- Train the model until the desired accuracy is reached.
- Do prediction using the testing data.

PROGRAM:

```
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
data = pd.read_csv('/content/email.csv')
X = data['Message'].values
y = data['Category'].values
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 String to Integer
cv = CountVectorizer()
X_train = cv.fit_transform(X_train)
X_test = cv.transform(X_test)
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
print("Classifier Score:")
print(classifier.score(X_test,y_test)
```

OUTPUT:

Classifier Score:
0.9811559192825112

INFERENCE:

The Support vector machine model is trained with the email dataset and the accuracy of 98% is achieved on test set.

RESULT:

The program to detect spam mails using Support vector machine has been executed successfully.

EX.NO : 3	FACE RECOGNITION USING CNN
DATE:	

AIM:

To write a program to recognize the face using CNN.

DATASET:

Face-images.zip
The data contains cropped face images of 16 people divided into Training and testing.

PROCEDURE:

- Import the necessary packages.
- Preprocess the images for training and testing data.
- Create lookup tables for all faces.
- Create CNN deep learning model and train the model using training data.
- Train the model until the accuracy is greater than 90%.
- Do prediction using the testing data.

PROGRAM:

 # IMAGE PRE-PROCESSING FOR TRAINING AND TESTING DATA

```
# Specifying the folder where images are present

TrainingImagePath = 'D:\\Face_Recognition\\Face-Images\\Face Images\\Final Training Images'
from keras.preprocessing.image import ImageDataGenerator

# Defining pre-processing transformations on raw images of training data
# These hyper parameters helps to generate slightly twisted versions of the original image,
#which leads to a better model, since it learns on the good and bad mix of images

train_datagen = ImageDataGenerator(
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True)

# Defining pre-processing transformations on raw images of testing data
# No transformations are done on the testing images
```

```
test_datagen = ImageDataGenerator()

# Generating the Training Data

training_set = train_datagen.flow_from_directory(
        TrainingImagePath,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')


# Generating the Testing Data

test_set = test_datagen.flow_from_directory(
        TrainingImagePath,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

# Printing class labels for each face
test_set.class_indices
```

OUTPUT:

```
Found 244 images belonging to 16 classes.

Found 244 images belonging to 16 classes.

{'face1': 0,

 'face10': 1,

 'face11': 2,

 'face12': 3,

 'face13': 4,

 'face14': 5,

 'face15': 6,

 'face16': 7,

 'face2': 8,

 'face3': 9,

 'face4': 10,
```

```
 'face5': 11,
 'face6': 12,
 'face7': 13,
 'face8': 14,
 'face9': 15}
```

 # CREATING LOOKUP TABLES FOR ALL FACES

```
# class indices have the numeric tag for each face

TrainClasses=training_set.class_indices

# Storing the face and the numeric tag for future reference

ResultMap={}
for faceValue,faceName in zip(TrainClasses.values(),TrainClasses.keys()):
    ResultMap[faceValue]=faceName

# Saving the face map for future reference

import pickle
with open("ResultsMap.pkl", 'wb') as fileWriteStream:
    pickle.dump(ResultMap, fileWriteStream)

# The model will give answer as a numeric tag
# This mapping will help to get the corresponding face name for it

print("Mapping of Face and its ID",ResultMap)
```

```
# The number of neurons for the output layer is equal to the number of faces
OutputNeurons=len(ResultMap)
print('\n The Number of output neurons: ', OutputNeurons)
```

OUTPUT:

```
Mapping of Face and its ID {0: 'face1', 1: 'face10', 2: 'face11', 3: 'face12', 4: 'face13', 5: 'face
14', 6: 'face15', 7: 'face16', 8: 'face2', 9: 'face3', 10: 'face4', 11: 'face5', 12: 'face6', 13: 'f
ace7', 14: 'face8', 15: 'face9'}

 The Number of output neurons:  16
```

 # CREATE CNN DEEPLEARNING MODEL

```
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPool2D
from keras.layers import Flatten
from keras.layers import Dense

'''Initializing the Convolutional Neural Network'''
classifier= Sequential()

''' STEP--1 Convolution
# Adding the first layer of CNN
# we are using the format (64,64,3) because we are using TensorFlow backend
# It means 3 matrix of size (64X64) pixels representing Red, Green and Blue components of pixels
'''
classifier.add(Convolution2D(32, kernel_size=(5, 5), strides=(1, 1), input_shape=(64,64,3), activation='relu'))
```

```
'''# STEP--2 MAX Pooling'''
classifier.add(MaxPool2D(pool_size=(2,2)))

'''############### ADDITIONAL LAYER of CONVOLUTION for better accuracy
#################'''
classifier.add(Convolution2D(64, kernel_size=(5, 5), strides=(1, 1), activation='relu'))

classifier.add(MaxPool2D(pool_size=(2,2)))

'''# STEP--3 FLattening'''
classifier.add(Flatten())

'''# STEP--4 Fully Connected Neural Network'''
classifier.add(Dense(64, activation='relu'))

classifier.add(Dense(OutputNeurons, activation='softmax'))

'''# Compiling the CNN'''
#classifier.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
classifier.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics=[''accuracy''])

############################################################
import time
# Measuring the time taken by the model to train
StartTime=time.time()

# Starting the model training
classifier.fit_generator(
```

```
        training_set,
        steps_per_epoch=30,
        epochs=10,
        validation_data=test_set,
        validation_steps=10)

EndTime=time.time()
print("###### Total Time Taken: ", round((EndTime-StartTime)/60), 'Minutes ######')
```

OUTPUT:

```
2.7025 - val_accuracy: 0.9318
Epoch 6/10
30/30 [==============================] - 9s 294ms/step - loss: 0.0161 - accuracy: 0.9967 - val_loss:
2.6716 - val_accuracy: 0.9545
Epoch 7/10
30/30 [==============================] - 9s 295ms/step - loss: 0.0383 - accuracy: 0.9890 - val_loss:
2.6862 - val_accuracy: 0.9545
Epoch 8/10
30/30 [==============================] - 9s 297ms/step - loss: 0.0306 - accuracy: 0.9870 - val_loss:
2.6709 - val_accuracy: 0.9392
Epoch 9/10
30/30 [==============================] - 8s 278ms/step - loss: 0.0313 - accuracy: 0.9923 - val_loss:
2.6518 - val_accuracy: 0.9903
Epoch 10/10
30/30 [==============================] - 9s 294ms/step - loss: 0.0155 - accuracy: 0.9946 - val_loss:
2.6692 - val_accuracy: 0.9708
############## Total Time Taken:  1 Minutes #############
```

 # MAKING SINGLE PREDICTION

```
import numpy as np
from keras.preprocessing import image

ImagePath='/Users/farukh/Python Case Studies/Face Images/Final Testing
Images/face4/3face4.jpg'
test_image=image.load_img(ImagePath,target_size=(64, 64))
```

```
test_image=image.img_to_array(test_image)

test_image=np.expand_dims(test_image,axis=0)

result=classifier.predict(test_image,verbose=0)
#print(training_set.class_indices)

print('####'*10)
print('Prediction is: ',ResultMap[np.argmax(result)])
```

OUTPUT:

```
########################################
Prediction is:  face4
```

INFERENCE:

The CNN model is trained with the dataset having an accuracy of 97%.

RESULT:

The program to recognize the face using CNN has been executed successfully.

EX.NO : 4	HANDWRITTEN CHARACTER RECOGNITION USING MLP
DATE:	

AIM:

To write a program to implement Handwritten Character Recognition using MLP

PROCEDURE:

- Import the necessary packages.
- Load the MNIST dataset
- Split the dataset into training and test data
- Create an MLP model and train the model using training data.
- Train the model until the accuracy is greater than 90%.
- Do prediction using the testing data.

PROGRAM:

```
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier

X, y = fetch_openml("mnist_784", version=1, return_X_y=True)
X = X / 255.0
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]

classifier = MLPClassifier(
    hidden_layer_sizes=(50,20,10),
    max_iter=100,
    alpha=1e-4,
    solver="sgd",
    verbose=10,
    random_state=1,
    learning_rate_init=0.1,
)
# fit the model on the training data
classifier.fit(X_train, y_train)

print("Training set score: %f" % classifier.score(X_train, y_train))
```

```
print("Test set score: %f" % classifier.score(X_test, y_test))

print(“Predictions”,classifier.predict(X_test))
```

OUTPUT:

```
Training set score: 0.993683
Test set score: 0.966700
Predictions ['7' '2' '1' ... '4' '5' '6']
```

INFERENCE:

The locally weighted regression model is trained with the MNIST dataset and the accuracy of 96% is achieved on test set

RESULT:

The program to implement Handwritten Character Recognition using MLP has been executed successfully.

EX.NO : 5	LOCALLY WEIGHTED REGRESSION
DATE:	

AIM:

To write a program to implement and plot the locally weighted regression line

PROCEDURE:

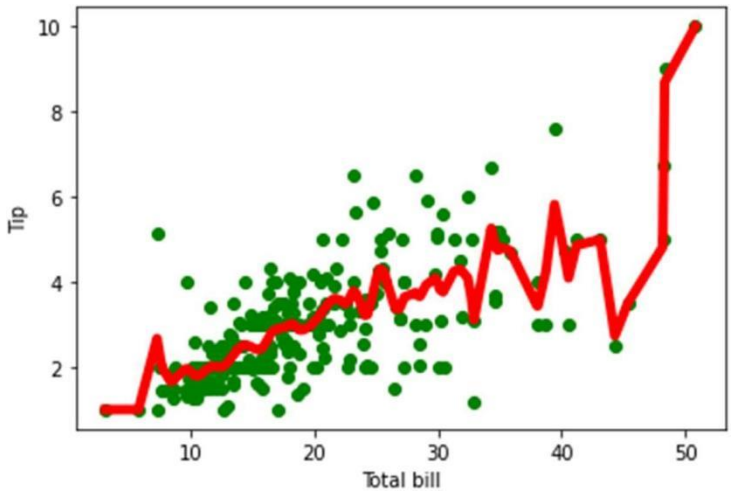
- Import the necessary packages.
- Load the dataset into a dataframe.
- Split the first two columns of data as X and Y.
- Train locally weighted regression model on the training data
- Plot the locally weighted regression line and label x-axis and y-axis.

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
```

```
data = pd.read_csv('tips.csv')
bill = np.array(data.total_bill)
tip = np.array(data.tip)

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))

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:



INFERENCE:

The locally weighted regression model is trained with the dataset and the regression is plotted

RESULT:

The program to implement locally weighted regression has been executed successfully.

EX.NO : 6	SENTIMENT ANALYSIS USING RANDOM FOREST OPTIMIZATION ALGORITHM
DATE:	

AIM:

To write a program to implement Sentiment Analysis using Random Forest Optimization Algorithm using Python/Java ML library.

PROCEDURE:

- Import the necessary packages.
- Load the heart disease dataset.
- Split the dataset into training and test data
- Create a Random forest optimization model and train the model using training data.
- Train the model until the desired accuracy is reached.
- Do prediction using the testing data.

PROGRAM:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import  nltk
#nltk.download()
#nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
#nltk.download('wordnet')

import sklearn
from sklearn import model_selection, preprocessing, linear_model, metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import ensemble

import re
```

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

%%time
train = pd.read_csv('/kaggle/input/twitter-sentiment-analysis-hatred-speech/train.csv')
test = pd.read_csv('/kaggle/input/twitter-sentiment-analysis-hatred-speech/test.csv')
print(colored("\nDATASETS WERE SUCCESFULLY LOADED...", color = "green", attrs =
["dark", "bold"]))

train.head(n = 5).style.background_gradient(cmap = "summer")

test.head(n = 5).style.background_gradient(cmap = "summer")

train.info()

train.groupby("label").count().style.background_gradient(cmap = "summer")

train['label'].value_counts()

pos = 100*len(train.loc[train['label']==0,'label'])/len(train['label'])
neg = 100*len(train.loc[train['label']==1,'label'])/len(train['label'])

train[train['label']==0].head(20)

train[train['label']==1].head(20)

train_len = train['tweet'].str.len()
test_len = test['tweet'].str.len()

c=CountVectorizer(stop_words='english')
word=c.fit_transform(train.tweet)
summation=word.sum(axis=0)

freq=[(word,summation[0,i]) for word,i in c.vocabulary_.items()]
freq=sorted(freq,key=lambda x:x[1],reverse=True)
frequency = pd.DataFrame(freq, columns=['word', 'freq'])

train.isnull().sum()

def num_of_words(df):
    df['word_count'] = df['tweet'].apply(lambda x : len(str(x).split(" ")))
    print(df[['tweet','word_count']].head())

def num_of_chars(train):
```

```
    train['char_count'] = train['tweet'].str.len() ## this also includes spaces
    print(train[['tweet','char_count']].head())

def avg_word(sentence):
    words = sentence.split()
    return (sum(len(word) for word in words)/len(words))

def avg_word_length(df):
    df['avg_word'] = df['tweet'].apply(lambda x: avg_word(x))

set(stopwords.words('english'))

def stop_words(df):
    df['stopwords'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x in stop]))

def hash_tags(df):
    df['hashtags'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.startswith('#')]))

def num_numerics(df):
    df['numerics'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.isdigit()]))

def num_uppercase(df):
    df['upper_case'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.isupper()]))

train["tweet"] = train["tweet"].apply(lambda x: " ".join(x.lower() for x in x.split()))
test["tweet"] = test["tweet"].apply(lambda x: " ".join(x.lower() for x in x.split()))

train["tweet"] = train["tweet"].str.replace('[^\w\s]','')
test["tweet"] = test["tweet"].str.replace('[^\w\s]','')

train['tweet'] = train['tweet'].str.replace('\d','')
test['tweet'] = test['tweet'].str.replace('\d','')

sw = stopwords.words("english")
train['tweet'] = train['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in sw))
test['tweet'] = test['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in sw))

train = train.drop("id", axis = 1)
test = test.drop("id", axis = 1)

corpus = [
        'This is the first document.',
        'This document is the second document.',
```

```
        'And this is the third one.',
        'Is this the first document?',
       ]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)

vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2))
X2 = vectorizer2.fit_transform(corpus)

corpus = [
        'This is the first document.',
        'This document is the second document.',
        'And this is the third one.',
        'Is this the first document?',
       ]
vectorizer = HashingVectorizer(n_features=2**4)
X = vectorizer.fit_transform(corpus)

def lower_case(df):
   df['tweet'] = df['tweet'].apply(lambda x: " ".join(x.lower() for x in x.split()))
   print(df['tweet'].head())

freq = pd.Series(' '.join(train['tweet']).split()).value_counts()[:10]
freq

freq = list(freq.index)

def frequent_words_removal(df):
   df['tweet'] = df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))

freq = pd.Series(' '.join(train['tweet']).split()).value_counts()[-10:]
freq

freq = list(freq.index)

def rare_words_removal(df):
   df['tweet'] = df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))

def spell_correction(df):
   return df['tweet'][:5].apply(lambda x: str(TextBlob(x).correct()))

def tokens(df):
```

```
    return TextBlob(df['tweet'][1]).words

st = PorterStemmer()

def stemming(df):
    return df['tweet'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))

def combination_of_words(df):
    return (TextBlob(df['tweet'][0]).ngrams(2))

def term_frequency(df):
    tf1 = (df['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis =
0).reset_index()
    tf1.columns = ['words','tf']
    return tf1.head()

tf1 = (train['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis =
0).reset_index()
tf1.columns = ['words','tf']
tf1.head()

tf2 = (test['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis =
0).reset_index()
tf2.columns = ['words','tf']
tf2.head()

tf1 = (train['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis =
0).reset_index()
tf1.columns = ['words','tf']

for i,word in enumerate(tf1['words']):
    tf1.loc[i, 'idf'] = np.log(train.shape[0]/(len(train[train['tweet'].str.contains(word)])))

tf1['tfidf'] = tf1['tf'] * tf1['idf']
tf1

tfidf = TfidfVectorizer(max_features=1000, lowercase=True, analyzer='word',
stop_words= 'english',ngram_range=(1,1))
train_vect = tfidf.fit_transform(train['tweet'])
train_vect

bow = CountVectorizer(max_features=1000, lowercase=True, ngram_range=(1,1),analyzer =
"word")
```

```
train_bow = bow.fit_transform(train['tweet'])
train_bow

def polarity_subjectivity(df):
    return df['tweet'][:5].apply(lambda x: TextBlob(x).sentiment)

def sentiment_analysis(df):
    df['sentiment'] = df['tweet'].apply(lambda x: TextBlob(x).sentiment[0] )
    return df[['tweet','sentiment']].head()

x = train["tweet"]
y = train["label"]
train_x, test_x, train_y, test_y = model_selection.train_test_split(x, y, test_size = 0.20, shuffle =
True, random_state = 11)
print(colored("\nDIVIDED SUCCESFULLY...", color = "green", attrs = ["dark", "bold"]))

vectorizer = CountVectorizer()
vectorizer.fit(train_x)

x_train_count = vectorizer.transform(train_x)
x_test_count = vectorizer.transform(test_x)

x_train_count.toarray()

tf_idf_word_vectorizer = TfidfVectorizer()
tf_idf_word_vectorizer.fit(train_x)

x_train_tf_idf_word = tf_idf_word_vectorizer.transform(train_x)
x_test_tf_idf_word = tf_idf_word_vectorizer.transform(test_x)

x_train_tf_idf_word.toarray()

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model = model.fit(x_train_count, train_y)
accuracy = model_selection.cross_val_score(model,
                                x_test_count,
                                test_y,
                                cv = 20).mean()

print(colored("\nRandom Forest Classifier model with 'count-vectors' method", color = "red",
attrs = ["dark", "bold"]))
```

```
print(colored("Accuracy ratio: ", color = "red", attrs = ["dark", "bold"]), accuracy)
```

OUTPUT:

```
Random Forest Classifier model with 'count-vectors' method
Accuracy ratio: 0.9466624216300941
```

EX.NO : 7	HEART DISEASE PREDICTION USING BAYESIAN NETWORK
DATE:	

AIM:

To write a program to implement Heart Disease prediction using Bayesian network using Python/Java ML library.

PROCEDURE:

- Import the necessary packages.
- Load the heart disease dataset.
- Split the dataset into training and test data
- Create a Bayesian network model and train the model using training data.
- Train the model until the desired accuracy is reached.
- Do prediction using the testing data.

PROGRAM:

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

# visualization tools
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_profiling as pp

df = pd.read_csv("../input/heart-disease-uci/heart.csv")

X=df.iloc[:, 0:13]
X.head()

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
classifier = GaussianNB()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))

from sklearn.metrics import f1_score
f1_score = f1_score(y_test, y_pred)
print("F1 Score:")
print(f1_score)
```

OUTPUT:

```
Accuracy Score:
0.8688524590163934

F1 Score:
0.870967741935484
```

INFERENCE:

The Bayesian network model is trained with the heart disease dataset and the accuracy of 86% is achieved on test set

RESULT:

The program to implement heart disease prediction using Bayesian network using Python/Java ML library has been executed successfully.

EX.NO : 8	MACHINE LEARNING ALGORITHM TO IMPLEMENT ONLINE FRAUD DETECTION
DATE:	

AIM:

To write a program to implement Online Fraud Detection using Deep Neural Network model using Python/Java ML library.

PROCEDURE:

- Import the necessary packages.
- Load the heart disease dataset.
- Split the dataset into training and test data
- Create a deep neural network model and train the model using training data.
- Train the model until the desired accuracy is reached.
- Do prediction using the testing data.

PROGRAM:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from imblearn.over_sampling import SMOTE

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, confusion_matrix

import tensorflow as tf
from tensorflow.keras import Sequential

data = pd.read_csv('../input/online-payments-fraud-detection-
dataset/PS_20174392719_1491204439457_log.csv')
data.head()
```

```
obj_cols = data.select_dtypes(include='object').columns
obj_cols

num_cols = data.select_dtypes(exclude='object').columns
num_cols

data['isFraud'].value_counts(normalize=True)

data['diffOrg'] = data['oldbalanceOrg']-data['newbalanceOrig']
data['diffDest'] = data['oldbalanceDest'] - data['newbalanceDest']

data = data.drop(['oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest',
'nameOrig', 'nameDest'], axis=1)

train_data, test_data = train_test_split(data, test_size=0.3, random_state=42)

x = train_data.drop('isFraud', axis=1)
y = train_data['isFraud']

x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size=0.25, random_state=42)

x_train.sample(8)

le = LabelEncoder()
x_train['type'] = le.fit_transform(x_train['type'])
x_valid['type'] = le.transform(x_valid['type'])

x_train.sample(4)

rs = RobustScaler()
x_train = rs.fit_transform(x_train)
x_valid = rs.transform(x_valid)

sm = SMOTE(random_state=2)
x_train_res, y_train_res = sm.fit_resample(x_train, y_train)

model = Sequential([
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics='accuracy')

callbacks = tf.keras.callbacks.EarlyStopping(patience=4, monitor='val_loss', mode='min')
history = model.fit(x_train_res, y_train_res, epochs=20, steps_per_epoch=100000,
validation_data=(x_valid, y_valid), callbacks=[callbacks])

x_test = test_data.drop('isFraud', axis=1)
y_test = test_data['isFraud']

x_test['type'] = le.transform(x_test['type'])
x_test = rs.transform(x_test)

model.evaluate(x_valid, y_valid)

model.evaluate(x_test, y_test)
```

OUTPUT:

```
loss: 0.0550 - accuracy: 0.9699

[0.05497144162654877, 0.9699360132217407]

loss: 0.0555 - accuracy: 0.9702

[0.0554625578224659, 0.9701946973800659]
```

INFERENCE:

The Deep Neural Network model is trained with the dataset and the accuracy of 97% is achieved on test set

RESULT:

The program to implement Online Fraud Detection using Deep Neural Network using Python/Java ML library has been executed successfully.

EX.NO :	SAGEMAKER (AWS)
DATE:	

AIM:

To Study and implement amazon toolkit: Sagemaker.

PROCEDURE:

- Create an Amazon SageMaker notebook instance
- Prepare the data.
- Access the jupyter notebook from the aws poral.
- Import the necessary packages.
- Load the data and split the dataset into training and testing samples.
- Train the ML model
- Deploy the model
- Evaluate the model performance

PROGRAM:

```
# import libraries
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
my_region = boto3.session.Session().region_name # set the region of the instance

# this line automatically looks for the XGBoost image URI and builds an XGBoost
container.
xgboost_container = sagemaker.image_uris.retrieve("xgboost", my_region, "latest")

print("Success - the MySageMakerInstance is in the " + my_region + " region. You
will use the " + xgboost_container + " container for your SageMaker
endpoint.")lables = data.pop('Prediction')
```

```
bucket_name  'your-s3-bucket-name' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME
FOR YOUR BUCKET
s3   boto3.resource('s3')
try:
    if  my_region == 'us-east-1':
      s3.create_bucket(Bucket=bucket_name)
    else:
      s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={
'LocationConstraint': my_region })
    print('S3 bucket created successfully')
except Exception as e:
    print('S3 error: ',e)


try:
  urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-
machine-learning-model-
sagemaker/bank_clean.27f01fbbdf43271788427f3682996ae29ceca05d.csv",
"bank_clean.csv")
  print('Success: downloaded bank_clean.csv.')
except Exception as e:
  print('Data load error: ',e)

try:
  model_data  =  pd.read_csv('./bank_clean.csv',index_col=0)
  print('Success: Data loaded into dataframe.')
except Exception as e:
   print(‘Data load error: ‘,e)


#print the predictions
print(classifier.predict(xtest))
#print the actual values
print(ytest.values)


# Splitting the Data into training data and test data
train_data, test_data  =  np.split(model_data.sample(frac=1, random_state=1729),
[int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)

# Train the ML model

pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)],
axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix,
'train/train.csv')).upload_file('train.csv')
s3_input_train =
sagemaker.inputs.TrainingInput(s3_data='s3://{}/{}/train'.format(bucket_name,
prefix), content_type='csv')
```

```
sess   = sagemaker.Session()
xgb   = sagemaker.estimator.Estimator(xgboost_container,role, instance_count=1,
  instance_type='ml.m4.xlarge',output_path='s3://{}/{}/output'.format(bucket_name,
  prefix),sagemaker_session=sess)
  xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=
  0.8,silent=0,objective='binary:logistic',num_round=100)
  xgb.fit({'train': s3_input_train})



  # Deploying the model

  xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')

  from sagemaker.serializers import CSVSerializer
  test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load
  the datainto an array
  xgb_predictor.serializer = CSVSerializer() # set the serializer type
  predictions = xgb_predictor.predict(test_data_array).decode('utf-8') #
  predict!predictions_array = np.fromstring(predictions[1:], sep=',') #
  and turn the prediction into an array
  print(predictions_array.shape)

  # Evaluate the model


  cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array),
  rownames=['Observed'], colnames=['Predicted'])
  tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p =
  (tp+tn)/(tp+tn+fp+fn)*100
  print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
  print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase", "Purchase"))
  print("Observed")
  print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Purchase",
  tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
  print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Purchase",
  fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))
```

OUTPUT:

```
In [7]: xgb.fit({'train': s3_input_train})

[93]#011train-error:0.095314
[01:15:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24 extra nodes, 30 pruned nodes, max_depth=5
[94]#011train-error:0.095314
[01:15:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 24 pruned nodes, max_depth=3
[95]#011train-error:0.095314
[01:15:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12 extra nodes, 30 pruned nodes, max_depth=5
[96]#011train-error:0.095279
[01:15:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18 extra nodes, 12 pruned nodes, max_depth=5
[97]#011train-error:0.094828
[01:15:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 22 pruned nodes, max_depth=2
[98]#011train-error:0.094863
[01:15:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 12 pruned nodes, max_depth=5
[99]#011train-error:0.094759


2021-01-09 01:16:19 Uploading - Uploading generated training model
2021-01-09 01:16:19 Completed - Training job completed
Training seconds: 48
Billable seconds: 48
```

```
In [11]: cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array), rownames=['Observed'], colnames=['Pred
         tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)*100
         print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
         print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase", "Purchase"))
         print("Observed")
         print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Purchase", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
         print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Purchase", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))

Overall Classification Rate: 89.5%

Predicted      No Purchase    Purchase
Observed
No Purchase    90% (10785)    35% (151)
Purchase        10% (1143)     65% (278)
```

INFERENCE:

The Machine learning model trained can classify Purchase with 89.5% accuracy.

RESULT:

The Sagemaker, a cloud machine learning platform was successfully studied and implemented amazon toolkit with real time data set to develop a machine learning model.

EX.NO : 10	MINI PROJECT
DATE:	

ABSTRACT

Weather forecasting is the use of science and technology to predict the condition of the weather for a given area. It is one of the most difficult issues the world over. This project aims to estimate the weather by utilizing predictive analysis. For this reason, analysis of various data mining procedures is needed before apply. This paper introduces a classifier approach for prediction of weather condition and shows how Naive Bayes and Chi square algorithm can be utilized for classification purpose. This system is a web application with effective graphical User Interface. User will login to the system utilizing his user ID and password. User will enter some information such as current outlook, temperature, humidity and wind condition. This system will take this parameter and predict weather after analyzing the input information with the information in database. Consequently two basic functions to be specific classification (training) and prediction (testing) will be performed. The outcomes demonstrated that these data mining procedures can be sufficient for weather forecasting.

General Terms

1. Data Mining
2. Classification
3. Prediction.

Keywords

1. Chi square
2. Classification
3. Naïve Bayes
4. Prediction
5. Weather Forecasting

1.INTRODUCTION

Weather forecasting has been a standout amongst the most experimentally and technologically troublesome issues over the world in the most recent century. Environmental change has been looking for a great deal of consideration since a long time because of the sudden changes that happen.

- There are several limitations in better execution of weather forecasting thus it ends up hard predicting weather here and now with effectiveness. Weather forecasting assumes a significant role in meteorology. To makes an exact prediction is one of the significant troubles standing up to meteorologist wherever all through the world.
- Weather warnings are vital in light of the fact that they are utilized to ensure life and property. Forecasts dependent on Temperature, Outlook, Humidity and Wind are important to farming, and along these lines to traders inside product markets. Temperature forecasts are utilized by utility companies to assess request over coming days.
- Since outdoor activities are seriously reduced by substantial rain, snow and wind chill, estimates can be utilized to design activity around these occasions, and to prepare and survive them. Without precise weather forecasts individuals may end up in hazardous circumstances as they were unprepared for and end up harmed or worse.

The difficulties of weather forecasting, among others, are learning weather representation utilizing an enormous volume of weather dataset. For this purpose, analysis of different data mining procedure is performed.

- Data mining techniques enables users to analyze data from a wide range of dimensions or angles, classify it, and condense the connections recognized. Some fundamental terms related to Data Mining are: Classification, Learning and Prediction. Classification is a data mining (machine learning) method used to predict aggregate participation for information cases. For instance, classification can be utilized to predict whether the weather on a specific day will be “sunny”, “rainy” or “cloudy” .
- Learning refers to training and mapping contribution to yield information. It tends to be performed in two different ways: Supervised and Unsupervised learning. A supervised learning algorithm analyzes the training data and produces a derived capacity utilizing Classifier. In machine learning, unsupervised learning alludes to the issue of trying to hidden structure in unlabeled information. Since the precedents given to the learner are unlabeled, there is no mistake or reward signal to assess a potential solution.
- the experimental result and analyses it and last segment is committed to the conclusion.

2.RELATED WORK

In the most recent decade, numerous significant efforts to solve weather forecasting issue utilizing statistical modeling including machine learning systems have been reported with successful results. Different Methods has been utilized in Weather Prediction System, for example, neural network-based algorithm utilizing Back Propagation Neural Network (BPN) and Hopfield Network, Recurrence Neural Network (RNN), Conditional Restricted Boltzmann Machine (CRBM), and Convolutional Network (CN) models, Artificial Neural Network and Decision tree Algorithms, predictive analysis in Apache Hadoop Framework utilizing Naive Bayes Algorithm.

2.1 BPN and Hopfield Network

- In this work Back Propagation Neural (BPN) Network is utilized for initial modeling. The outcomes acquired by BPN model are sustained to a Hopfield Network. In BPN, the info and yield layer comprises of 3 neurons where as the hidden layer has 5 neurons and Hopfield Network display work with the assistance of training data set. The system must perform Temperature or Wind Speed or Humidity flow with the end goal to establish equilibrium. This procedure will proceed iteratively and in every iteration bias and weight esteems should be updated until it converges.

2.2 RNN, CRBM and CN models

- The goal of this work is to investigate the capability of profound learning technique for weather forecasting. The investigations, on deep networks, on energy-based models have progressed toward becoming establishments for the emerging deep learning as deep architecture generative models in the most recent decade. Three climate estimating models will be investigated in this examination which are in particular: (i) Recurrence Neural Network (RNN),
  (ii) Conditional Restricted Boltzmann Machine (CRBM), and
  (iii) Convolutional Network (CN).
- Every one of these models will be prepared and tried utilizing the predetermined weather dataset. Parameter learning algorithm for each model, for instance: gradient descent for CRBM and CN, is executed to gain testing error below the predetermined threshold value and compared with the prominent time series forecasting models for example, Recurrent NN.

2.3 ANN and Decision Tree

- Artificial Neural Networks (ANN) and Decision Trees (DT) were utilized to analyze meteorological data, accumulated with the end goal to develop classification rules for the Application of Data Mining Techniques in Weather Prediction. There are three fundamental components of a neuron model, which

are, (i) an arrangement of synapses, interfacing links, every one of which is considered by a weight/strength of its own (ii) an adder, for summing the info signals, weighted by particular neuron's neural connections (iii) an activation function, for restricting the amplitude of neuron's yield. The MLP network is prepared through the back-propagation learning algorithm. The Prediction is performed through Decision tree.

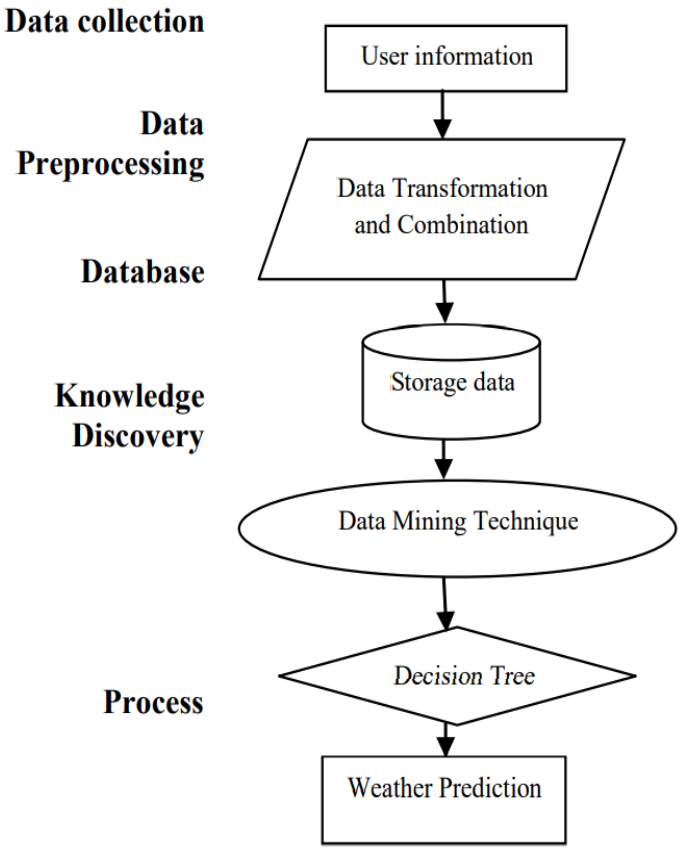
No	Attributes	Class
1	Outlook	Sunny,overcast,rainy
2	Temperature	High, Mild, Cool
3	Humidity	High, Normal
4	Windy	True, False

2.4 Naive Bayes algorithm utilizing Hadoop

- The project aims to forecast the chances of rainfall by utilizing predictive analysis in Hadoop. Predictive analysis models catch connections among numerous elements in a data set to evaluate chance with a specific arrangement of conditions to allocate a score or a weight. Here, Apache Hadoop Framework and Map Reduce Framework are utilized to decrease the data and Naïve Bayes Algorithm is utilized in classification and prediction. Naïve Bayes Algorithm is classification technique based on Bayes Theorem.
- Naïve Bayes is anything but difficult to assemble and especially valuable for expansive datasets. It is exceptionally utilized in different looks into which contains substantial datasets, for example, Disease prediction. Hadoop is open source programming and it is accustomed to storing large data set in a distributed computing environment, Hadoop makes it conceivable to run applications on system with several hardware nodes.
- The Hadoop Distributed File System (HDFS) is like the Google File System (GFS) and it utilizes large cluster of data and it gives appropriated distributed file system, fault- tolerant way.

3.METHODOLOGY

In this paper, the system predicts the future weather conditions based on current weather data. The data mining techniques namely Chi square test and Naïve Base statisticsare applied on the dataset to extract the useful information from the dataset. The System Methodology shows in fig. 1:



3. 1 Data Collection and Preprocessing

The initial stage in data mining process is data collection and preprocessing. The crucial stage is data preprocessing, because only valid data will yield accurate output. The data is used in this project collected from users. Though the data set contained many attributes, data preprocessing step considered only the relevant information, ignoring the rest. Then data transformation performed, into a format, which is suitable for Data mining. Four Attributes are used to identify the Weather Forecasting .They are Shown in a table below:

- It is to find out the Class Level of Weather Forecast where, Class Levels are Good or Bad.

3. 2 Database

The Transformed dataset is store in database that is collected from user. So, there is no prviously stored store is in use. After real time data collection, Data mining techniques applied to predict weather condition.

3.3 Data Mining Technique

In this work, data classification is performed using two data mining technique: Chi square test and Naive Bays Statistics. The data which have to be classified is called training dataset, is fixed. By using this data with testing data, Weather Forecast will be possible. The algorithm of chi square and naïve bays finds relationships between the values of the predictors and the values of the target. The model learns from the training set and that knowledge is used as test data to predict in the scoringse

3.3.1 Chi Square Algorithm

Chi Square Algorithm is a predictive technique used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories. The Equation is as follows: $\chi c\ 2 = \Sigma(Oi\text{-}Ei)\ 2\ /Ei$ where, The subscript "c" is the degrees of freedom, "O" is observed value and "E" is expected value. A chi square ($X\ 2$) statistic is used to investigate whether distributions of categorical variables differ from one another. In our project we use chi square statistic to determine the best attribute of weather forecast.

3.3.2 Naïve Bayes Algorithm

Naïve Bayes Algorithm is a classification technique based on Bayes Theorem. Naïve Bayes is easy to build and very much useful for large datasets. By using the Naïve Bayes equation we can find the future probability [12].The Equation is as follows:

$$\mathbf{P(c|x)=\frac{P(x|c)*P(c)}{P(x)}}$$

Where, (c|x) is future probability of class(c, target), P(c) is the prior probability of the class, P(x|c) is the likelihood which is the probability of predictor given class, P(x) is the prior probability of predictor.

The condition of predicting weather of our project is as follows:

Class:

C1: Weather Forecasting = 'Good'

C2: Weather Forecasting = 'Bad'.

To find the class, Ci that maximizes P (X| Ci)*P (Ci)compute:
P(Weather = Good | x) ∝ P(Weather=Good) ·

[P(O = s | Weather = Good) · P(T = c | Weather = Good) · P(H = h | Weather = Good) . P(W = t | Weather = Good)]

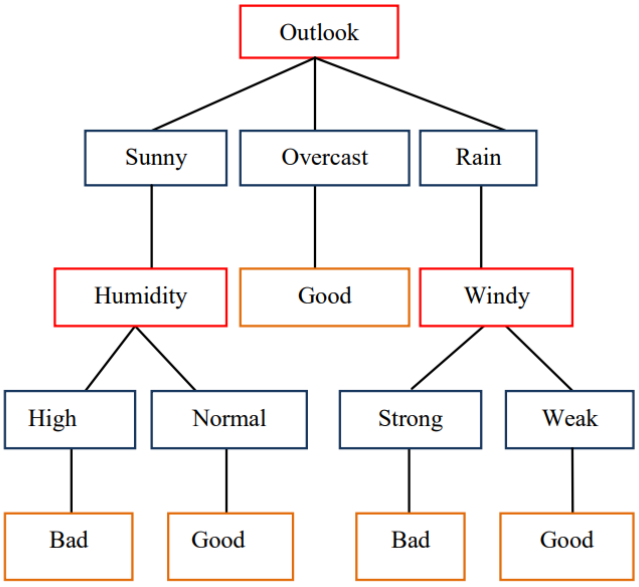
P(Weather =Bad | x) ∝ P(Weather=Bad) · [P(O = s | Weather = Bad) · P(T = c | Weather = Bad) · P(H = h | Weather=Bad) · P(W = t | Weather =Bad]

IF P(Weather= Good| X) < P(Weather = Bad | X), so classify X as Weather= Bad Otherwise, Classify X as Weather=Good.

Thus using the above probability prediction of the future chances of weather good or bad will be easy.

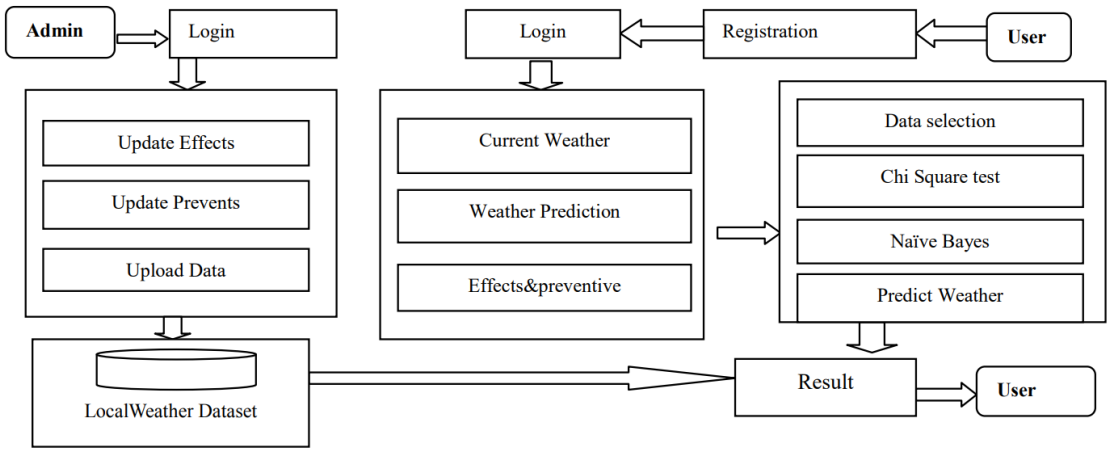
3.3.3 Decision Tree

The decision Tree generated from training data is helpful in making prediction. Construction of the decision tree is done by selecting the best possible attribute that will be able to split set of samples in most effective manner. The decision tree for this proposed system is figured below in fig. 2:



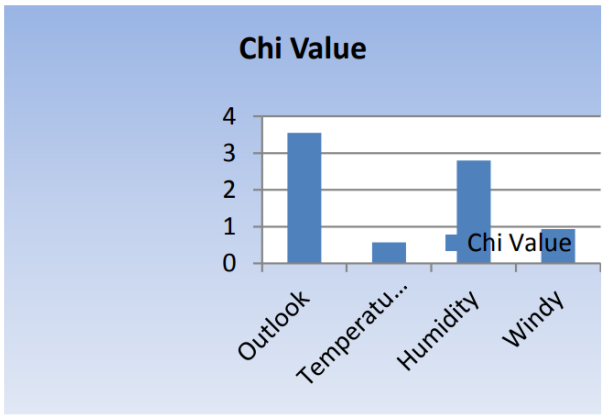
4.DESIGN AND ANALYSIS

This topic incorporates the methodology within the design of the system. This system analyzes and measures weather data. The architecture is given in fig.3, clarifies the working model of the project. The Architecture characterizes the behavior, structure and perspectives of our system.

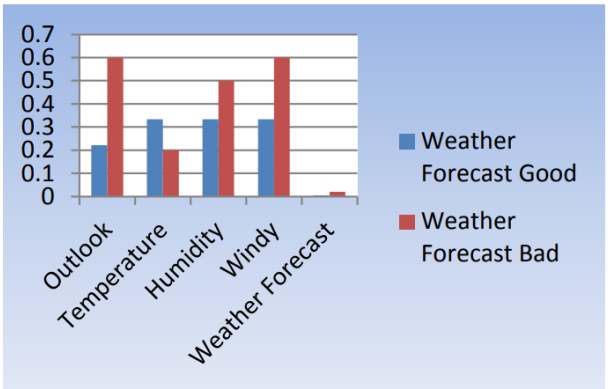


5.EXPERIMENTAL RESULT AND ANALYSIS

The analysis and prediction of weather forecast are implemented using Java Language and using by tool of Eclipse and all data are stored by MySql server. Chi square test summarizes the difference between our data and our independence hypothesis.



It depicts the weather forecast class bad is greater than the weather forecast class good.



Accuracy

- In this Project, Accuracy is the general rightness of the model and is determined as the whole of correct classifications separated by the aggregate number of classifications.

  Accuracy = (TP + TN)/ (TP + TN + FP + FN).

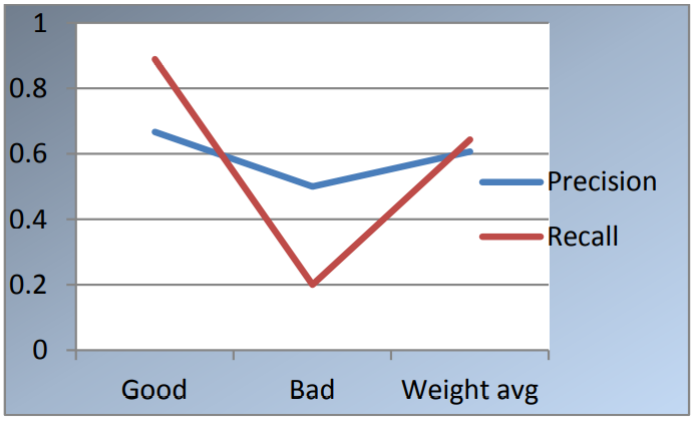
- Precision is a proportion of the accuracy gave that an explicit class has been predicted. It is characterized as

  Precision = TP/ (TP + FP)

- Where, TP and FP are the numbers of true positive and false positive predictions for the considered class.
- Recall is a proportion of the capacity of a prediction model to choose examples of a specific class from a data set. It is likewise called sensitivity, and compares to the True positive rate.

  Recall = Sensitivity = TP / (TP + FN)

- Where, TP and FN are the numbers of true positive and false negative predictions for the considered class.
- The System is computed and demonstrated utilizing the figure 6. In this figure the X axis contains the methods implemented and the Y axis demonstrates the percentage accuracy of the system. As indicated by the acquired execution the proposed



6. CONCLUSION

This paper works with mix of Naïve Bayes and Chi Square algorithm to predict weather condition. The constant information i.e. time-series data is assembled and analysis is performed on this dataset utilizing an interface named Weather Prediction System, developed utilizing Java using Eclipse tools. This framework arranges the given information into various classifications and furthermore predicts the risk of the weather prediction of obscure example is given as an input. The system can be filled in as training tool for Meteorology Students. This methodology can decide the nonlinear relationship that exists between the historical data (temperature, wind speed, humidity, and so forth.,) provided to the system during the training phase and on that premise, make a prediction of what the weather would be in future. The Future work of this project is to incorporate more attribute of weather condition to predict and to work with other classification algorithm to become more accurate in prediction.

7.REFERENCES

[1] Dhanashree S. Medhekar, Mayur P. Bote, Shruti D. Deshmukh, “Heart Disease Prediction System using Naive Bayes”, INTERNATIONAL JOURNAL OF ENHANCED RESEARCH IN SCIENCE TECHNOLOGY & ENGINEERING, VOL. 2, ISSUE 3, MARCH.-2013, pp 1-5

[2] Mehrnoosh Torabi, Sattar Hashemi, “A Data Mining Paradigm to Forecast Weather”, The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012),IEEE, pp 579-584.

[3] Amruta A. Taksande, P. S. Mohod, “Applications of Data Mining in Weather Forecasting Using Frequent Pattern Growth Algorithm”, International Journal of Science and Research (IJSR), Volume 4 Issue 6, June 2015, pp 3048-3051

[4] Mr. Sunil Navadia, Mr. Jobin Thomas, Mr. Pintukumar Yadav, Ms. Shakila Shaikh, “Weather Prediction: A novel approach for measuring and analyzing weather data”, International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), (I-SMAC 2017), IEEE, pp 414-417

[5] Ghosh et al., "Weather Data Mining using Artificial Neural Network," 2011 IEEE Recent Advances in Intelligent Computational Systems, Trivandrum, 2011, pp. 192-195.