# Configuration Manual

MSc Research Project
Data Analytic

Sakshi Khanvilkar
Student ID: x22117776

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | | | |
|---|---|---|---|
| **Student Name:** | Sakshi Kishor Khanvilkar | | |
| **Student ID:** | X22117776 | | |
| **Programme:** | M.Sc. Data Analytics | **Year:** | 2023-2024 |
| **Module:** | M.Sc. Research Project | | |
| **Lecturer:** | Abid Yaqoob | | |
| **Submission Due Date:** | 31/01/2024 | | |
| **Project Title:** | Sentimental analysis of English Tweets on Demonetization and Analysing the Impact of Demonetization on Digital wallet in India | | |
| **Word Count:** | 2964 | **Page Count:** | 13 |

| | |
|---|---|
| **Signature:** | Sakshi Kishor Khanvilkar |
| **Date:** | 31/01/2024 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Sakshi Khanvilkar
Student ID: x22117776

## 1 Introduction

This configuration manual highlights the hardware and software necessity for doing research on "*Sentimental analysis of English Tweets on Demonetization and Analysing the Impact of Demonetization on Digital wallet in India"*. It demonstrates step by step direction for correctly reproducing the test, allowing simple replication and full outcome analysis. The following configuration manual provides full explanation for each step, as well as snippets of code and as well as significant output.

## 2 Data Collection

The dataset used for the sentimental analysis contains 14,490 tweets records and was collected from Kaggle, an open-source platform[1]. The tweet dataset confines the tweet for November 2016 and April 2017. On other hand, the dataset for time series analysis is taken from the Reserve Bank of India (RBI) database and covers the period from April 2004 to October 2019[2].

## 3 Hardware Configuration

### 3.1 Local Machine Hardware Specification

Table 1 shows the configuration of the laptop used to conduct the study.

| Hardware | Specification |
|---|---|
| Processor | 11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz |
| Installed RAM | 8.00 GB (7.70 GB usable) |
| System Type | 64-bit operating system, x64-based processor |
| Operating System | Windows 11 |

**Table 1: Laptop Configuration.**

### 3.2 Google Collab Hardware Specification

Table 2 shows the configuration of the Google Collab used to conduct the study.

| Hardware | Specification |
|---|---|
| RAM | 12.7 GB |
| Disk | 107.7 GB |

**Table 2: Google Collab Configuration**

[1] https://www.kaggle.com/datasets/arathee2/demonetization-in-india-twitter-data
[2] https://dbie.rbi.org.in/#/dbie/statistics/Payment%20Systems

## 4 Software Configuration

Table 3 shows the software Configuration used for this Project.

| Software | Specification |
|---|---|
| Programming Language | Python version 3.10.12 |
| IDE | Google Collab |
| Libraries | TensorFlow, Keras, Matplotlib, NumPy, Pandas, Stats model |

**Table 3: Software configuration.**

## 5 Project Implementation

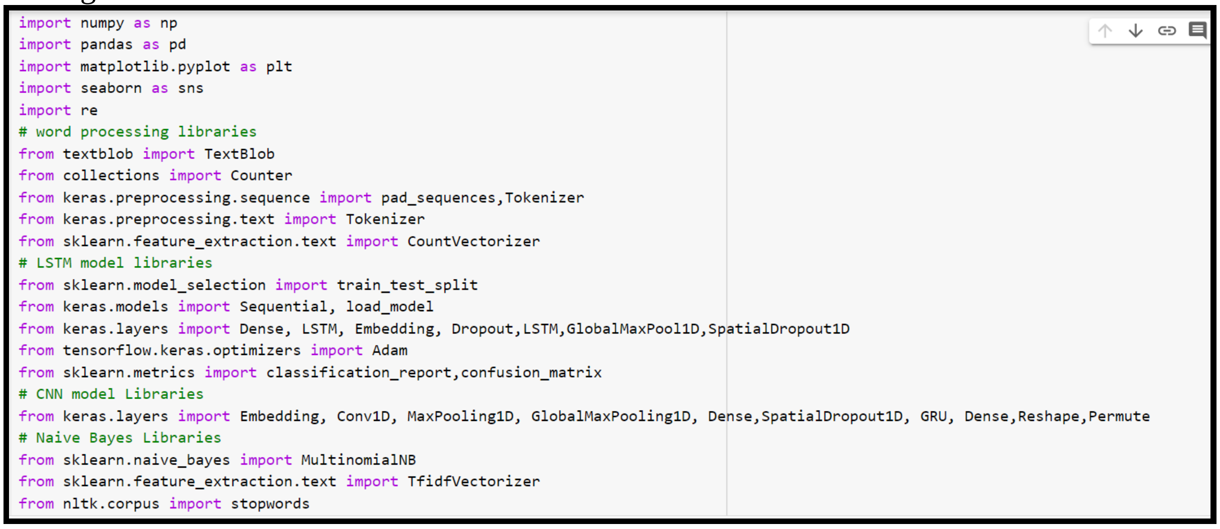
This Section demonstrates a detailed description of the steps used to achieve to accomplish the study's objective from the beginning to end.

### 5.1 Sentimental analysis on Demonetization tweets

This section will explain the step use to execute the sentimental analysis.

#### 5.1.1 Importing Libraries

Into Google Collab I installed and imported necessary package and libraries to accomplish sentimental analysis. These libraries are used for processing the data and visualise critical finding.

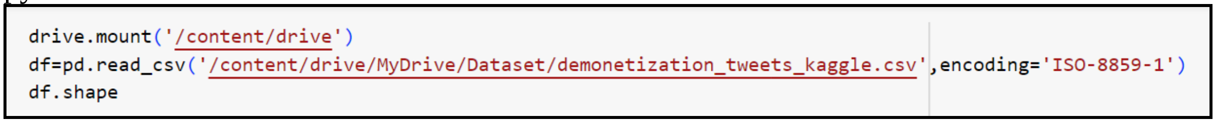


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
# word processing libraries
from textblob import TextBlob
from collections import Counter
from keras.preprocessing.sequence import pad_sequences,Tokenizer
from keras.preprocessing.text import Tokenizer
from sklearn.feature_extraction.text import CountVectorizer
# LSTM model libraries
from sklearn.model_selection import train_test_split
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, Embedding, Dropout,LSTM,GlobalMaxPool1D,SpatialDropout1D
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report,confusion_matrix
# CNN model Libraries
from keras.layers import Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dense,SpatialDropout1D, GRU, Dense,Reshape,Permute
# Naive Bayes Libraries
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
```

**Figure 1: Libraries.**

#### 5.1.2 Loading Dataset

The Following code loads dataset from google drive into google collab using drive library of python.

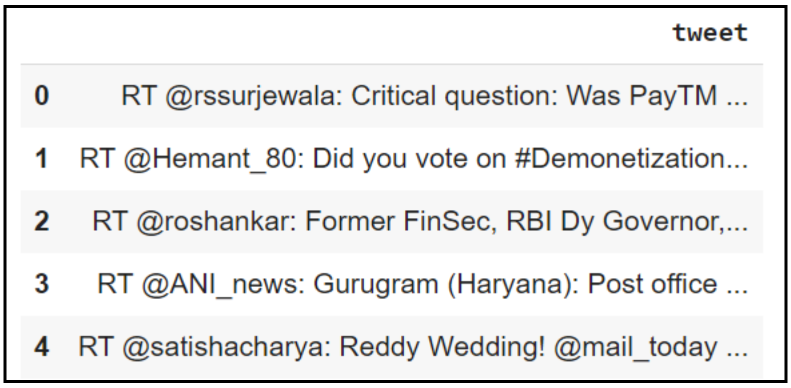


```
drive.mount('/content/drive')
df=pd.read_csv('/content/drive/MyDrive/Dataset/demonetization_tweets_kaggle.csv',encoding='ISO-8859-1')
df.shape
```

**Figure 2: Import Dataset.**

## 5.1.3 Data preprocessing

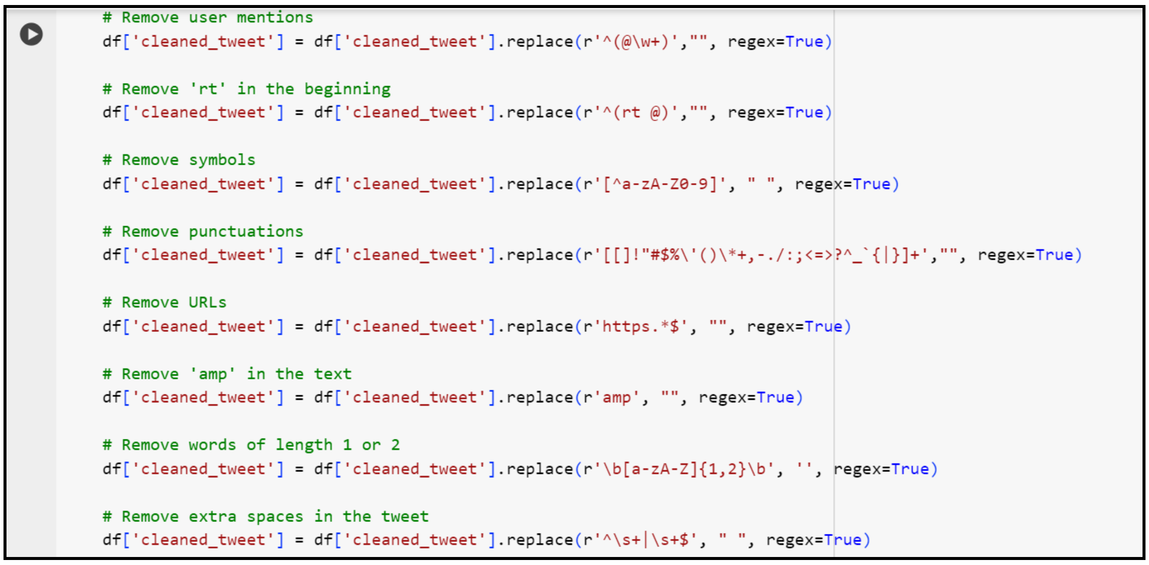
The dataset consists of many columns such as text, favourited, favouriteCount, relpytoSN, created, truncated and so on from all this column the necessary column is ‘text’ column. Next, I replace the column name from ‘text’ to ‘tweet’.



| | tweet |
|---|---|
| 0 | RT @rssurjewala: Critical question: Was PayTM ... |
| 1 | RT @Hemant_80: Did you vote on #Demonetization... |
| 2 | RT @roshankar: Former FinSec, RBI Dy Governor,... |
| 3 | RT @ANI_news: Gurugram (Haryana): Post office ... |
| 4 | RT @satishacharya: Reddy Wedding! @mail_today ... |

**Figure 3: Tweet column.**

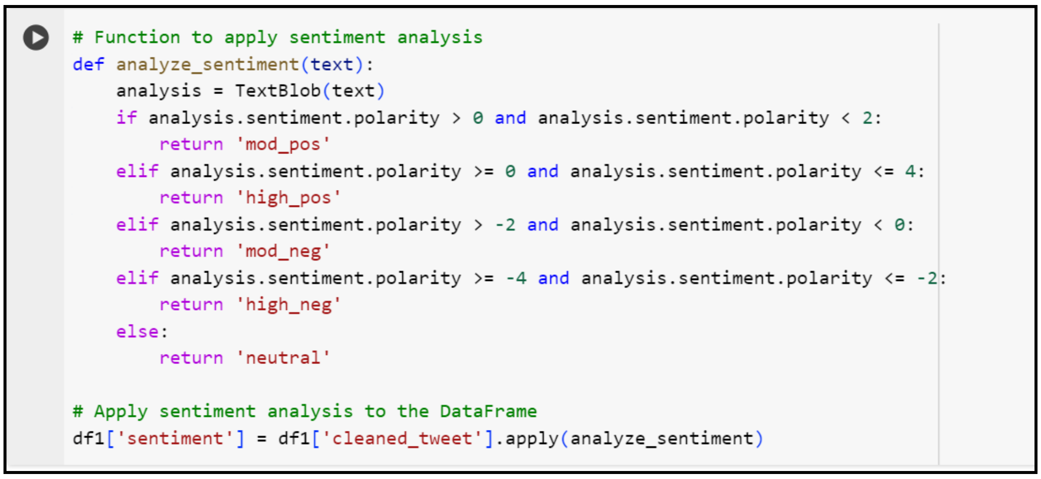
Further data processing using Regular Expression library in python was done on ‘tweet’ column which consist of URL, Hashtages, @, usernames, unwanted punctuation (Kannan Suseelan Unnithan, 2022).



```
# Remove user mentions
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'^(@\w+)',"", regex=True)

# Remove 'rt' in the beginning
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'^(rt @)',"", regex=True)

# Remove symbols
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'[^a-zA-Z0-9]', " ", regex=True)

# Remove punctuations
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'[[]!"#$%\'()\*+,-./:;<=>?^_`{|}]+',"", regex=True)

# Remove URLs
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'https.*$', "", regex=True)

# Remove 'amp' in the text
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'amp', "", regex=True)

# Remove words of length 1 or 2
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'\b[a-zA-Z]{1,2}\b', '', regex=True)

# Remove extra spaces in the tweet
df['cleaned_tweet'] = df['cleaned_tweet'].replace(r'^\s+|\s+$', " ", regex=True)
```

**Figure 4: Filtering the Tweet column.**

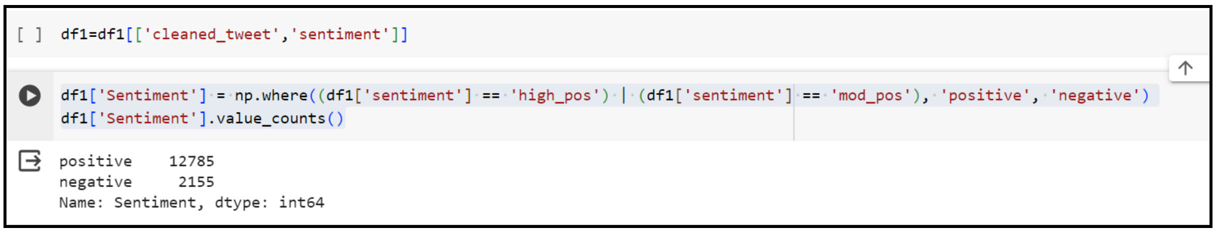
Next Sentimental Score was assigned to each tweet mentioned in ‘Cleaned_tweet’ column using TextBlob. The emotions are classified into categories such as ‘mod_pos’ stating moderately positive, ‘high_pos’ meaning High Positive and ‘high_neg’ and ‘neutral’ called as High positive and Neutral.



```
# Function to apply sentiment analysis
def analyze_sentiment(text):
    analysis = TextBlob(text)
    if analysis.sentiment.polarity > 0 and analysis.sentiment.polarity < 2:
        return 'mod_pos'
    elif analysis.sentiment.polarity >= 0 and analysis.sentiment.polarity <= 4:
        return 'high_pos'
    elif analysis.sentiment.polarity > -2 and analysis.sentiment.polarity < 0:
        return 'mod_neg'
    elif analysis.sentiment.polarity >= -4 and analysis.sentiment.polarity <= -2:
        return 'high_neg'
    else:
        return 'neutral'

# Apply sentiment analysis to the DataFrame
df1['sentiment'] = df1['cleaned_tweet'].apply(analyze_sentiment)
```

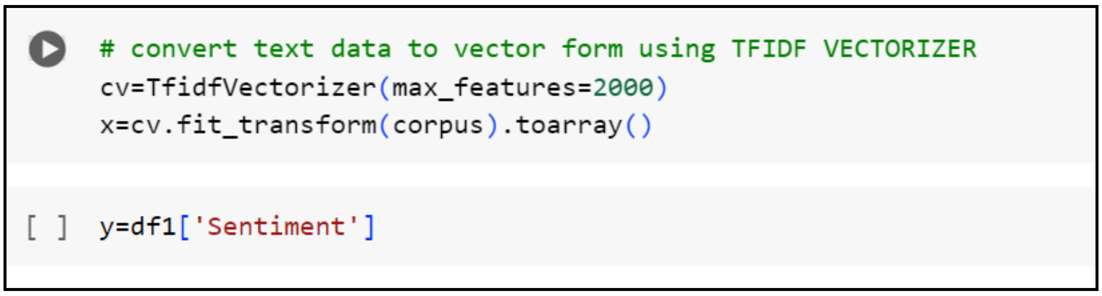
**Figure 5: Assigning labels.**

Further a new dataframe named as “df1” have been modified to concentrate on important columns that is “cleaned_tweet”, and “sentiments”. The Sentiments label is then reduced, were ‘high_pos’ and ‘mod_pos’ designated as ‘positive’ and the remaining as ‘negative’. as shown in Figure 6. Furthermore figure 7, the ‘cleaned_tweet’ column is transformed into a vectorized using TFID vectorizer. In the vectorized participation, the TFID vectorizer has been confirgured maximum of 2000 features that is words.



```
[ ]  df1=df1[['cleaned_tweet','sentiment']]

df1['Sentiment'] = np.where((df1['sentiment'] == 'high_pos') | (df1['sentiment'] == 'mod_pos'), 'positive', 'negative')
df1['Sentiment'].value_counts()

positive    12785
negative     2155
Name: Sentiment, dtype: int64
```

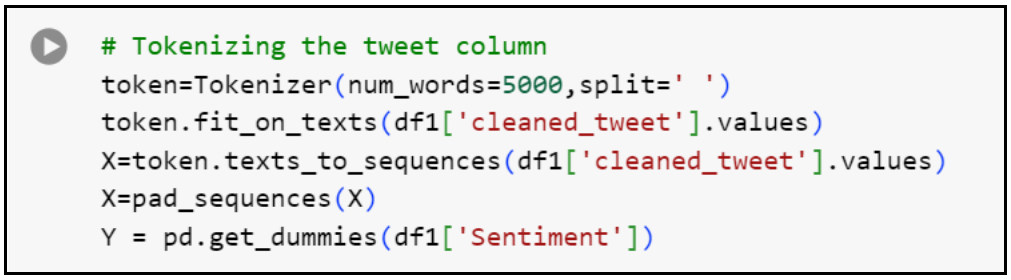
**Figure 6: Data Cleaning.**



```
# convert text data to vector form using TFIDF VECTORIZER
cv=TfidfVectorizer(max_features=2000)
x=cv.fit_transform(corpus).toarray()

[ ]  y=df1['Sentiment']
```

**Figure 7: Vectorizing Tweet Colum.**

The tweet text was tokenized using a Tokenizer, the resulting sequences are then padded to make sure that model input is uniform in length (Kumar Abhishek, 2020). Moreover, using one hot encoding dummy variables are created for the reduced sentence as mentioned in Figure 7.



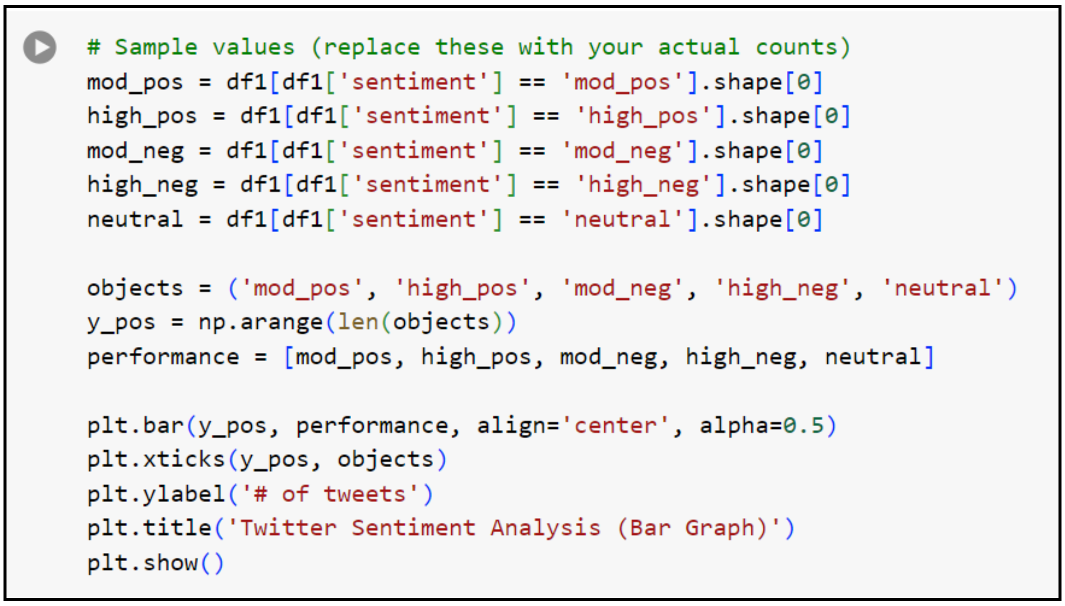
```python
# Tokenizing the tweet column
token=Tokenizer(num_words=5000,split=' ')
token.fit_on_texts(df1['cleaned_tweet'].values)
X=token.texts_to_sequences(df1['cleaned_tweet'].values)
X=pad_sequences(X)
Y = pd.get_dummies(df1['Sentiment'])
```

**Figure 7: Tokenizing the tweet Column.**

The TFID Vectorizer and Tokenizer are two distinct methods. The TFID Vectorizer, used in a conventional machine learning that converts text into dense matrix, highlighting word importance based on rate and reverse document frequency. This representation is suitable for machine learning model. Whereas tokenization is crucial step for deep learning models that is used for sequential data such as LSTM ands CNN. The goal of tokenizer is to convert the text into integer format (Ghulam Musa Raza, 2021).

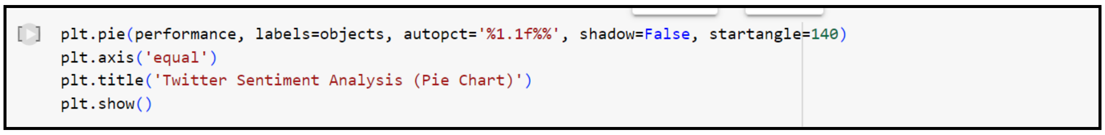
### 5.1.4 Exploratory Data Analysis

This Section depict the python code need to perform the EDA. It is useful for checking the distribution of the sentiments.



```python
# Sample values (replace these with your actual counts)
mod_pos = df1[df1['sentiment'] == 'mod_pos'].shape[0]
high_pos = df1[df1['sentiment'] == 'high_pos'].shape[0]
mod_neg = df1[df1['sentiment'] == 'mod_neg'].shape[0]
high_neg = df1[df1['sentiment'] == 'high_neg'].shape[0]
neutral = df1[df1['sentiment'] == 'neutral'].shape[0]

objects = ('mod_pos', 'high_pos', 'mod_neg', 'high_neg', 'neutral')
y_pos = np.arange(len(objects))
performance = [mod_pos, high_pos, mod_neg, high_neg, neutral]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('# of tweets')
plt.title('Twitter Sentiment Analysis (Bar Graph)')
plt.show()
```

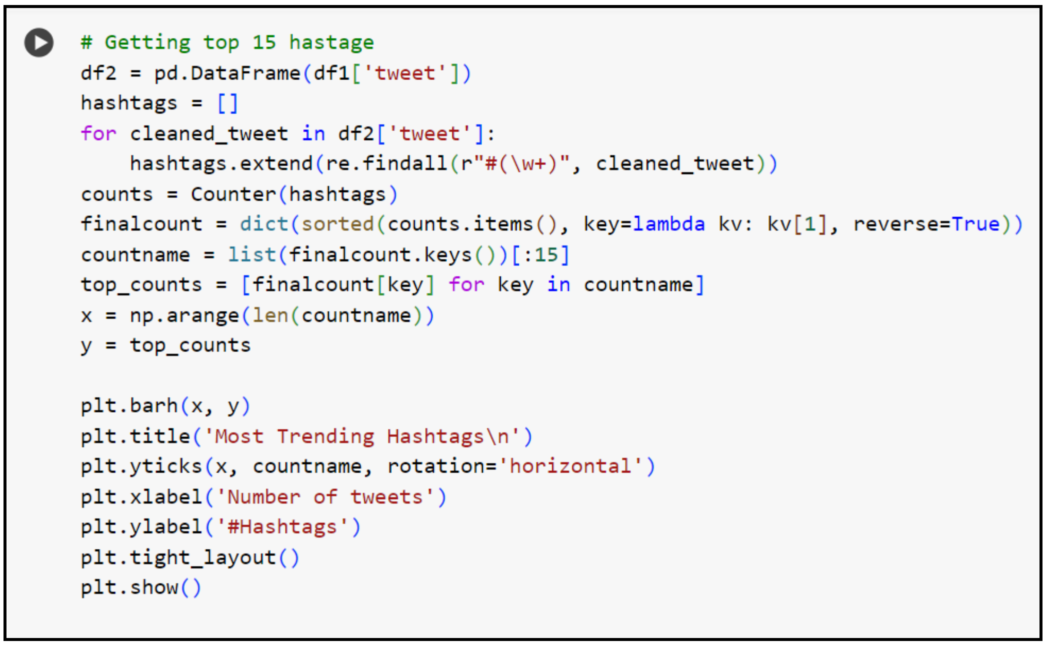
**Figure 8: Presenting Distribution of Sentiments in form of Bar Graphs.**



```python
plt.pie(performance, labels=objects, autopct='%1.1f%%', shadow=False, startangle=140)
plt.axis('equal')
plt.title('Twitter Sentiment Analysis (Pie Chart)')
plt.show()
```

**Figure 9: Presenting Distribution of Sentiments in form of Pie Chart.**

The Figure 8, snip states the code of visual interpretation of the sentiment's distribution. Following that a bar graph is developed indicating an emotional Category and the height of the bar showing the number of tweets in that Category. Figure 9, the code that provide graphical representation of each sentiments class proportional distribution to the dataset's total sentiments components.



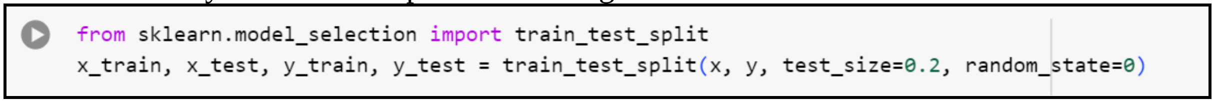
```python
# Getting top 15 hastage
df2 = pd.DataFrame(df1['tweet'])
hashtags = []
for cleaned_tweet in df2['tweet']:
    hashtags.extend(re.findall(r"#(\w+)", cleaned_tweet))
counts = Counter(hashtags)
finalcount = dict(sorted(counts.items(), key=lambda kv: kv[1], reverse=True))
countname = list(finalcount.keys())[:15]
top_counts = [finalcount[key] for key in countname]
x = np.arange(len(countname))
y = top_counts

plt.barh(x, y)
plt.title('Most Trending Hashtags\n')
plt.yticks(x, countname, rotation='horizontal')
plt.xlabel('Number of tweets')
plt.ylabel('#Hashtags')
plt.tight_layout()
plt.show()
```

**Figure 10: Presenting Distribution of Sentiments in form of Pie Chart.**

The code in above snip Figure 10, collects hashtags from the original dataset ‘tweet’ column’, calculate the number of times each hashtag shows up and create a horizontal bar graph exhibiting the top 15 trending hashtags.

### 5.1.5 Data Split

The data used by the model is split into training set and test set in ratio of 80/20.



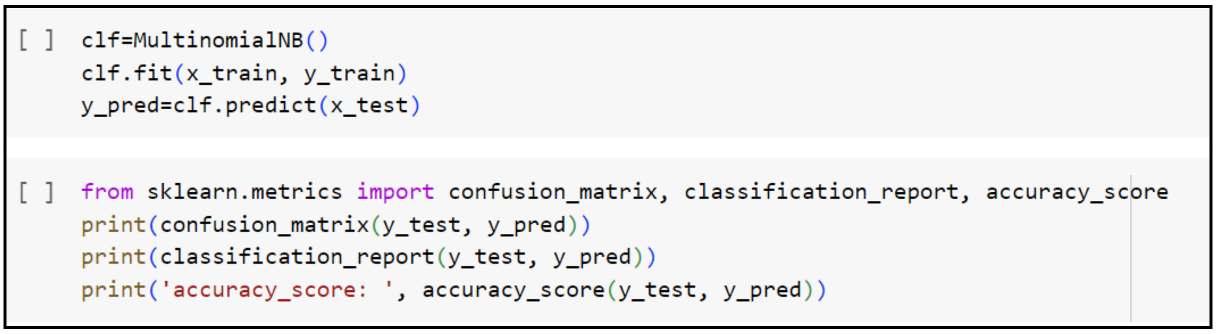
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

**Figure 11: Data Split.**

### 5.1.6 Model Implementation

This section contains includes the Python code for executing sentimental analysis on the demonetization Twitter Dataset, the models used are Naïve Bayes, LSTM and CNN.

#### 5.1.6.1 Naïve Bayes model



```python
[ ]  clf=MultinomialNB()
     clf.fit(x_train, y_train)
     y_pred=clf.predict(x_test)
```

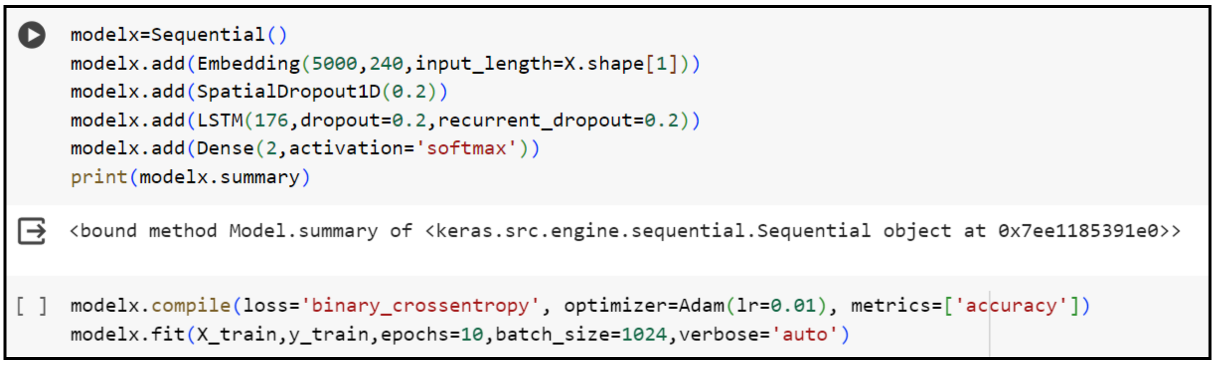
```python
[ ]  from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
     print(confusion_matrix(y_test, y_pred))
     print(classification_report(y_test, y_pred))
     print('accuracy_score: ', accuracy_score(y_test, y_pred))
```

**Figure 12: Naïve Bayes Model**

After the separation of the dataset into the training and testing the Naïve bayes model is carried out Figure 12. The multinomial Naïve Bayes model (clf) is developed and trained using training data (x_train, y_train). Next the model forecasts the sentimental labels for the test data (x_test) and in (y_pred) the outcomes are saved. The performance metric such as confusion metrix, classification report, accuracy score are then measured.

### 5.1.6.2 LSTM Model

The following Figure 11, illustrate the deployment of sentimental analysis model using LSTM architecture. Using the keras library the model is defined, which start with an embedding layer to understand word representation. It consists of dropout layer for regularity for collecting sequential connections. Further using the binary cross entropy loss and trained for 10 epochs using Adam optimizer.



```
modelx=Sequential()
modelx.add(Embedding(5000,240,input_length=X.shape[1]))
modelx.add(SpatialDropout1D(0.2))
modelx.add(LSTM(176,dropout=0.2,recurrent_dropout=0.2))
modelx.add(Dense(2,activation='softmax'))
print(modelx.summary)
```

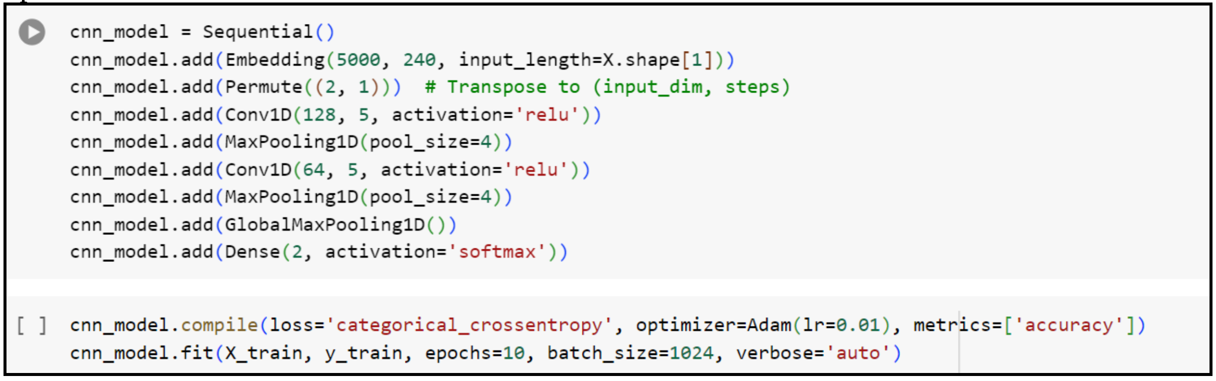
```
<bound method Model.summary of <keras.src.engine.sequential.Sequential object at 0x7ee1185391e0>>
```

```
[ ]  modelx.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.01), metrics=['accuracy'])
     modelx.fit(X_train,y_train,epochs=10,batch_size=1024,verbose='auto')
```

**Figure 11: LSTM Model**

### 5.1.6.3 CNN Model

The Figure 12 shows the implementation of CNN model. The model includes different layer such as embedding layer, convolution layer with max pooling for feature extraction and a dense layer for classification. The compilation is done using cross entropy loss and Adam optimizer.



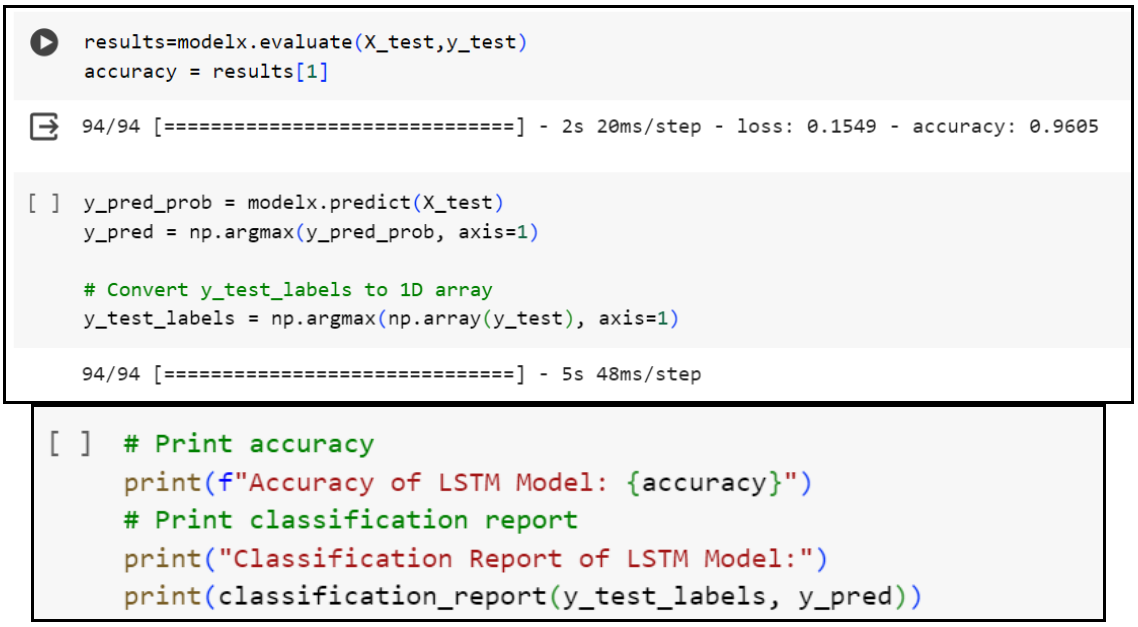
```
cnn_model = Sequential()
cnn_model.add(Embedding(5000, 240, input_length=X.shape[1]))
cnn_model.add(Permute((2, 1)))  # Transpose to (input_dim, steps)
cnn_model.add(Conv1D(128, 5, activation='relu'))
cnn_model.add(MaxPooling1D(pool_size=4))
cnn_model.add(Conv1D(64, 5, activation='relu'))
cnn_model.add(MaxPooling1D(pool_size=4))
cnn_model.add(GlobalMaxPooling1D())
cnn_model.add(Dense(2, activation='softmax'))
```

```
[ ]  cnn_model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.01), metrics=['accuracy'])
     cnn_model.fit(X_train, y_train, epochs=10, batch_size=1024, verbose='auto')
```

**Figure 12: CNN Model**

# 6 Evaluation of Sentimental Analysis Models

Furthermore in the Figure 13, the evaluate function figures the model performances, by extracting accuracy from the results. The model use ‘predict’ to calculate sentimental probabilities for test set, and argmax is used to calculate predicted sentimental labels. Next, the classification function from sklearn provided the classification report. Similary approach is used for measuring the CNN performances as well.



```
results=modelx.evaluate(X_test,y_test)
accuracy = results[1]

94/94 [==============================] - 2s 20ms/step - loss: 0.1549 - accuracy: 0.9605

[ ] y_pred_prob = modelx.predict(X_test)
    y_pred = np.argmax(y_pred_prob, axis=1)

    # Convert y_test_labels to 1D array
    y_test_labels = np.argmax(np.array(y_test), axis=1)

    94/94 [==============================] - 5s 48ms/step

[ ] # Print accuracy
    print(f"Accuracy of LSTM Model: {accuracy}")
    # Print classification report
    print("Classification Report of LSTM Model:")
    print(classification_report(y_test_labels, y_pred))
```

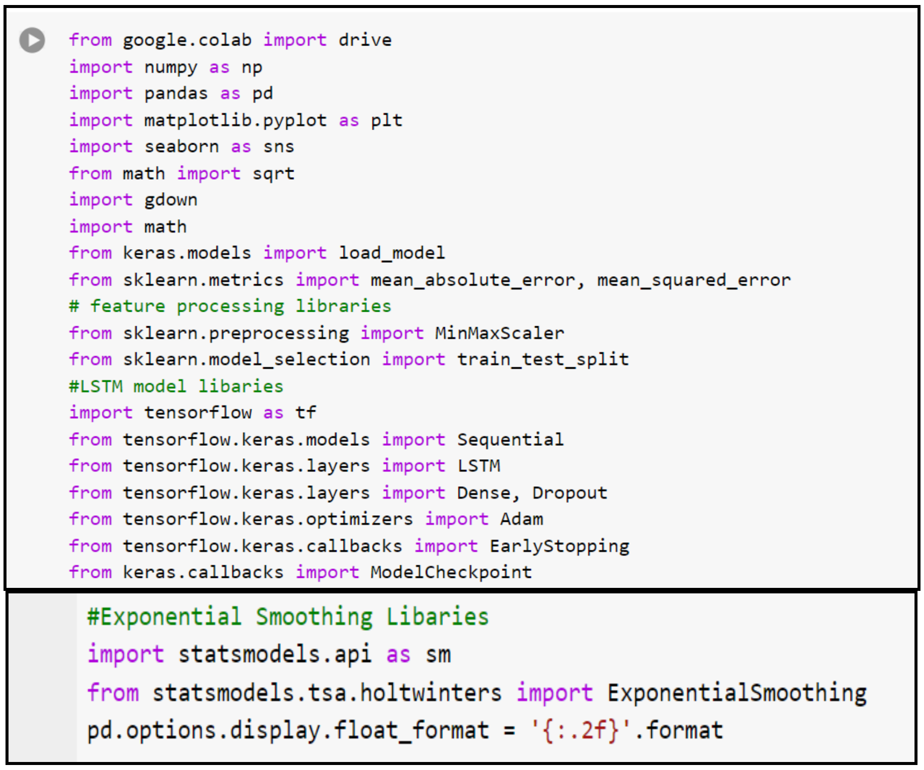
**Figure 13: Model Evaluation**

## 7 Time Series Analysis on Digital Payment dataset

This section will explain the step use to execute the Time series analysis.

### 7.1.1 Importing Libraries

In Google collab, necessary libraries were imported and installed to support the time series analysis. These Libraries are important for data processing, and visualization also to develop the LSTM and Exponential Smoothing model for time series analysis.

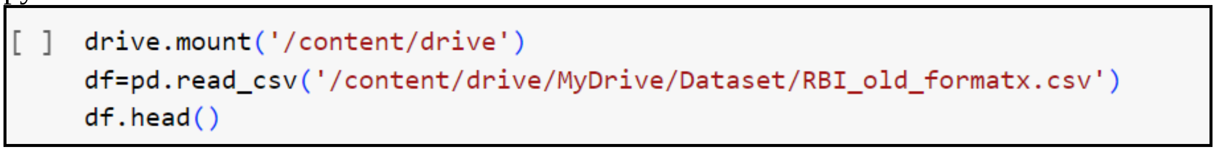


```
from google.colab import drive
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from math import sqrt
import gdown
import math
from keras.models import load_model
from sklearn.metrics import mean_absolute_error, mean_squared_error
# feature processing libraries
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
#LSTM model libaries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
#Exponential Smoothing Libaries
import statsmodels.api as sm
from statsmodels.tsa.holtwinters import ExponentialSmoothing
pd.options.display.float_format = '{:.2f}'.format
```

**Figure 14: Libraries**

### 7.1.2 Loading Dataset

The Following code loads dataset from google drive into google collab using drive library of python.

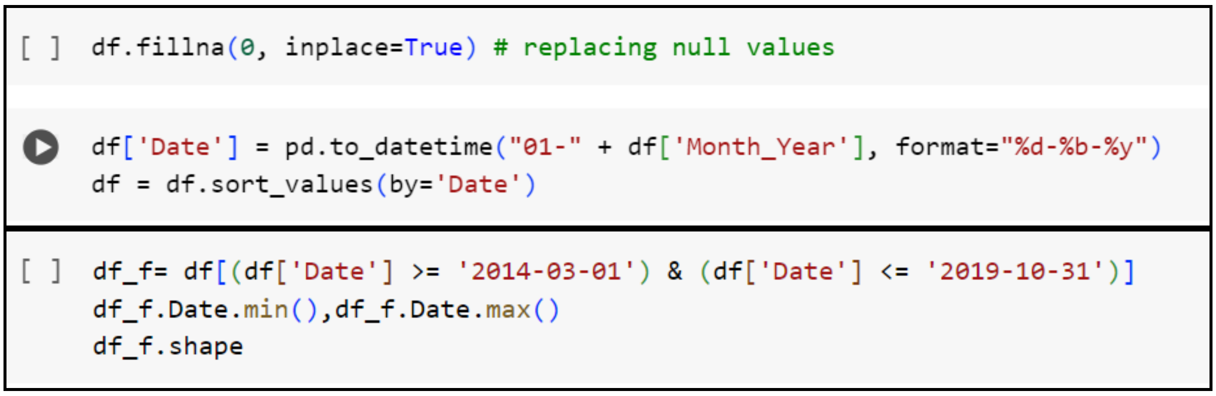


```
[ ]  drive.mount('/content/drive')
     df=pd.read_csv('/content/drive/MyDrive/Dataset/RBI_old_formatx.csv')
     df.head()
```

**Figure 15: Loading RBI Dataset**

### 7.1.3 Data Pre-processing

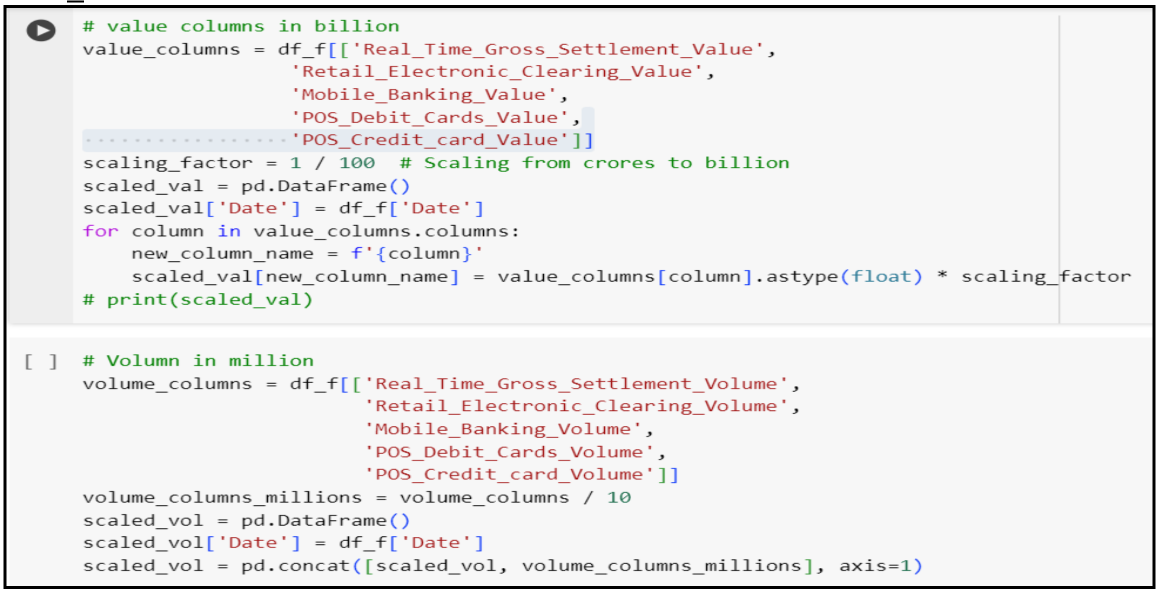
In Figure 16, it states that the null or missing values were replaced by zeros. Next the ‘Month_Year’ column is then transformed into a datetime format and a new column name as ‘Date’ is created. Finally, the dataset is sort based on the ‘Date’ column. Furthermore, the dataset is narrow down to time of March 2014 to October 2019.



```
[ ]  df.fillna(0, inplace=True) # replacing null values

     df['Date'] = pd.to_datetime("01-" + df['Month_Year'], format="%d-%b-%y")
     df = df.sort_values(by='Date')

[ ]  df_f= df[(df['Date'] >= '2014-03-01') & (df['Date'] <= '2019-10-31')]
     df_f.Date.min(),df_f.Date.max()
     df_f.shape
```

**Figure 16: Date Formating**

As mentioned in Figure 17, scaling is done (Anon., 2019) on the volume columns from crores to billion by creating another dataframe named as ‘scaled_val’. The scaling factor used to each column is 1/100. Similarly the volumn columns are converted to millions creating ‘scaled_vol’ database.

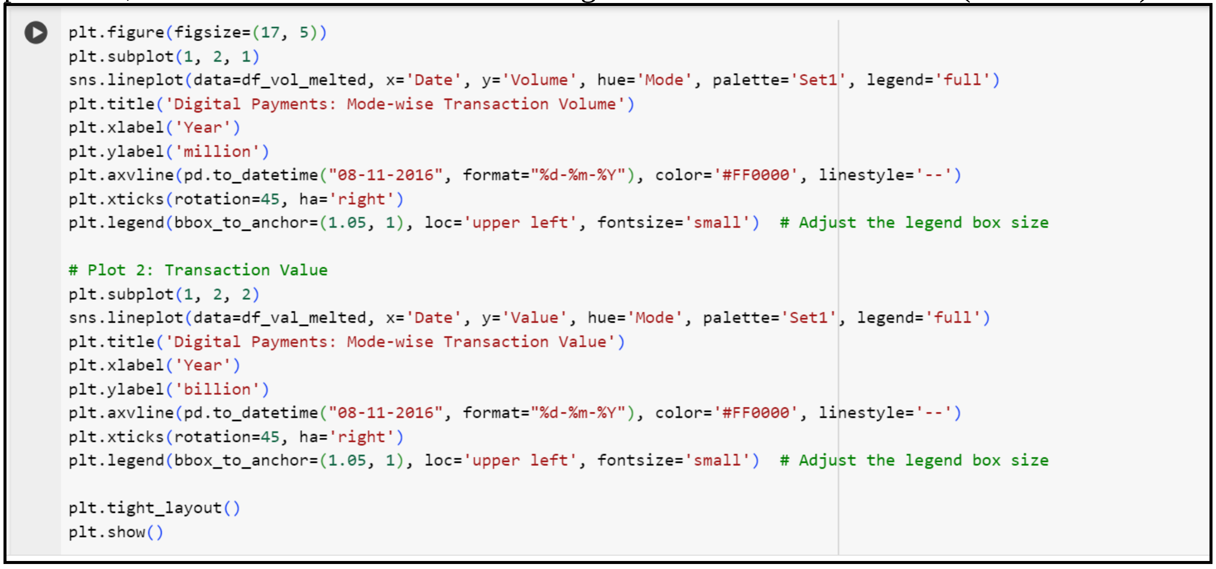


```
# value columns in billion
value_columns = df_f[['Real_Time_Gross_Settlement_Value',
                 'Retail_Electronic_Clearing_Value',
                 'Mobile_Banking_Value',
                 'POS_Debit_Cards_Value',
                 'POS_Credit_card_Value']]
scaling_factor = 1 / 100  # Scaling from crores to billion
scaled_val = pd.DataFrame()
scaled_val['Date'] = df_f['Date']
for column in value_columns.columns:
    new_column_name = f'{column}'
    scaled_val[new_column_name] = value_columns[column].astype(float) * scaling_factor
# print(scaled_val)

[ ]  # Volumn in million
volume_columns = df_f[['Real_Time_Gross_Settlement_Volume',
                       'Retail_Electronic_Clearing_Volume',
                       'Mobile_Banking_Volume',
                       'POS_Debit_Cards_Volume',
                       'POS_Credit_card_Volume']]
volume_columns_millions = volume_columns / 10
scaled_vol = pd.DataFrame()
scaled_vol['Date'] = df_f['Date']
scaled_vol = pd.concat([scaled_vol, volume_columns_millions], axis=1)
```

**Figure 17: Scaling Formating**

### 7.1.4 Exploratory Data Analysis

Mode wise transaction value and transaction volume of digital payment over the time is represent through below code snip Figure 18. Each subplot shows trends for various processes, with the vertical dashed line stating the date of Demonetization (8th Nov 2016).

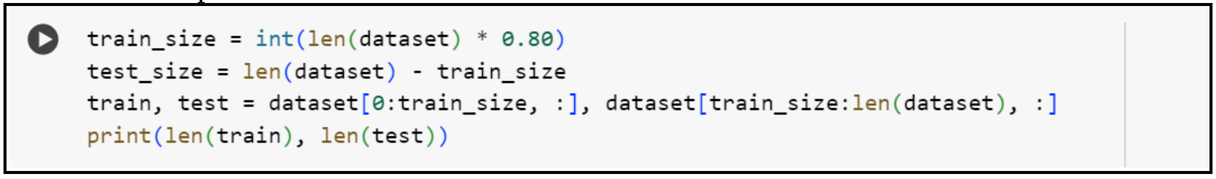


```
plt.figure(figsize=(17, 5))
plt.subplot(1, 2, 1)
sns.lineplot(data=df_vol_melted, x='Date', y='Volume', hue='Mode', palette='Set1', legend='full')
plt.title('Digital Payments: Mode-wise Transaction Volume')
plt.xlabel('Year')
plt.ylabel('million')
plt.axvline(pd.to_datetime("08-11-2016", format="%d-%m-%Y"), color='#FF0000', linestyle='--')
plt.xticks(rotation=45, ha='right')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', fontsize='small')  # Adjust the legend box size

# Plot 2: Transaction Value
plt.subplot(1, 2, 2)
sns.lineplot(data=df_val_melted, x='Date', y='Value', hue='Mode', palette='Set1', legend='full')
plt.title('Digital Payments: Mode-wise Transaction Value')
plt.xlabel('Year')
plt.ylabel('billion')
plt.axvline(pd.to_datetime("08-11-2016", format="%d-%m-%Y"), color='#FF0000', linestyle='--')
plt.xticks(rotation=45, ha='right')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', fontsize='small')  # Adjust the legend box size

plt.tight_layout()
plt.show()
```

**Figure 18: Mode wise Transaction ploting.**

### 7.1.5 Data Spilting

The split is done in a ratio of 80-20, where 80% of data comes under training and rest 20% goes to testing. The training set covers from March 2014 to August 2018, while testing set covers from September 2018 to October 2019.



```
train_size = int(len(dataset) * 0.80)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
print(len(train), len(test))
```

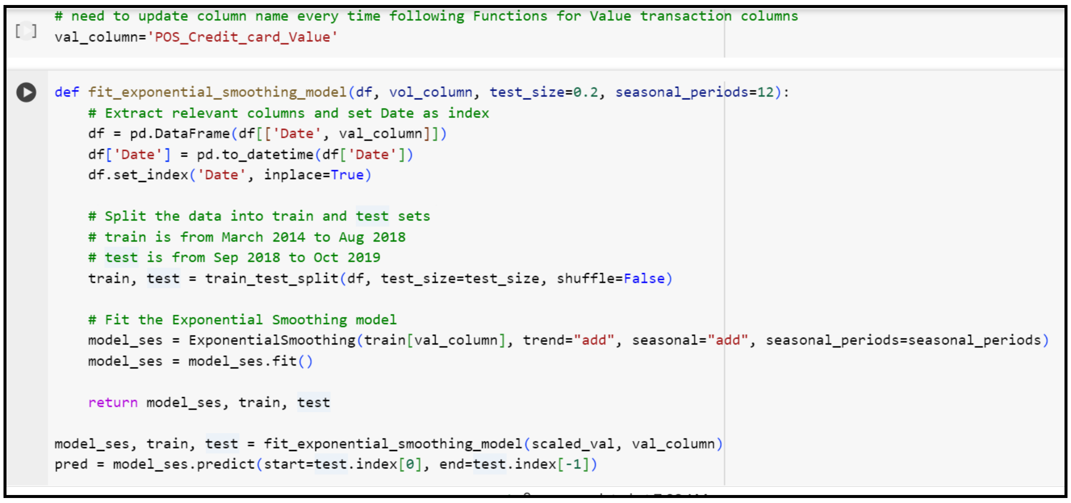
**Figure 19: Data Split.**

### 7.1.6 Model implementation

This section contains includes the Python code for executing Time series analysis on the payment dataset, the models used are LSTM, Exponential Smoothing.

#### 7.1.6.1 Exponential Smoothing

The Exponential Smoothing model is developed Figure 20, the model is fitted to the training dataset, including the addictive trends, seasonality based seasonal period of 12 months. Further the model is tested on testing dataset providing conclusion on its efficacy on unseen data.



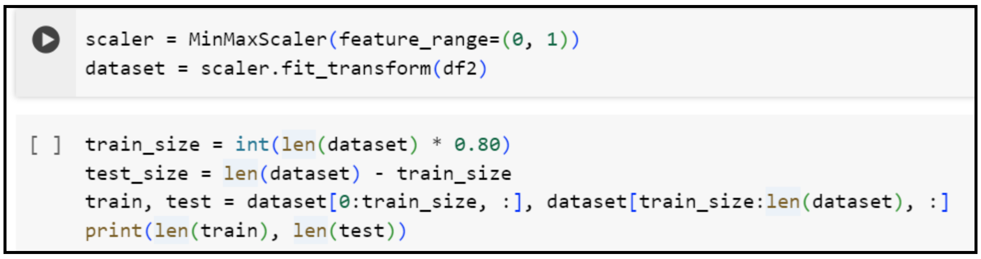
```
[ ] # need to update column name every time following Functions for Value transaction columns
    val_column='POS_Credit_card_Value'

    def fit_exponential_smoothing_model(df, vol_column, test_size=0.2, seasonal_periods=12):
        # Extract relevant columns and set Date as index
        df = pd.DataFrame(df[['Date', val_column]])
        df['Date'] = pd.to_datetime(df['Date'])
        df.set_index('Date', inplace=True)

        # Split the data into train and test sets
        # train is from March 2014 to Aug 2018
        # test is from Sep 2018 to Oct 2019
        train, test = train_test_split(df, test_size=test_size, shuffle=False)

        # Fit the Exponential Smoothing model
        model_ses = ExponentialSmoothing(train[val_column], trend="add", seasonal="add", seasonal_periods=seasonal_periods)
        model_ses = model_ses.fit()

        return model_ses, train, test

    model_ses, train, test = fit_exponential_smoothing_model(scaled_val, val_column)
    pred = model_ses.predict(start=test.index[0], end=test.index[-1])
```

**Figure 19: Exponential model**

### 7.1.6.2 LSTM Model

The LSTM model has been developed to perform the time series analysis on the training set and test set of the digital payment dataset. Deep learning libraries, particularly from TesnorFlow and keras structure are included to detect trends and relationships in the data during modelling.

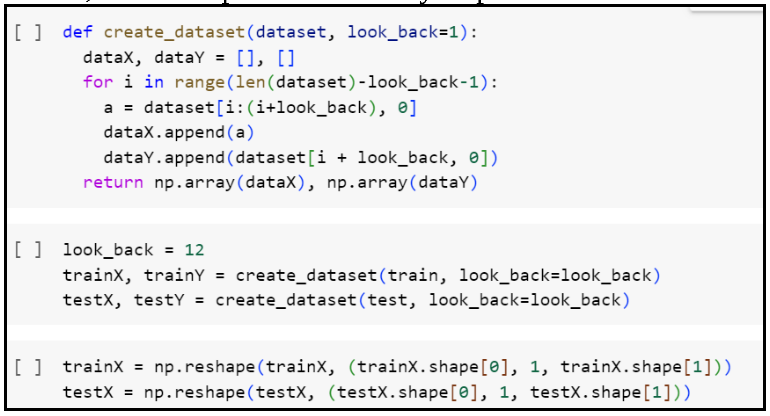
The MinMax Scalar function is used to scale the features. All the values have been transformed in the range of [0,1]. Model such as LSTM need scaled data, therefore feature scaling is done.



```
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(df2)

[ ] train_size = int(len(dataset) * 0.80)
    test_size = len(dataset) - train_size
    train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
    print(len(train), len(test))
```

**Figure 20: MinMax Scaling of feature**

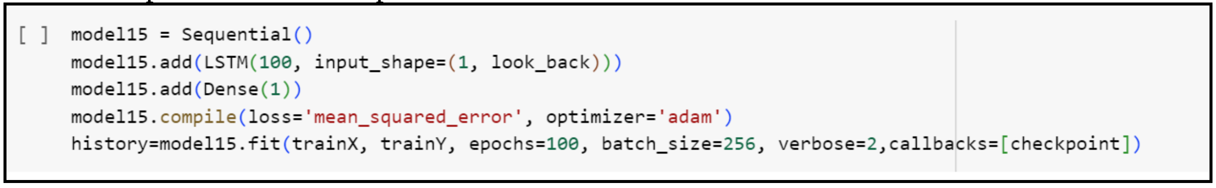
The following code Figure 21, creates the input and output pairs with a specific look-back window this approach produces time series data for training an LSTM model. It ensure that the input sequences are structured correctly for LSTM model, has it require three dimensional shape for storing sample, features, and time steps. The look back window allows the model to learn from the past data, which improves its ability in prediction.



```
[ ] def create_dataset(dataset, look_back=1):
      dataX, dataY = [], []
      for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
      return np.array(dataX), np.array(dataY)

[ ] look_back = 12
    trainX, trainY = create_dataset(train, look_back=look_back)
    testX, testY = create_dataset(test, look_back=look_back)

[ ] trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
    testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

**Figure 21: Created Sequences of input data**

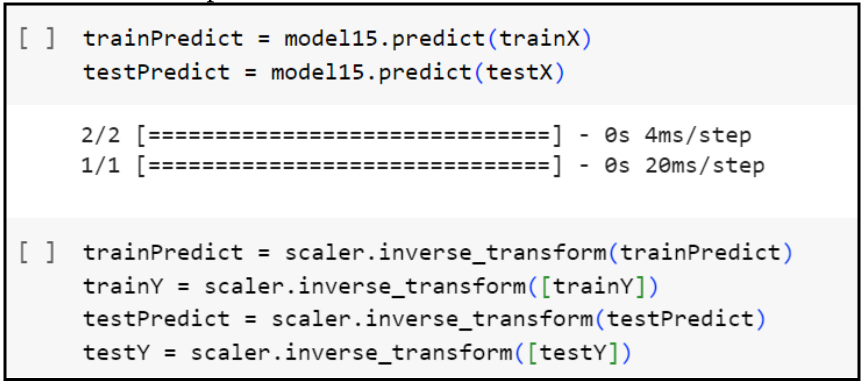
The LSTM is constructed using LSTM layer with 100 units, by taking input sequences (1, look_back). The model is trained on mean squared error as loss function, and it is compiled on Adam optimizer for 100 epochs with a batch size of 256.



```
[ ]  model15 = Sequential()
     model15.add(LSTM(100, input_shape=(1, look_back)))
     model15.add(Dense(1))
     model15.compile(loss='mean_squared_error', optimizer='adam')
     history=model15.fit(trainX, trainY, epochs=100, batch_size=256, verbose=2,callbacks=[checkpoint])
```

**Figure 20: LSTM Model**

The projection made by LSTM model on both train and test data are inserve transformed using a scaler to return the expected values in the initial data.

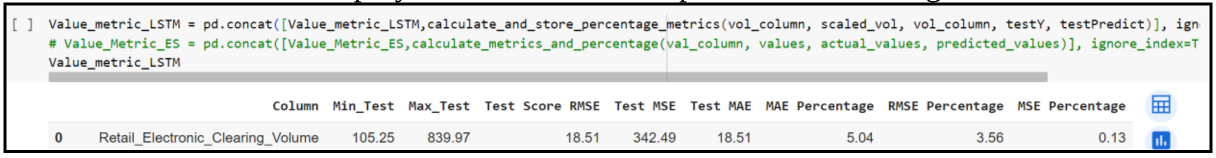


```
[ ]  trainPredict = model15.predict(trainX)
     testPredict = model15.predict(testX)

     2/2 [==============================] - 0s 4ms/step
     1/1 [==============================] - 0s 20ms/step

[ ]  trainPredict = scaler.inverse_transform(trainPredict)
     trainY = scaler.inverse_transform([trainY])
     testPredict = scaler.inverse_transform(testPredict)
     testY = scaler.inverse_transform([testY])
```

**Figure 21: Inverse Transform**

## 8 Evaluation of Time series models

A function was created named as "calculate_and _store_percentage_metric" to measure various parameters such as RMSE, MSE, and MAE and stores the results in a data frame. This same function was employed to evaluate the Exponential smoothing execution as well.



```
[ ]  Value_metric_LSTM = pd.concat([Value_metric_LSTM,calculate_and_store_percentage_metrics(vol_column, scaled_vol, vol_column, testY, testPredict)], ign
     # Value_Metric_ES = pd.concat([Value_Metric_ES,calculate_metrics_and_percentage(val_column, values, actual_values, predicted_values)], ignore_index=T
     Value_metric_LSTM
```

|  | Column | Min_Test | Max_Test | Test Score RMSE | Test MSE | Test MAE | MAE Percentage | RMSE Percentage | MSE Percentage |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Retail_Electronic_Clearing_Volume | 105.25 | 839.97 | 18.51 | 342.49 | 18.51 | 5.04 | 3.56 | 0.13 |

**Figure 22: Evaluation Metric for Time Series model**

## 9 Conclusion

In summary, this confirguration manual allows researchers to go through the process of investigating people' opinions regarding demonetization through English tweets and the impact on the digital payment due to the demonetization. It covers the hardware specifications of a local machine and Google Collab along with the software setup using python and specific libraries like tensorflow which is used for the implementation of this research. The dataset for sentimental analysis is collected from open source platforms called kaggle, whereas the payment data for time series analysis is collected from Reserve Bank of India (RBI) database. After loading of datasets furthers steps includes data preparation, exploratory data analysis, model implementation and evaluation as mentioned above. For the sentimental analysis model such as LSTM, CNN, and Naïve bayes is developed, the implementation of this models are briefly mentioned in the provide code snip. For analysing the trend of digital payments after the demonetizatio following model such as LSTM and Exponential Smoothing are deployed, the architecture of the model is mentioned in above code. The performance of sentimental classification model are measured using Accuracy, precision, recall, and F1 scores. Similarly to evaluate the performances of Time series problem MSE, MAE, and RMSE is calculated. The purpose is to provide reseachers with clear, sequential method to study public attitudes and digital payment trends throughout the demonetization era.

## 10 References

Anon., 2019. DEMONETISATION AND ITS IMPACT ON DIGITISATION IN INDIA. *Management Dynamics*, 19(1), p. 17.

Ghulam Musa Raza, Z. S. B. S. L. A. W., 2021. Sentiment Analysis on COVID Tweets: An Experimental Analysis on the Impact of Count Vectorizer and TF-IDF on Sentiment Predictions using Deep Learning Models. *2021 International Conference on Digital Futures and Transformative Technologies,* p. 6.

Kannan Suseelan Unnithan, S. A. S. a. K. S., 2022. Analyzing Tweet Data to Identify the Impact of Demonetarization. *Proceedings of International Conference on Emerging Technologies and Intelligent Systems,* p. 7.

Kumar Abhishek, M. M. M. S. S. M., 2020. SENTIMENTAL ANALYSIS FOR MOVIE REVIEWS. *International Journal of Advanced Research in Computer Science* , 11(1), p. 6.