Artefact

Thesis is done using Python Programming language as it supports multiple libraries in support if machine learning. Python is a very powerful high-level and structured language with simple syntax and flexible semantics. Compared to other languages Python is very simple. Coding with Python is very convenient and requires less effort as there are many inbuilt libraries.

It is derived or developed from the language that that people use to interact in day to day life. Anyone can attempt to understand this Python code without the pressure of digging loads of computer code. Python employs indentation instead of braces to the code looks well organised and structured. Even with a smaller code good efficiency can be achieved. Python is open source and it supports many libraries.

# 1 Python Code Description

## Library Dependencies

The main libraries used to carry out the research are:

### 1.1.1 NumPy

NumPy is python library used for scientific computation.

import numpy as np

### 1.1.2 Pandas

This module is used for data analysis. It should be manually installed as it is not bundled with Python. It gives descriptive and adaptable intelligent mechanisms such as information structures that enables data management and its control simple and clear across various objects. One such structure in data frame.

import pandas as pd

### 1.1.3 Matplotlib.plotly

This is python library used for visualization of various features to understand data in depth.

import matplotlib.pyplot as plt

### 1.1.4 OS

OS python library supports numerous operating system reliant functionalities like listdir () to list all files in directory.

import os

### 1.1.5 Keras

Keras is a Python library built on top of the TensorFlow framework.it is flexible and convenient to learn. Keras is created to emphasize o recognizing the methods of deep learning.

For quick prototyping and guided research keras is used. Keras has a simple interface so it is easy to construct. The errors are clearly mentioned. It has few limitations as it combines configurable structure squares custom structure squares built can be used to articulate new ideas for research.

from keras import models

from keras.layers import Dense, Dropout

from keras.optimizers import Adam

from keras import losses

from keras import metrics

from keras.models import Sequential

### 1.1.6 Train and Test

Split arrays or matrices into random train and test subsets which is imported from sklearn

from sklearn.model\_selection import train\_test\_split

### 1.1.7 Evaluation

Confusion Matrix, accuracy and classification report is needed to evaluate the models. It is imported from the sklearn.metrics.

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score

### 1.1.8 Other Libraries

Python’s Itertool is a module that provides various functions that work on iterators to produce complex iterators.

**Pathlib module** in Python provides various classes representing file system paths with semantics appropriate for different operating systems.

import itertools

from pathlib import Path

## 1.2 Functions for Machine learning models

Sklearn is a commonly used library of python for building traditional machine learning models. For deep leaning keras and tensorflow are commonly used

1.2.1 Read files, split in training, and test subsets Data is read in pandas data frame from local drive and then split into training and testing

x = data.drop('class',axis =1)

y = data['class']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, shuffle=True)

train\_data = pd.concat([x\_train, y\_train], axis=1, join='inner')

test\_data = pd.concat([x\_test, y\_test], axis=1, join='inner')

### 1.2.1 Artificial Neural Network

Data is scaled using Sklearn pre-processing for making sure that there is no bias due to different scales of input data. To scaling data, MinMaxScaler function of Sklearn pre-processing library of python is used. After data is scaled it is split into training and test. Model is build using keras library.

X and Y values are initialised. Y consists of dependent variables which can be numerical or categorical and X consists of dependent variables. Here X has the concentration levels of gases and Y has Class.

Epochs specifies number of times dataset should be iterated. So, 5 is the value given for Epochs. Seed makes components of algorithm to be dependent on random number generator seed for randomization. The seed value given is 121. Several types of activations are supported for hidden layers.

from keras.utils import to\_categorical

y\_train\_binary = to\_categorical(y\_train)

y\_test\_binary = to\_categorical(y\_test)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x\_std\_train = scaler.fit\_transform(x\_train)

x\_std\_test = scaler.fit\_transform(x\_test)

model =Sequential()

model.add(Dense(32, input\_dim =6, activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(512, activation='relu'))

model.add(Dense(5, activation='sigmoid'))

model.compile(

optimizer=Adam(lr =0.0001),

loss= 'categorical\_crossentropy',

metrics=['accuracy']

)

model.fit(

x\_std\_train,

y\_train\_binary,

epochs=5,

validation\_data=(x\_std\_test, y\_test\_binary),

shuffle=True

)

y\_pred = model.predict(x\_std\_test, batch\_size=10, verbose=0)

y\_predict = []

for i in range(0, len(y\_pred)):

y\_predict.append(int(np.argmax(y\_pred[i])))

len(y\_predict)

y\_true = []

for i in range(0, len(y\_test\_binary)):

y\_true.append(int(np.argmax(y\_test\_binary[i])))

len(y\_true)

For evaluation confusion matrix, accuracy and classification report is used.

def plot\_confusion\_matrix(cm, classes,title='Confusion matrix'):

plt.imshow(cm, interpolation='nearest', cmap='YlGnBu')

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.ylabel('True label')

plt.xlabel('Predicted label')

plt.tight\_layout()

plt.show()

print(classification\_report(y\_true=y\_true,y\_pred=y\_predict))

acc = accuracy\_score(y\_true=y\_test, y\_pred= y\_predict)

print ('Accuracy : ',acc\*100)

cm = confusion\_matrix(y\_true=y\_true, y\_pred=y\_predict)

plot\_confusion\_matrix(cm, cm\_plot\_labels, title='Confusion Matrix')

### 1.2.2 Logistic Regression

Sklearn has library called LogisticRegression which is used to perform logistic regression on train data and test data.

LogisticRegression is imported from sklearn.linear\_model. The data is divided into training and test sets. C value given in 1.0 which is the Inverse of regularization strength and it must be a positive float.

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression(C=1.0)

logreg.fit(x\_train,y\_train)

y\_pred = logreg.predict(x\_test)

For evaluation of Logistic Regression confusion matrix, accuracy and classification report is used.

acc\_lr = accuracy\_score(y\_true=y\_test, y\_pred= y\_pred)

print("Overall accuracy of LR model using test-set is : %f" %(acc\_lr\*100))

print(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_true=y\_test, y\_pred=y\_pred)

plot\_confusion\_matrix(cm, cm\_plot\_labels, title='Confusion Matrix')

### 1.2.3 Support Vector Machine

Sklearn has library called SVC which is used to perform Support Vector Classification on train data and test data.

For implementing Support Vector Machine in Python standard libraries are imported and Scikit-Learn’s support vector classifier is used to train a Support Vector Machine model on the dataset. Here, linear kernel is used to fit Support Vector Machine.

from sklearn.svm import SVC

from sklearn.metrics import classification\_report,confusion\_matrix,accuracy\_score

model\_svm = SVC(C=1e20)

model\_svm.fit(x\_train, y\_train)

predictions\_svm = model\_svm.predict(x\_test)

For evaluation of Support Vector Classification confusion matrix, accuracy and classification report is used.

predictions\_svm = model\_svm.predict(x\_test)

acc\_svm = accuracy\_score(y\_true=y\_test, y\_pred= predictions\_svm)

print ("Overall accuracy of SVM model using test-set is : %f" %(acc\_svm\*100))

print(classification\_report(y\_test,predictions\_svm))

### 1.2.4 Naïve Bayes

Sklearn has library called GaussianNB which is used to perform Naïve Bayes on train data and test data.

Gaussian Naïve Bayes is used to predict the air quality. The dataset is divided into train and test test. GaussianNB is imported from sklearn.naive\_bayes and it does not require initialisation of many parameters.

from sklearn.naive\_bayes import GaussianNB

nb\_model = GaussianNB()

nb\_model.fit(x\_train,y\_train)

y\_pred = nb\_model.predict(x\_test)

For evaluation of Naïve Bayes Classification confusion matrix, accuracy and classification report is used.

acc\_nb = accuracy\_score(y\_true=y\_test, y\_pred= y\_pred)

print("Overall accuracy of NB model using test-set is : %f" %(acc\_nb\*100))

print(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_true=y\_test, y\_pred=y\_pred)

plot\_confusion\_matrix(cm, cm\_plot\_labels, title='Confusion Matrix')

### 1.2.5 Random Forest Classifier

Sklearn has library called RandomForestClassifier which is used to perform Naïve Bayes on train data and test data.

The data is divided into training and test sets. The most important parameter in Random Forest Classifier is n\_estimators which defines the number of trees in the random forest model. N\_estimator value given is 600.

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=600)

rfc = rfc.fit(x\_train,y\_train)

For evaluation of Random Forest Classifier confusion matrix, accuracy and classification report is used.

predictions\_rf = rfc.predict(x\_test)

acc\_rf = accuracy\_score(y\_true=y\_test, y\_pred= predictions\_rf)

print("Overall accuracy of RF model using test-set is : %f" %(acc\_rf\*100))

print(classification\_report(y\_test,predictions\_rf))

cm = confusion\_matrix(y\_true=y\_test, y\_pred=predictions\_rf)

plot\_confusion\_matrix(cm, cm\_plot\_labels, title='Confusion Matrix')

## 1.3 Model Accuracy Plot

Seaborn is a Python library created for enhanced data visualization. It’s a very timely and relevant tool for data professionals working today precisely because effective data visualization – and communication in general – is a particularly essential skill.

import seaborn as sns

li\_x = ['SVM', 'LR', 'NB','ANN', 'RF']

li\_y = [acc\_svm, acc\_lr,acc\_nb,acc,acc\_rf]

print(li\_y)

sns.barplot(x=li\_x, y=li\_y)

## 1.4 To find next three months air quality

Auto regression model is used to find the air quality for next three months by taking the average value of every month. The results are then saved into a data.csv file which will be used for testing.

From the dataset monthly average value for all the gases is found and then Auto Regression is used to find next three months air quality. The monthly average value which is found for gases is given as the value of X. Now this is divided into train and test set.

start = 0

end = 30

l = test\_df.shape[0]

l = l/30

l = round(l)

l = int(l)

NO = []

NO2 = []

NOx = []

NH3 = []

CO = []

SO2 = []

for i in range(0,l):

a = test\_df.loc[start:end , ['NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2']]

NO.append(a['NO'].mean())

NO2.append(a['NO2'].mean())

NOx.append(a['NOx'].mean())

NH3.append(a['NH3'].mean())

CO.append(a['CO'].mean())

SO2.append(a['SO2'].mean())

start += 30

end += 30

datas = {'NO': NO,'NO2': NO2,'NOx': NOx,'NH3': NH3,‘CO': CO,'SO2': SO2}

df = pd.DataFrame(datas, columns = ['NO', 'NO2','NOx','NH3','CO','SO2'])

print (df)

Auto Regression is imported from statsmodels.tsa.ae\_model

from statsmodels.tsa.ar\_model import AR

def df\_cluster(data):

X = data

train, test = X[0:-1], X[-1:]

l = len(train) + 2

model = AR(X)

model\_fit = model.fit()

predictions = model\_fit.predict(start=len(train), end=l, dynamic=True)

return predictions

no = df\_cluster(df['NO'])

no2 = df\_cluster(df['NO2'])

nox = df\_cluster(df['NOx'])

nh3 = df\_cluster(df['NH3'])

co = df\_cluster(df['CO'])

so2 = df\_cluster(df['SO2'])

datas = {'NO': no, 'NO2': no2,'NOx': nox,'NH3': nh3,'CO': co,'SO2': so2}

df = pd.DataFrame(datas, columns = ['NO', 'NO2','NOx','NH3','CO','SO2'])

print (df)

df.to\_csv('input/data.csv',index=False)

## 1.4 To test the Air quality Index prediction

test\_data = pd.read\_csv('input/data.csv')

def predict\_result(data, model):

result = model.predict\_classes(data)

return result

results = predict\_result(data=test\_data, model= trained\_model)

print('Predicted results for given input is : ',results)

for res in results:

print(class\_names[res])