National College of Ireland

Comparing Machine Learning Models for Predicting the Global Internet Usage

MSc Research Project
Data Analytics

Tijo Sebastian
Student ID: x21139237

School of Computing
National College of Ireland

Supervisor: Mr. Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Tijo Sebastian
Student ID:	x21139237
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Mr. Hicham Rifai
Submission Due Date:	01/02/2023
Project Title:	Comparing Machine Learning Models for Predicting the Global Internet Usage
Word Count:	850
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	31st January 2023

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):	

Comparing Machine Learning Models for Predicting the Global Internet Usage

Tijo Sebastian
x21139237

1 Introduction

The following describes the hardware and software specifications required for the working of this research titled "Comparing Machine Learning Models for Predicting the Global Internet Usage".

2 System Configuration

2.1 Hardware Configuration

All the hardware requirements for this research are mentioned in the table below.

2.2 Software Configuration

All the software used for this study along with their versions is presented below.

2.2.1 Anaconda Navigator

The python code required for this research was entirely done in a Jupyter environment. The Jupyter notebook environment was provided by the Anaconda Navigator.

2.2.2 Jupyter Notebook

The entire coding required for this research was done in Jupyter notebook.

2.2.3 Python

All the processing and the machine learning models are performed in python. Python provided support for all the required packages. The list of the packages used is shown in the table below.

Hardware	Configuration
RAM	8 GB
Processor	11th GEN INTEL Core (TM)
Hard disk storage	580 GB

Software	Configuration
Anaconda	2.1.4
Jupyter	6.4.5
Python	3.10.4

Package	Version
pandas	1.3.4
matplotlib	3.4.3
numpy	1.20.3
tensorflow	2.10.0
sklearn	0.24.2
statmodels	0.12.2

3 Project Development

3.1 Data Preparation

The entire coding was done in Python. Both LSTM and SES models are implemented in different Jupyter notebooks. As mentioned above the Python version of 3.10.4 is used.

3.2 Loading the Data

The required data has been gathered from an open-source platform known as Kaggle datasets. This original data source has 4 different CSV files which have the global internet usage data over the years from 1970 till 2019. This research considers only 2 files among them. The original dataset has been cut down from 11000 to 2395 when the data was considered between the years 2010 till 2019. This dataset is loaded into the Jupyter environment and loaded using pandas and it is saved under the name 'Internet$_data(F)'$.

3.3 Identifying missing values

As mentioned before, this research considers only data ranging from the years 2010 to 2019. After the dataset is loaded into the Jupyter environment, Isnull.sum () is performed to identify the missing values. There were no missing values included in the dataset.

3.4 Feature Engineering

The original dataset has only yearly data whereas there was no mention of the month at all. to perform the monthly data forecasting if needed, this research has included the month along with each year. The null values were cleared when the data was considered between 2010 till 2019. These different CSV files have been combined into one so that the data can be retrieved from a single source. Smyl (2020) explained the difficulty faced during their research. Adding the month feature will surpass this problem.
The final dataset contains 2395 rows and 6 columns.

3.5 Splitting of Data

The data have been cut down for training and testing the model. The test and train split ratio for the LSTM model is 6.7:3.3. That is, among the 2395 rows, 1604 values (67%)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

In [2]:
```
dataset = pandas.read_csv('Internet_data(F).csv', usecols=[7], engine='python')
dataset.head()
```

Out[2]:

	Mobile_cellular_subscriptions
0	35.003127
1	45.813626
2	49.227977
3	52.083576
4	55.159515

Figure 1: Loading the Dataset

Reading the Dataset

In [19]:
```
df = pd.read_csv('Internet_data(F).csv')
df.head(10)
```

Out[19]:

	Id	Country_name	Code	Year	Individuals_using_the_Internet	Mobile_cellular_subscriptions
0	1	Afghanistan	AFG	31/12/2010	2.136141	35.003127
1	2	Afghanistan	AFG	31/12/2011	2.977944	45.813626
2	3	Afghanistan	AFG	31/12/2012	3.070895	49.227977
3	4	Afghanistan	AFG	31/12/2013	3.715304	52.083576
4	5	Afghanistan	AFG	31/12/2014	6.837084	55.159515
5	6	Afghanistan	AFG	31/12/2015	8.217330	57.271068
6	7	Afghanistan	AFG	31/12/2016	10.252818	61.054638
7	8	Afghanistan	AFG	31/12/2017	11.876680	65.929134
8	9	Afghanistan	AFG	31/12/2018	14.366508	59.120848
9	10	Afghanistan	AFG	31/12/2019	17.483667	59.356020

Figure 2: Loading the Dataset

```
In [3]: dataset.isnull().sum()

Out[3]: Individuals_using_the_Internet    0
        dtype: int64
```

Figure 3: Identifying the missing values

```
In [19]: df = pd.read_csv('Internet_data(F).csv')
         df.head(10)
```

Out[19]:

	Id	Country_name	Code	Year	Individuals_using_the_Internet	Mobile_cellular_subscriptions
0	1	Afghanistan	AFG	31/12/2010	2.136141	35.003127
1	2	Afghanistan	AFG	31/12/2011	2.977944	45.813626
2	3	Afghanistan	AFG	31/12/2012	3.070895	49.227977
3	4	Afghanistan	AFG	31/12/2013	3.715304	52.083576
4	5	Afghanistan	AFG	31/12/2014	6.837084	55.159515
5	6	Afghanistan	AFG	31/12/2015	8.217330	57.271068
6	7	Afghanistan	AFG	31/12/2016	10.252818	61.054638
7	8	Afghanistan	AFG	31/12/2017	11.876680	65.929134
8	9	Afghanistan	AFG	31/12/2018	14.366508	59.120848
9	10	Afghanistan	AFG	31/12/2019	17.483667	59.356020

Figure 4: Combined Dataaset with added month

```
In [47]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2395 entries, 0 to 2394
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Id                              2395 non-null   object
 1   Country_name                    2395 non-null   object
 2   Code                            1952 non-null   object
 3   Year                            2395 non-null   object
 4   Individuals_using_the_Internet  2395 non-null   float64
 5   Mobile_cellular_subscriptions   2395 non-null   float64
dtypes: float64(2), object(4)
memory usage: 112.4+ KB
```

Figure 5: Dataset info

are used for model training whereas the remaining 1604 values (33%) are used for model testing. The SES model uses the weighted average method which takes the most recent values for testing. 2395 values are used for training and recent 30 observations are used for testing the model.

```
In [6]: # split into train and test sets
        train_size = int(len(dataset) * 0.67)
        test_size = len(dataset) - train_size
        train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
        print(len(train), len(test))

        1604 791
```

Figure 6: Data Splitting

3.6 Feature Scaling

The chances of having diversity in the feature on the combined dataset are high. To normalize this, a package called MinMaxScaler is imported from the Sklearn library. This transforms all the features on a scale ranging from 0 to 1.

4 Model Application

This research will forecast global internet usage on the top of two variables. They are the individuals using the internet and mobile cellular subscriptions. The dataset has values till the year 2019 and with that data, this research will try to predict the future using LSTM and SES models. These models are evaluated with appropriate error metrics and select the best model.

4.1 Long Short-term Memory (LSTM) Model

The Long Short-Term Memory network, also known as the LSTM network, is a recurrent neural network that fixes the vanishing gradient issue. It was trained using backpropagation through time. As a result, it can be used to build substantial recurrent networks, which can then be utilized to tackle challenging sequence issues in machine learning and produce cutting-edge outcomes.
The above figure shows the sequential LSTM model implementation. The look$_b ackvalueissetto10whichu$

4.2 Simple Exponential Smoothing (SES) Model

Exponential smoothing is a time series forecasting technique for univariate data. The forecast of time series methods like the Box-Jenkins ARIMA family of methods is a weighted linear sum of recent past data or lags.

4.3 Model Evaluation

Both models are evaluated using the RMSE value. the following figures show how the RMSE value is calculated.

Train and Test data splitting

```
In [4]: #train data split
        train = X.iloc[:-30]

In [5]: #test data split
        test = X.iloc[-30:]

In [6]: #printing the shape of the train data
        train.shape

Out[6]: (2365,)

In [7]: #printing the shape of the test data
        test.shape

Out[7]: (30,)
```

Figure 7: Data Splitting in SES

```
In [5]: # normalize the dataset
        scaler = MinMaxScaler(feature_range=(0, 1))
        dataset = scaler.fit_transform(dataset)
```

Figure 8: Feature Scaling

Creating the LSTM model

In [10]:
```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(2, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=10, batch_size=1, verbose=2)
```

Figure 9: Creating LSTM model

```
fitSES = SimpleExpSmoothing(np.asarray(train)).fit( smoothing_level = 0.1,optimized= False)
fcst_gs_pred = fitSES.forecast(30)
timeseries_evaluation_metrics_func(test,fcst_gs_pred)
```

Figure 10: Creating SES model

In [11]:
```
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# calculate root mean squared error
trainScore = np.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = np.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
51/51 [==============================] - 1s 1ms/step
25/25 [==============================] - 0s 2ms/step
Train Score: 20.78 RMSE
Test Score: 21.30 RMSE
```

Figure 11: Calculating RMSE value

In [11]:
```
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# calculate root mean squared error
trainScore = np.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = np.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
51/51 [==============================] - 1s 2ms/step
25/25 [==============================] - 0s 1ms/step
Train Score: 16.98 RMSE
Test Score: 16.63 RMSE
```

Figure 12: Calculating RMSE value

In [46]:
```
#Automated smoothing_level
fitSESauto = SimpleExpSmoothing(np.asarray(train)).fit( optimized= True, use_brute = True)
fcst_auto_pred = fitSESauto.forecast(30)
timeseries_evaluation_metrics_func(test,fcst_auto_pred)
```

```
Evaluation metric results:-
MSE is : 1425.9920458139231
MAE is : 28.486112293892585
RMSE is : 37.762309858030704
MAPE is : inf
R2 is : -1.0846192636801795
```

Figure 13: Calculating RMSE value

In [44]:
```
#Automated smoothing_level
fitSESauto = SimpleExpSmoothing(np.asarray(train)).fit( optimized= True, use_brute = True)
fcst_auto_pred = fitSESauto.forecast(30)
timeseries_evaluation_metrics_func(test,fcst_auto_pred)
```

```
Evaluation metric results:-
MSE is : 1373.2473729944647
MAE is : 32.51489314261733
RMSE is : 37.05735248226004
MAPE is : 53.37092999343395
R2 is : -3.1691802635838435
```

Figure 14: Calculating RMSE value

References

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, *International Journal of Forecasting* **36**(1): 75–85.