MEASUREMENT BASED PERFORMANCE ANALYSIS OF 3G AND 4G LTE NETWORK

By:

RUPAK KALITA (CSE/16032/187)

ADITYA (CSE/16003/160)

ABHISHEK KUMAR (CSE/16002/159)

Bachelor Thesis submitted to

Indian Institute of Information Technology Kalyani

for the partial fulfillment of the degree of

Bachelor of Technology

in

Computer Science and Engineering

June 2020

Certificate

This is to certify that the thesis entitled "Measurement based performance analysis of 3G and 4G LTE network" being submitted by Rupak Kalita, Aditya and Abhishek Kumar, undergraduate students, Reg. No 0000187, Roll No - CSE/16032, Reg. No 0000160 Roll No – CSE/16003 and Reg. No 0000159, Roll No - CSE/16002 respectively in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India, for the award of Bachelors of Technology in Computer Science and Information Engineering /Information Technology is an original research work carried by them under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of Indian Institute of Information Technology Kalyani and in my opinion, has reached the standards needed for submission. The work, techniques and the results presented have not been submitted to any other University or Institute for the award of any other degree or diploma.

Dr. Dalia Nandi (Das)

Assistant Professor

Electronics and Communication Engineering

Indian Institute of Information Technology Kalyani

Declaration

I hereby declare that the work being presented in this thesis entitled,"Measurement based performance analysis of 3G and 4G LTE network", submitted to Indian Institute of Information Technology Kalyani in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering during the period from July, 2019 to June, 2020 under the supervision of Dr Dalia Nandi (Das), Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India.

Rupak Kalita (CSE/16032/187)

Aditya (CSE/16003/160)

Abhishek Kumar(CSE/16002/159)

Computer Science and engineering
Indian Institute of Information Technology Kalyani

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

....

Dr. Dalia Nandi (Das)

Assistant professor

Electronics and Communication Engineering

Place: IIIT Kalyani

Date: 17/06/20

Acknowledgments

I would like to take this opportunity to thank our supervisor Dr. Dalia Nandi for being kind to help us through and guiding us through the technicalities and giving some very intuitive, insightful and valuable feedback regarding the same.

Rupak Kalita (CSE/16032/187)

Aditya (CSE/16003/160)

Abhishek Kumar(CSE/16002/159)

Computer Science and Engineering

Indian Institute of Information Technology Kalyani

Place: Indian Institute of Information Technology Kalyani, Nadia, West Bengal

Date: 17/06/2020

Abstract

Measurement based performance analysis of 3G and 4G LTE network

This project explores the various aspects related to 3G and 4G networks and its various parameters such as download speed, signal strength etc. In the previous semester, we made our own Android application in which we applied linear regression for analyze between various fields mentioned among various network operators. In this, we saw how various operator works differently in a remote place of India i.e Kalyani, West Bengal. The main motive was to make a analysing application which can be used by the mobile network operators to compare and improve their networks. In this semester, we have mainly focussed to make a predictive model by using machine learning concepts such as LSTM which is a part of RNN. The data was provided by our mentor which required a complete pre-processing according to our model. The RMSE was able to be reduced to a large extent.

Table of contents

1. Introduction

1.1 Network analysis

The society's increased reliance on Mobile networks has led to big challenges for service providers in terms of provisioning uninterrupted coverage, providing high network performance, and achieving high user quality of experience (QoE) .It helps service providers to further enhance the capabilities of their mobile networks by designing new technologies that provide new applications and services, and be able to cope with the growth in traffic volume with a wide variety of user devices. This study aims to understand the actual user Mobile network experience in rural area of India.
In addition, this can also help consumers to check the differences between 3G,4G and 5G networks, understand the performance benefits of 4G over 3G and to identify performance differences between different mobile operator's networks. Also, this study will be able to contribute to planning 5G networks efficiently. Our measurements covers one MBB service i.e web browsing through 2 distinctive platforms: Google chrome and Mozilla Firefox.

1.2 Prediction of network parameters

In the everyday world, people use mobile phones on a daily basis. Due to this, use of networks is increasing enormously. In India itself, more than 500 million people uses mobile phones on the basis of a survey done in 2019. These has led to a enormous competition between the mobile network operators. Every operator is trying to reach people and incresing their bandwidths. In this competitive network world, we tried to put our efforts and reach two goals. First goal is to make an analysis algorithm which will help to compare signal strengths and download speed between different operators. This might also help the users in choosing the best network operator in their region/city. The second goal is to help out or ease out for the users. We tried to make an algorithm which will help to predict the signal strength and download speed on the basis of previous records. This will help the users to do the actions required on the basis of the information for the next upcoming minutes/hours.For eg, they can schedule/postpone their activites, can save the current work etc. This will help the IT companies as well where people work on live projects and can prevent data loss due to sudden network failure by monitoring the predicted signal strengths in the upcoming hours. The benefits will also be much higher if it can be applied in the practical world and is incorporated in the mobile devices.

2: Objective

2.1 Network analysis

The main objective of this study/project is to make a analysis model for the dimensions of mobile network which will help the users many folds. We have divided the tasks keep this objective in mind i.e data analysis and machine learning. The analysis is done in a single location i.e Kalyani, West Bengal, India. The data is being collected in real time and processed on the basis of the analysis required.

2.2 Prediction of network parameters

The main objective of this study/project is to make a predictive model for network strength and signal speed .Here,we used two machine learning algo named as Linear regression and LSTM. The machine learning part includes the predictive model which is done using the data provided to us.In this prediction ,we also calculate error and accuracy measures of both the algorithms. The objective in this was to reduce the RMSE to a large extent.

3. Methodology

3.1 Network analysis

To attain the planned objective, we divided our work in different phases. In the first phase, we tried and completed the analysis model while collecting the data through our own mobile application made by using technologies and modules provided by Android. To ensure that each network was tested on an equal basis, our test processes were designed with the following assumptions and settings:

(i) Each network was tested concurrently to ensure that environmental conditions were the same for each service operator.

(ii) Identical handsets were used for each network. Undue contention was avoided by testing networks in parallel and ensuring that no concurrent tests were run on the same network.

We collected our data using a mobile application developed by us, namely a Speedtest app which can store the network speed. These two applications are selected due to the following reasons:

(iv) The Speediest app allows customised test sets, close to the consumer experience of Mobile broadband access. These data sets are then saved locally into the test device. The test app itself requires no modification of the handsets.

(v) The test apps are set to continuously run the test in cycle, which make it suitable and convenient as there is less interaction required between tester and the phones.

3.2 Prediction of network parameters

In the second phase, we tried and completed the predictive model for signal strength measurement and download speed prediction.We pre-processed the data based on our model. The pre-processing of data is includes reading of data, creating data-frames and binning of the data on the basis of time slots i.e 1 min. We selected a best model which is used for prediction.

We used Linear regression for prediction which is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables(or independent variables). In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data.

Also, we used deep learning algorithm LSTM, **Long short-term memory (LSTM)** is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).
LSTM networks are well-suited to classifying, processing and making predictions based on time series data, and our prediction model is also based on time-series basic.

3.3 Selection of the model

The prediction models basically depends upon the previous data which is given as input and is time dependent. We started with Linear Regression in which the RMSE levels were quite high, which led us to shift towards deep learning model which is time dependent. RNN models are the best in these cases. We used LSTM as because it is time dependent and works on the basis of time stamps. As it is recurrent, it also helps in reducing the RMSE to a larger extent.

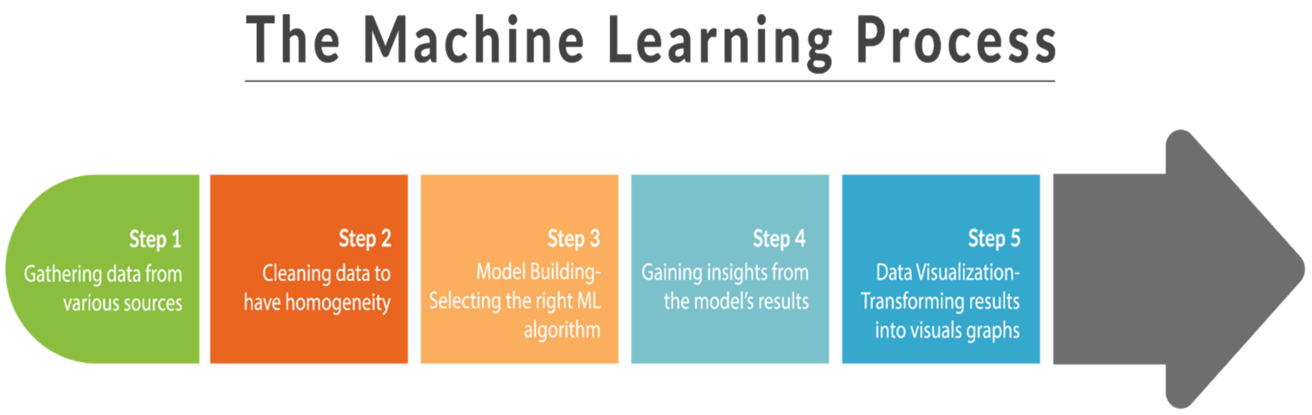

The Machine Learning Process
Step 1
Gathering data from various sources
Step 2
Cleaning data to have homogeneity
Step 3
Model Building-Selecting the right ML algorithm
Step 4
Gaining insights from the model's results
Step 5
Data Visualization-Transforming results into visuals graphs

Fig 3.1 Various steps in Machine learning process

4. Network Analysis

4.1 Development of the mobile application

The data of different operators has been collected through a mobile application developed by us, named as SpeedTest. The basic functionality of this app is that it check the number of web cycles required to upload and download a 17MB photo through various platforms depending upon the user. Various modules such as TelephonyManager, CellInfo(Gsm/LTE), CellSignalStrength etc. For storing the history, we have used SQLite i.e the local storage of the cellphone. The application is made entirely in Java. There is an option to retrieve the data stored in local storage to a .csv file whenever required. This .csv file can be used further for analysis and predictions.

4.2 Data preprocessing and Analysis

We have collected data for the whole, for every 15 minutes,using this application and record upload and download speed.

The preprocessing steps are:

1) Average of the intervals i.e 15mins for each day into a single value.
2) Usage of Regular Expressions to recover the values from the text of the csv files.
3) Separation of the files on the basis of:
    a) Operator names
    b) Operations performed i.e upload, download etc.
    c) On the basis of platforms that is used to perform the operations.
    d) On the basis of network type i.e 3G, 4G etc.

Various analysis has been done throughout the process.

1. Comparison of download speeds in 4G between 2 operators i.e Airtel and Jio.
2. Comparison of upload speeds in 4G between 2 operators i.e Airtel and Jio.
3. Comparison of Signal strengths in 2 platforms using different operators.
4. Comparison of download speeds in 3G and 4G using a single operator.
5. Comparison of upload speeds in 3G and 4G using a single operator

Result and analysis-

Measurement data are collected using a proprietary testing application, MBB explorer for web browsing services. Two metrics parameters, namely network signal strength and speed are considered in this research. Our results show that, on average, 4G networks performed much better than 3G networks throughout across all the measurement areas considered in this research. These observations were found consistent across all mobile operators.According to the analysis, Google Chrome performs better in both the network operators.
It can be seen how 4G has evolved. It shows a way above performance than 3G. In every aspects, 4G beats 3G in terms of speed, signal strength etc.

5. Preparation of Data for speed prediction

This step is the most important as every model has its own specification of data that has to be given as input. The steps followed are:

5.1 Reading data and creating DataFrames

The data is read from the raw dataset given. It is converted to a csv format. CSV format is more preferred as reading and writing data is quite easy and easily accessible.

Code of Reading Data:

```
# Preprocessing of data
# Reading of data

data = pd.read_excel (r'/content/gdrive/My Drive/VideoTestData_23.1.16-29.1.16.xlsx')
df = pd.DataFrame(data)
df = df.iloc[1:]
df.to_csv('/content/gdrive/My Drive/VideoTestData_23.1.16-29.1.16.csv', header=False, index=False)
```

The data frames are created on the basis of the columns required from the dataset given and proper formatting to data time.

Code of Create DataFrame:

```
# create DataFrame

ts = pd.Timestamp
df = pd.read_csv('/content/gdrive/My Drive/VideoTestData_23.1.16-29.1.16.csv')
df = df[['ID','Test Time','Celluar Signal Strength(dBm)','Video Total DL Rate(kbps)']]
df['Test Time'] = pd.to_datetime(df['Test Time'])
```

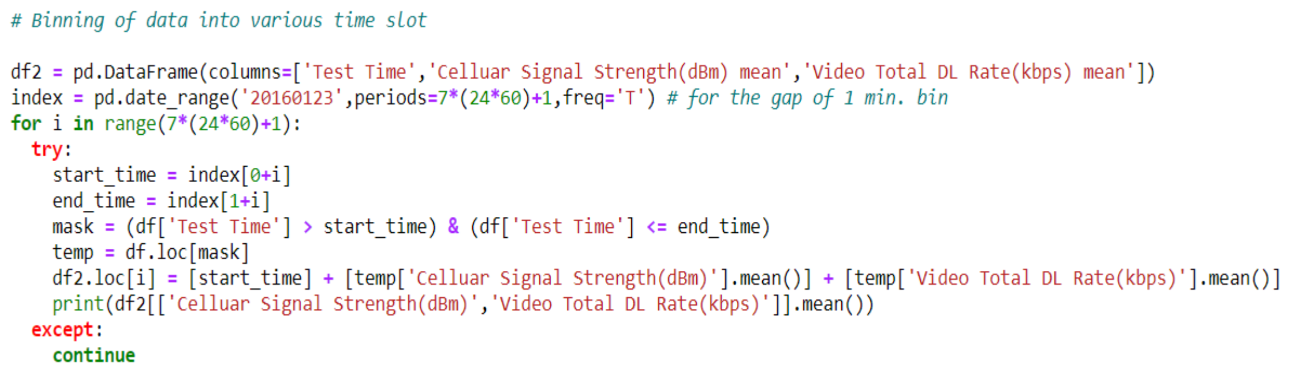
Output of reading and creating DataFrame:

```
0       1612
1       1458
2       1208
3       1304
4       2677
        ...
8301    2265
8302    2262
8303    1201
8304    1401
8305     678
Name: Video Total DL Rate(kbps), Length: 8306, dtype: int64
```

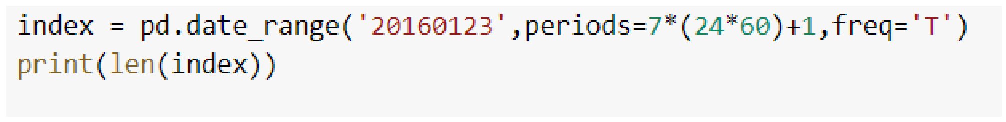
5.2 Binning of the data

The data is binned with the interval of 1 min which will be used as the independent set during providing the input. To bin for a interval, we have taken a mean of all the data points provided during the specified interval. The independent set will be of size 60 i.e will cover an hour of data at a single recursion.

Code of Binning of the Data:



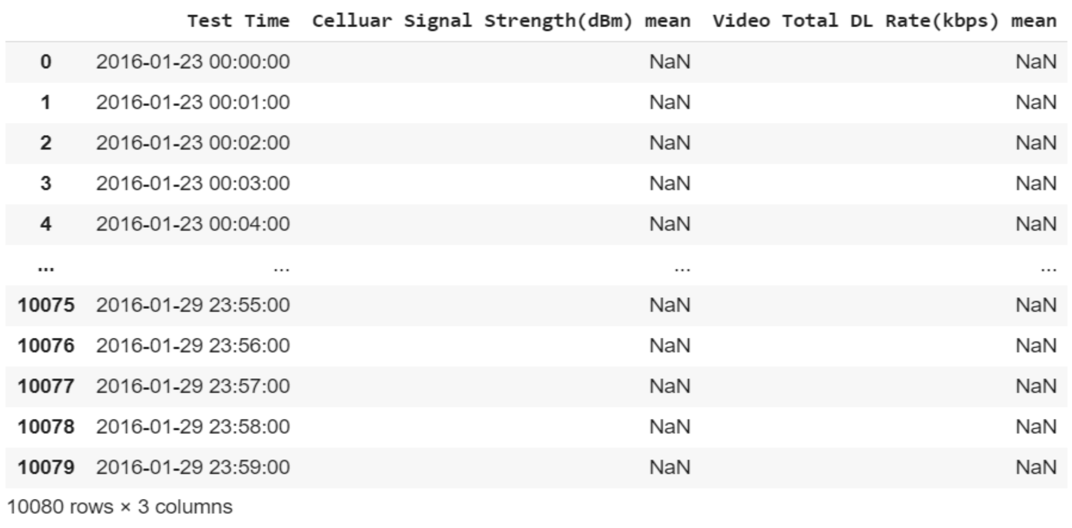
```
# Binning of data into various time slot

df2 = pd.DataFrame(columns=['Test Time','Celluar Signal Strength(dBm) mean','Video Total DL Rate(kbps) mean'])
index = pd.date_range('20160123',periods=7*(24*60)+1,freq='T') # for the gap of 1 min. bin
for i in range(7*(24*60)+1):
  try:
    start_time = index[0+i]
    end_time = index[1+i]
    mask = (df['Test Time'] > start_time) & (df['Test Time'] <= end_time)
    temp = df.loc[mask]
    df2.loc[i] = [start_time] + [temp['Celluar Signal Strength(dBm)'].mean()] + [temp['Video Total DL Rate(kbps)'].mean()]
    print(df2[['Celluar Signal Strength(dBm)','Video Total DL Rate(kbps)']].mean())
  except:
    continue
```



```
index = pd.date_range('20160123',periods=7*(24*60)+1,freq='T')
print(len(index))
```

NOTE: As you see here, We have taken the data of 7th day for prediction as per your suggestion and the time is starting from 00:00:00 to 24:00:00.

Output of Binning of the data:



	Test Time	Celluar Signal Strength(dBm) mean	Video Total DL Rate(kbps) mean
0	2016-01-23 00:00:00	NaN	NaN
1	2016-01-23 00:01:00	NaN	NaN
2	2016-01-23 00:02:00	NaN	NaN
3	2016-01-23 00:03:00	NaN	NaN
4	2016-01-23 00:04:00	NaN	NaN
...	...	...	...
10075	2016-01-29 23:55:00	NaN	NaN
10076	2016-01-29 23:56:00	NaN	NaN
10077	2016-01-29 23:57:00	NaN	NaN
10078	2016-01-29 23:58:00	NaN	NaN
10079	2016-01-29 23:59:00	NaN	NaN

10080 rows × 3 columns

5.3 Data Visualization

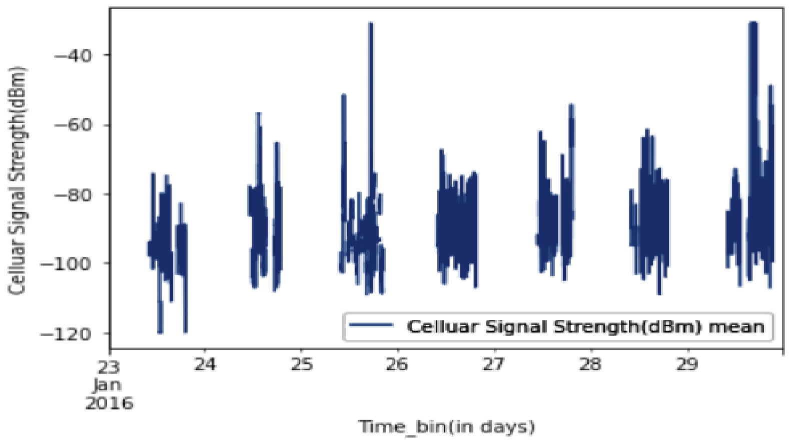
Now,we plot the real data after binning of data using matplotlib.

Code of Data visualization (real data graph)

```
# Plotting of DataFrame to see the trend of data

# for normal plot

ax = plt.gca()
df2.plot(kind='line',x='Test Time',y='Celluar Signal Strength(dBm) mean',ax=ax)
df2.plot(kind='line',x='Test Time',y='Video Total DL Rate(kbps) mean', color='red')
plt.show()
```

```
# for sacttered plot

sns.lmplot( x="Test Time", y="Celluar Signal Strength(dBm) mean", data=df2, fit_reg=False, legend=False, height=8, aspect=1.5, scatter_kws=
{"s": 10,"color":"red"})
sns.lmplot( x="Test Time", y="Video Total DL Rate(kbps) mean", data=df2, fit_reg=False, legend=False, height=8, aspect=1.5, scatter_kws=
{"s": 10,"color":"red"})
```

Output of Data Visualization



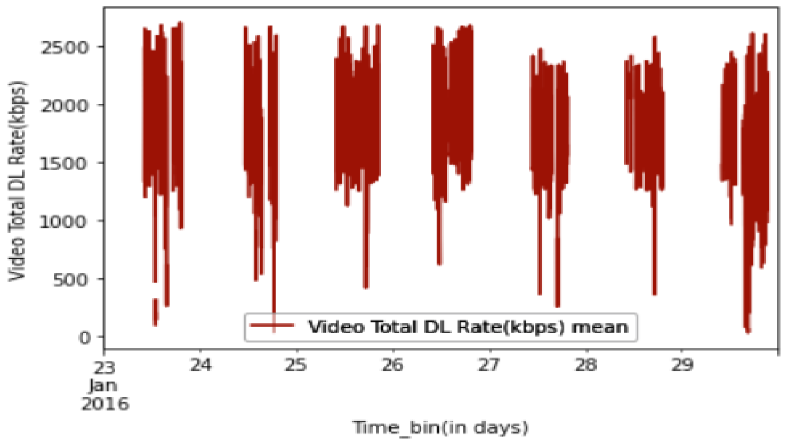
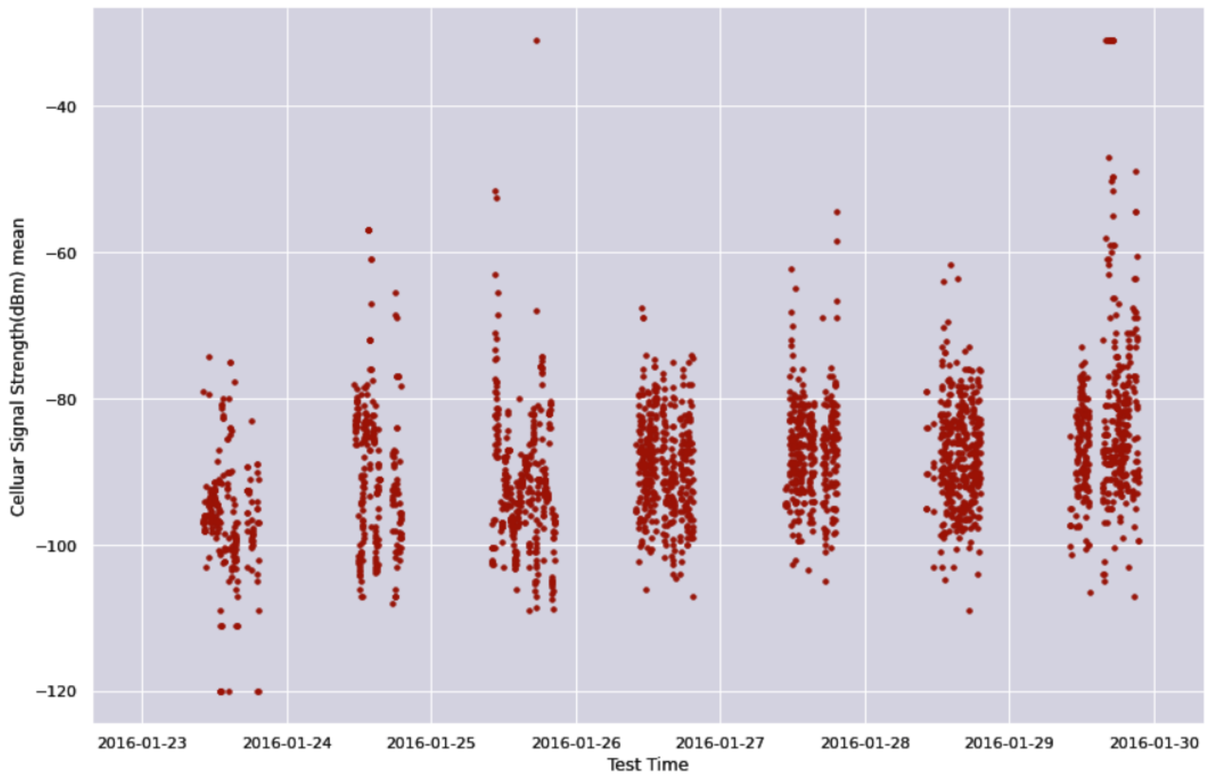


Fig 5.1 Show the normal plot of real data. As we see here, the “cellular Signal Strength” is in negative y-axis while in other fig. “ video download speed” is on positive y-axis.

```
<seaborn.axisgrid.FacetGrid at 0x7fb20fcfbb00>
```



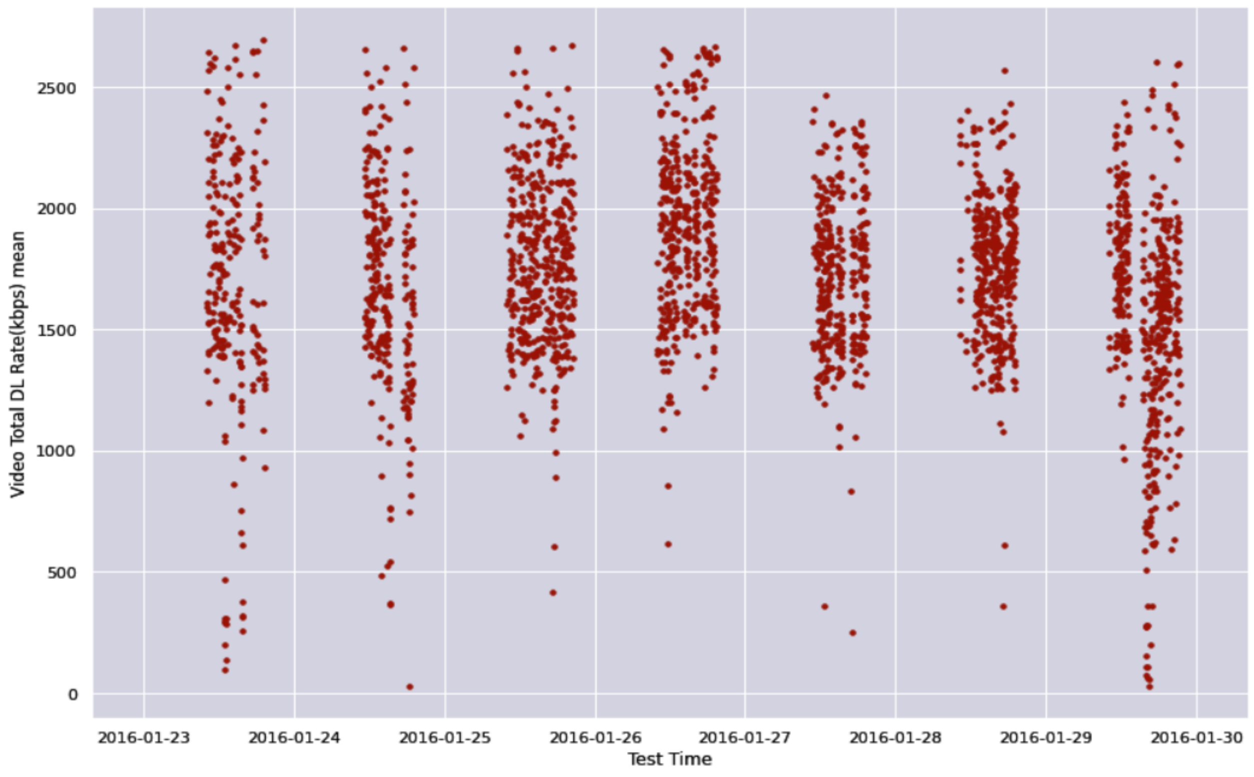


Fig 5.2 Show the scattered plot of real data. As we see here, the “cellular Signal Strength” is in negative y-axis while in other fig. “ video download speed” is on positive y-axis.

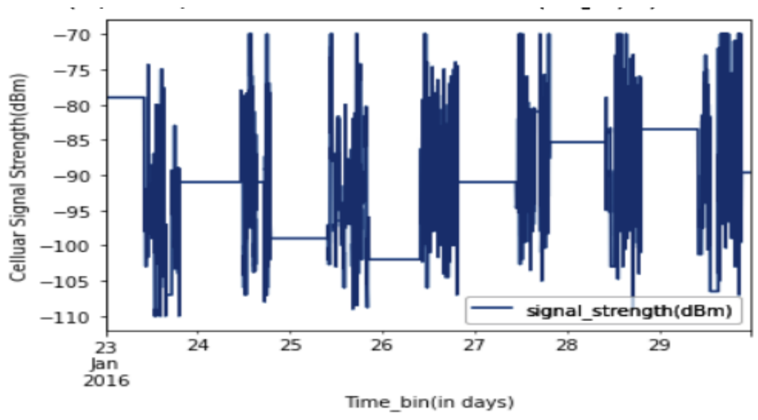
5.4 Filling of NaN value

Then we fill the Not A number(NaN) value by its means value so that there we get a continuous graph (not any gaps between data and time stamp) so that we gets more accurate predictive result.

Code of filling of NaN value

```
# Filling of NaN value by its previous value and cliping

df3 = df2[['Test Time','Celluar Signal Strength(dBm) mean','Video Total DL Rate(kbps) mean']]
df3['signal_strength(dBm)'] = df3['Celluar Signal Strength(dBm) mean'].fillna(method='ffill').fillna(method='bfill')
df3['Video_total_dl_rate(kbps)'] = df3['Video Total DL Rate(kbps) mean'].fillna(method='ffill').fillna(method='bfill')
df3 = df3.drop(columns=['Celluar Signal Strength(dBm) mean','Video Total DL Rate(kbps) mean'])
df3['signal_strength(dBm)'] = df3['signal_strength(dBm)'].clip(-110,-70)
df3['Video_total_dl_rate(kbps)'] = df3['Video_total_dl_rate(kbps)'].clip(1200,2400) # define range so that sudden spike in data gets removed

#ax = plt.gca()
#df3.plot(kind='line',x='Test Time',y='signal_strength(dBm)',ax=ax)
#df3.plot(kind='line',x='Test Time',y='Video_total_dl_rate(kbps)', color='r')

sns.lmplot( x="Test Time", y="signal_strength(dBm)", data=df3, fit_reg=False, legend=False, height=8, aspect=1.5, scatter_kws={"s":
10,"color":"red"})
sns.lmplot( x="Test Time", y="Video_total_dl_rate(kbps)", data=df3, fit_reg=False, legend=False, height=8, aspect=1.5, scatter_kws={"s":
10,"color":"red"})
```

Output of filling of NaN value



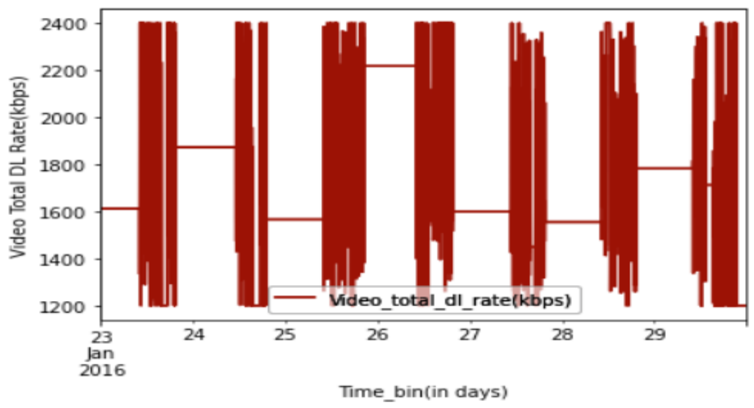
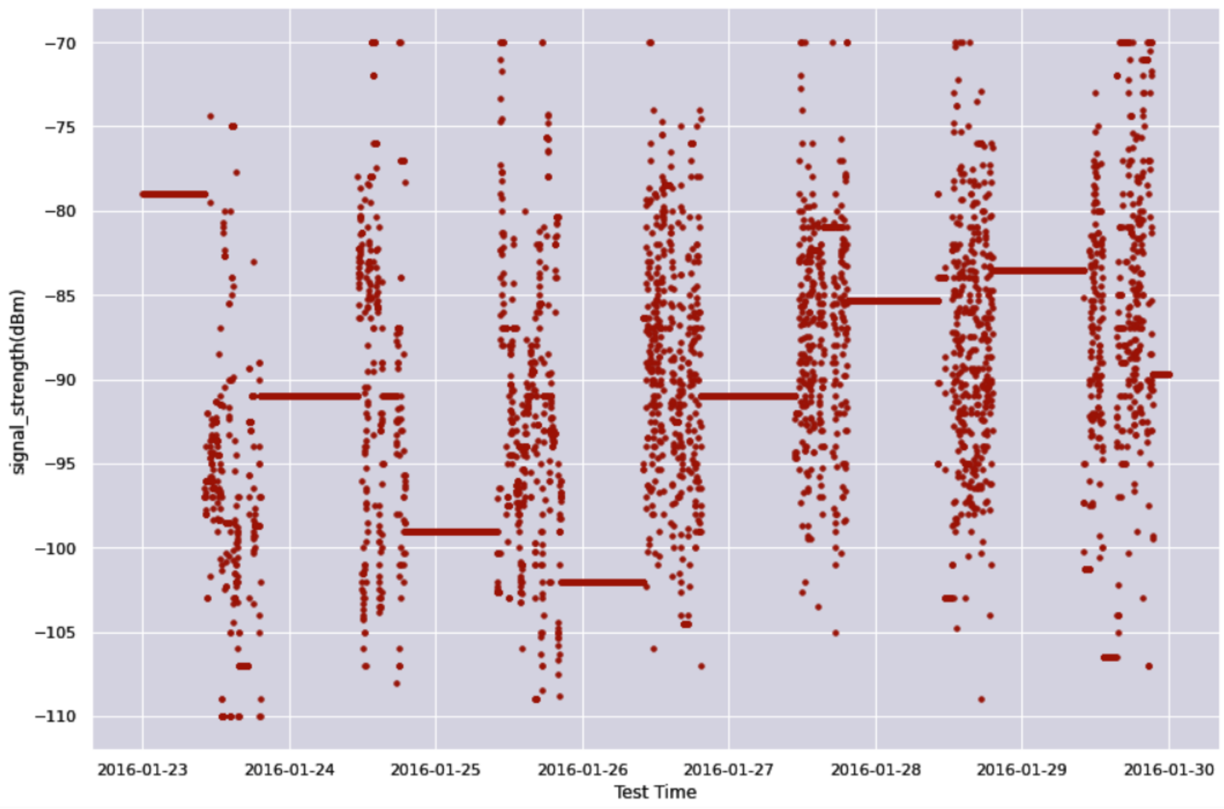


Fig 5.3 show that normal graph plot after filling the NaN value. As we see here, the "cellular Signal Strength" is in negative y-axis while in other fig. " video download speed" is on positive y-axis.

```
<seaborn.axisgrid.FacetGrid at 0x7fb25ad5e0f0>
```



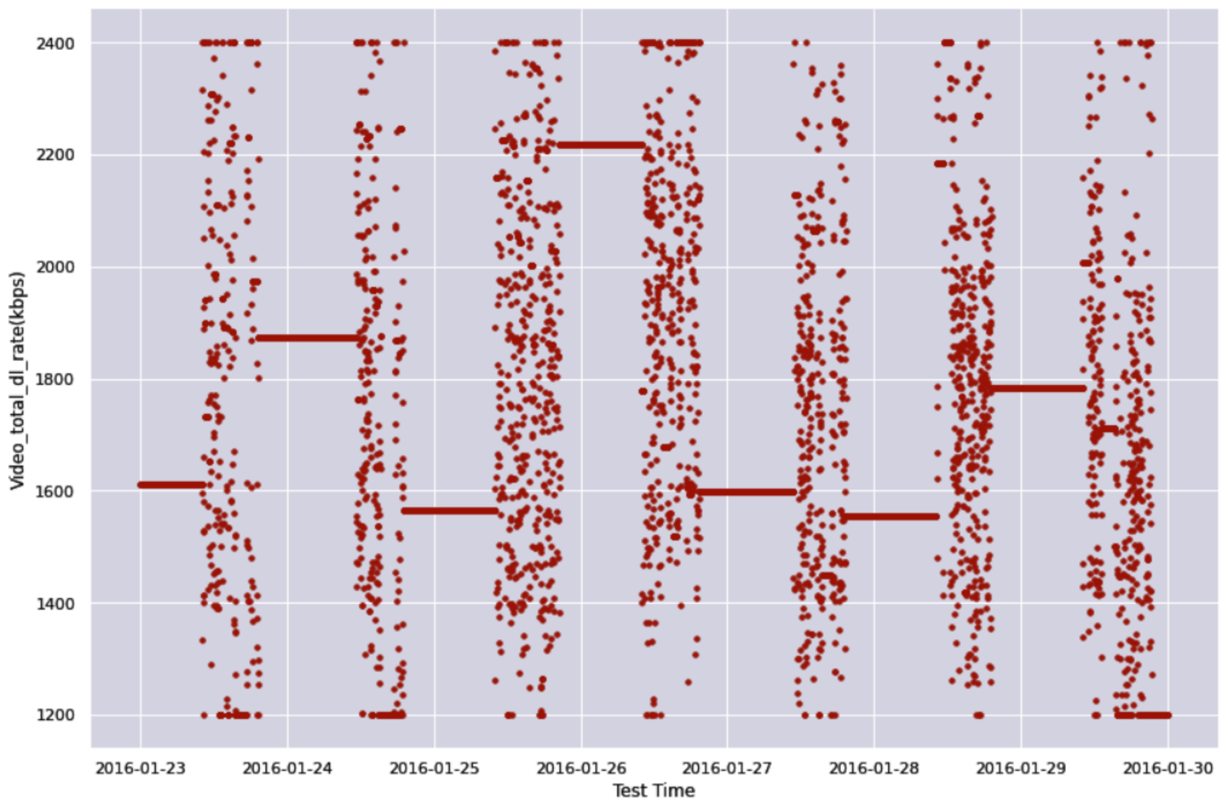


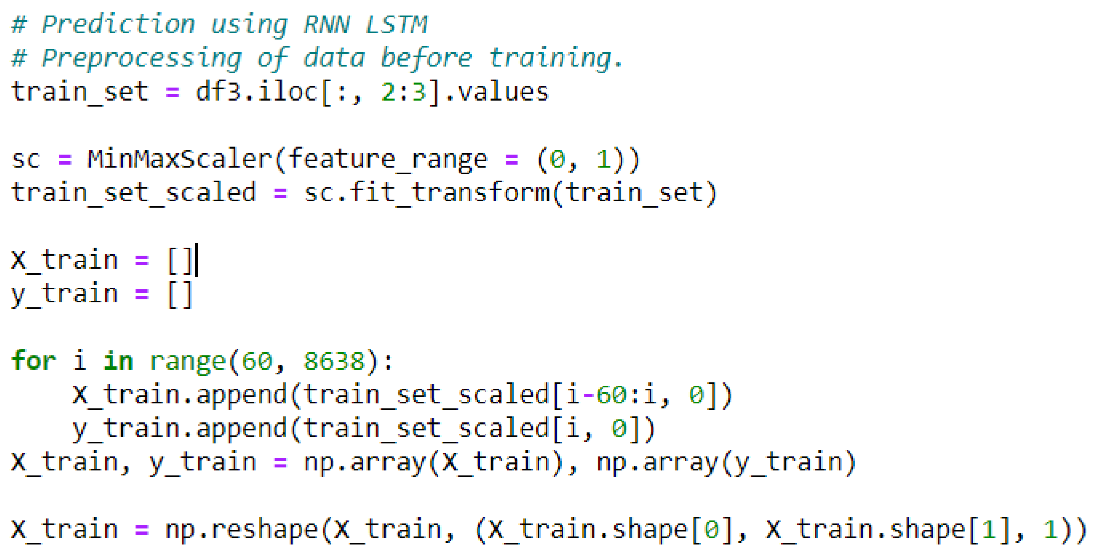
Fig 5.4 show that Scattered graph plot after filling the NaN value. As we see here, the "cellular Signal Strength" is in negative y-axis while in other fig. " video download speed" is on positive y-axis.

6. Prediction using LSTM

6.1 Preprocessing of Data before Training (Scaling the data)

Scaling of data is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. There are various types of normalization. We have choosen to do min-max scaling, which is the simplest method and consists in rescaling the range of features to scale the range in [0, 1] or [−1, 1]. Selecting the target range depends on the nature of the data.

Code of Pre-processing Data using LSTM



```
# Prediction using RNN LSTM
# Preprocessing of data before training.
train_set = df3.iloc[:, 2:3].values

sc = MinMaxScaler(feature_range = (0, 1))
train_set_scaled = sc.fit_transform(train_set)

X_train = []
y_train = []

for i in range(60, 8638):
    X_train.append(train_set_scaled[i-60:i, 0])
    y_train.append(train_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)

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

6.2 Training Phase of LSTM model

After scaling we need to transform the data into a format that is appropriate for modeling with LSTM

Long Short-Term Memory (LSTM) models are a type of recurrent neural network capable of learning sequences of observations.This may make them a network well suited to time series forecasting.An issue with LSTMs is that they can easily overfit training data, reducing their predictive skill.

Dropout is a regulazation method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance.

The number of epochs is the number of complete passes through the training dataset.

The LSTM use the sequence in the training phase : Let's focus on the 1st sequence. The model takes the feature of the time bar at index 0 and it tries to predict the target of the time bar at index 1. Then it takes the feature of the time bar at index 1 and it tries to predict the target of the time bar at index 2, etc. The feature of 2nd sequence is shifted by 1 time bar from the feature of 1st sequence,

the feature of 3rd sequence is shifted by 1 time bar from 2nd sequence, etc. With this procedure, we get many shorter sequences that are shifted by a single time bar.
Note that in classification or regression tasks, we usually have a set of features and a target that we are trying to predict.

Code of Training of Data using LSTM

```
# Training phase of LSTM model.
# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
history = regressor.fit(X_train, y_train, epochs = 100, batch_size = 60)

# Saving and Loading of model.

regressor.save("/content/gdrive/My Drive/project/regressor.h5")
regressor = load_model("/content/gdrive/My Drive/project/regressor.h5")
```

Output of Training of Data using LSTM

```
Epoch 1/100
143/143 [==============================] - 21s 146ms/step - loss: 0.0237
Epoch 2/100
143/143 [==============================] - 21s 148ms/step - loss: 0.0140
Epoch 3/100
143/143 [==============================] - 21s 148ms/step - loss: 0.0124
```

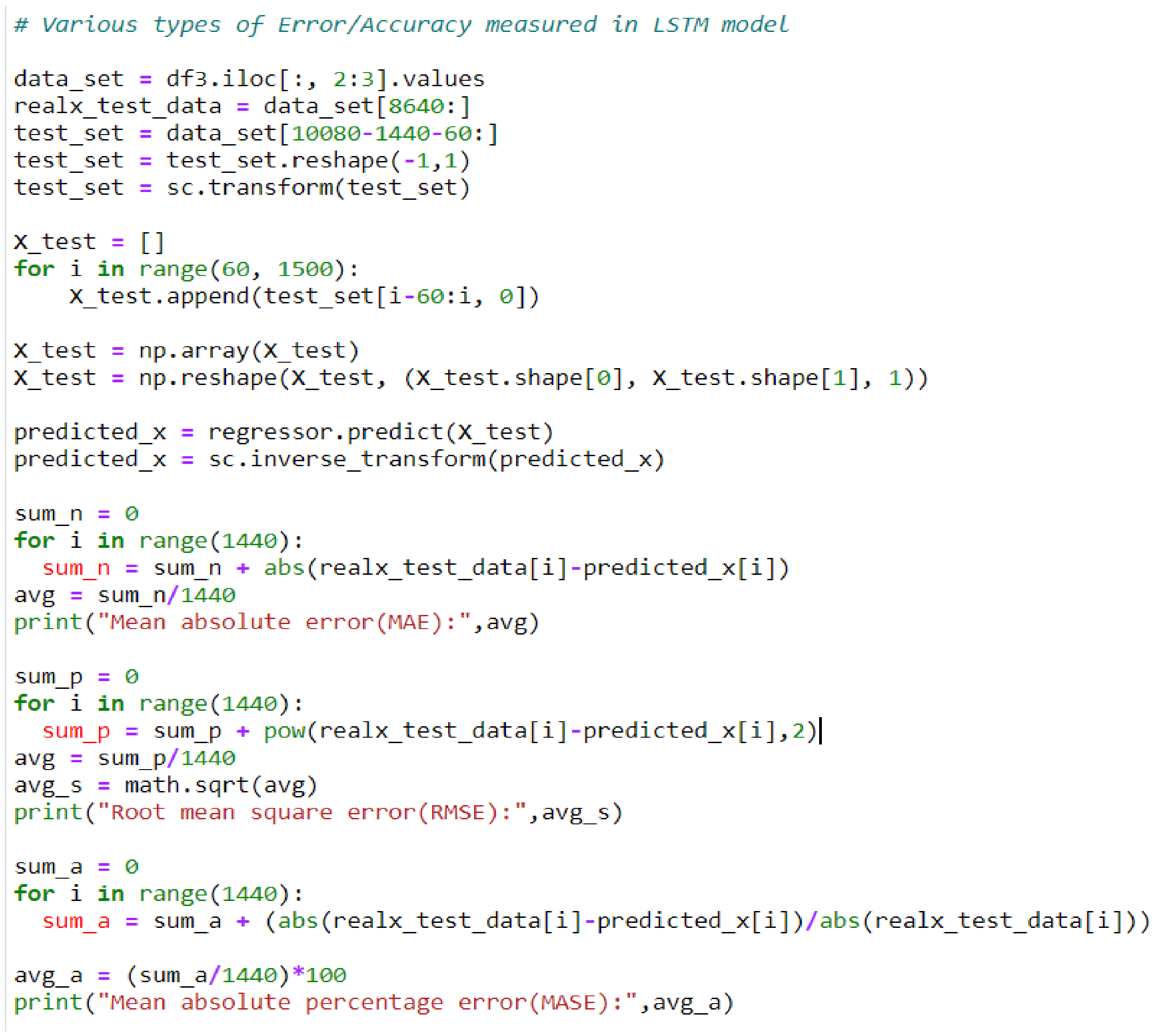
```
....        .......     .......     .......     .......     .......

....        .......     .......     .......     .......     .......
```

```
Epoch 62/100
143/143 [==============================] - 22s 151ms/step - loss: 0.0063
Epoch 63/100
143/143 [==============================] - 22s 150ms/step - loss: 0.0062
Epoch 64/100
 42/143 [=======>......................] - ETA: 15s - loss: 0.0061
```

6.3 Accuracy measure in LSTM model

After training phase ,you take the entire sequence as input to your RNN and extract the final output produced. You can then compare your predicted output with the expected output,for each input in the sequence, you want to predict an output. You'll then end up with a sequence of outputs. So you simply compare the predicted sequence with the expected sequence and measure the error. Usually, people train RNNs by measuring the cross-entropy error for each output pair in the predicted and expected sequence, and then take the mean of that value. W.r.t. measuring accuracy, that largely depends on your application.

Code of finding error and accuracy measured in LSTM



```
# Various types of Error/Accuracy measured in LSTM model

data_set = df3.iloc[:, 2:3].values
realx_test_data = data_set[8640:]
test_set = data_set[10080-1440-60:]
test_set = test_set.reshape(-1,1)
test_set = sc.transform(test_set)

X_test = []
for i in range(60, 1500):
    X_test.append(test_set[i-60:i, 0])

X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

predicted_x = regressor.predict(X_test)
predicted_x = sc.inverse_transform(predicted_x)

sum_n = 0
for i in range(1440):
  sum_n = sum_n + abs(realx_test_data[i]-predicted_x[i])
avg = sum_n/1440
print("Mean absolute error(MAE):",avg)

sum_p = 0
for i in range(1440):
  sum_p = sum_p + pow(realx_test_data[i]-predicted_x[i],2)
avg = sum_p/1440
avg_s = math.sqrt(avg)
print("Root mean square error(RMSE):",avg_s)

sum_a = 0
for i in range(1440):
  sum_a = sum_a + (abs(realx_test_data[i]-predicted_x[i])/abs(realx_test_data[i]))

avg_a = (sum_a/1440)*100
print("Mean absolute percentage error(MASE):",avg_a)
```

6.4 Graphical Analysis of Data

Code of Plotting real data vs predicted data using LSTM

```
# plot of real data vs predicted data

df['realx_test_data'] = pd.DataFrame(data_set[8640:])
df['Video_dl_speed(kbps)'] = pd.DataFrame(predicted_x)
df['Time Bin(gap of 1 min.)'] = pd.DataFrame([int(x) for x in range(1440)])

# normal plot
|
plt.figure(figsize=(6.4,6.0),dpi=100)
plt.plot(realx_test_data, color = 'red', label = 'Real data')
plt.plot(predicted_x, color = 'blue', label = 'Predicted data')
plt.title('Video Total Download Speed')
plt.xlabel('Time_bin(Gap of 1 min.)')
plt.ylabel('Download_speed(kbps)')
plt.legend()
plt.show()

# scattered plot

fig, axs = plt.subplots(ncols=1)
fig.set_size_inches(16.5, 8.5)
sns.regplot(x='Time Bin(gap of 1 min.)', y='realx_test_data', data=df, fit_reg = False, scatter_kws={"s": 50})
sns.regplot(x='Time Bin(gap of 1 min.)', y='Video_dl_speed(kbps)', data=df, marker="+", fit_reg = False, scatter_kws={"s": 50})
```

Output of Plotting real data vs predicted data using LSTM

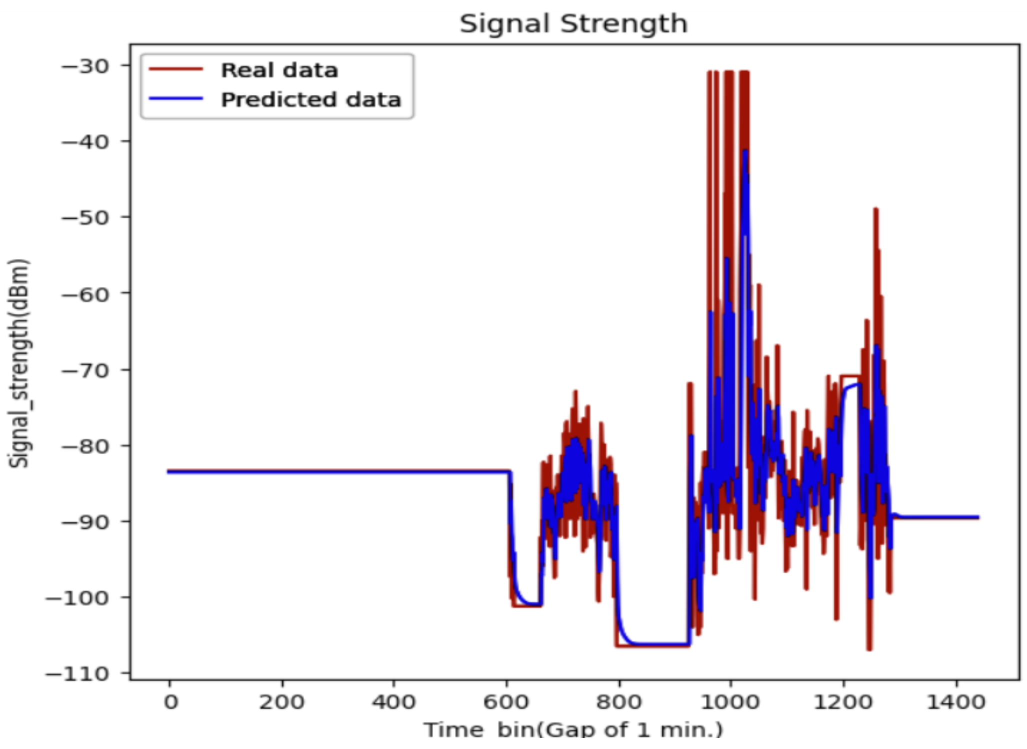


Fig 6.1 Shown that normal graph plot of real data vs predicted data using LSTM. As you see that signal strength is in negative y-axis.

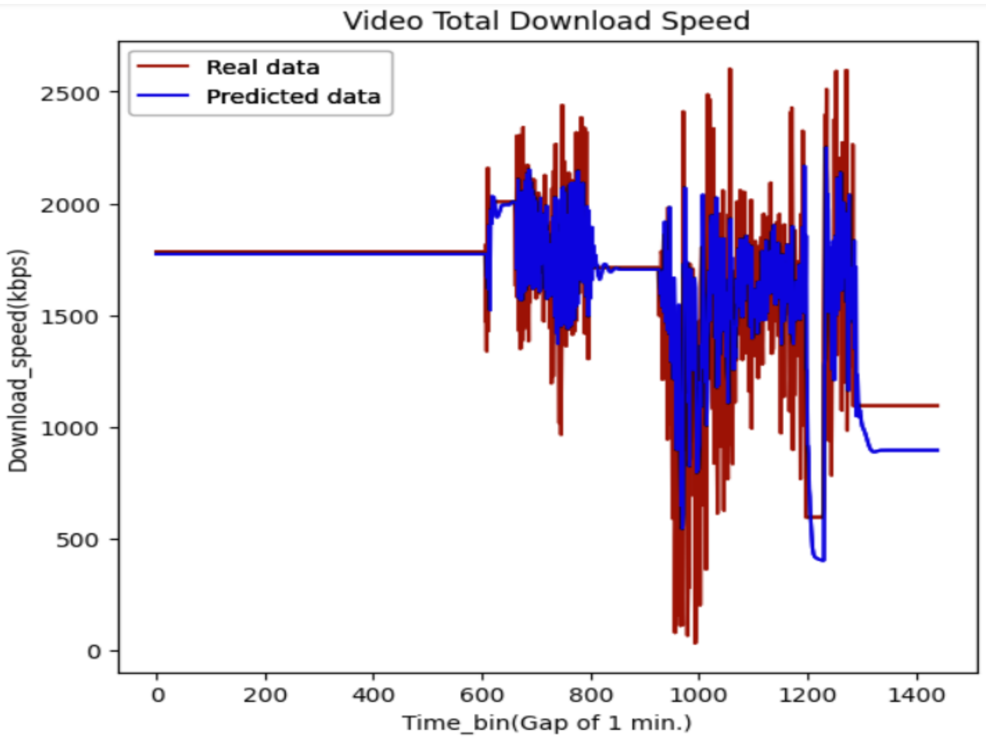


Fig 6.2 Shown that normal graph plot of real data vs predicted data using LSTM. As you see that Download speed is in positive y-axis.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2155a39b0>
```

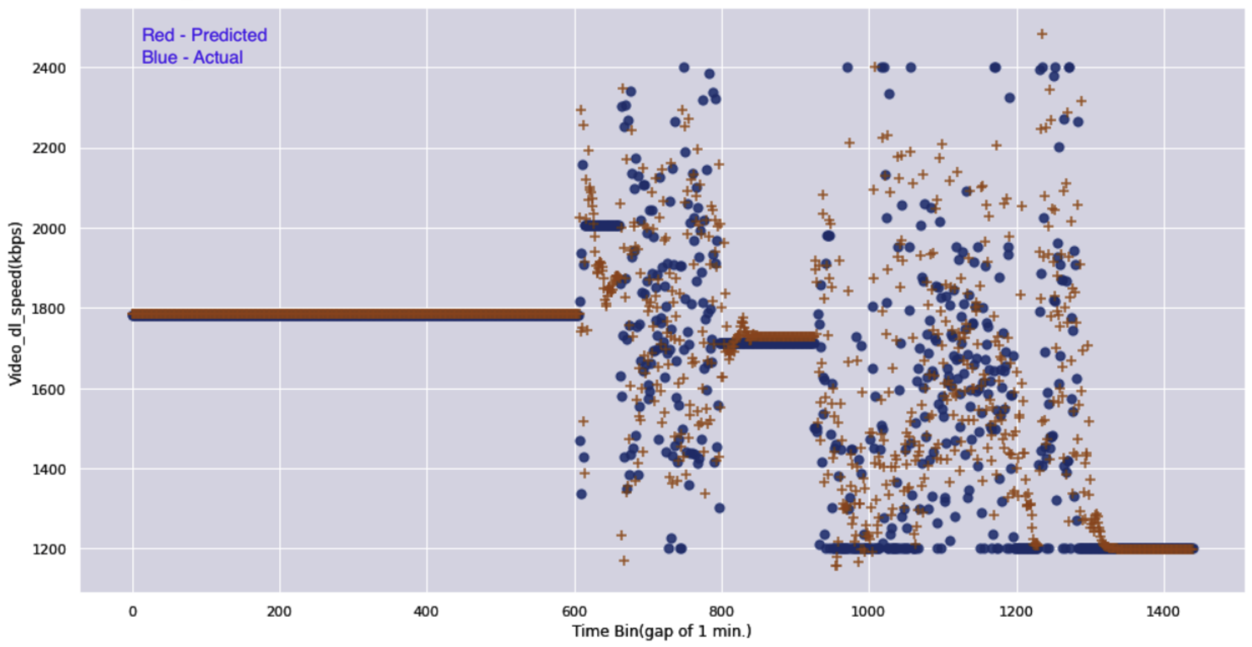


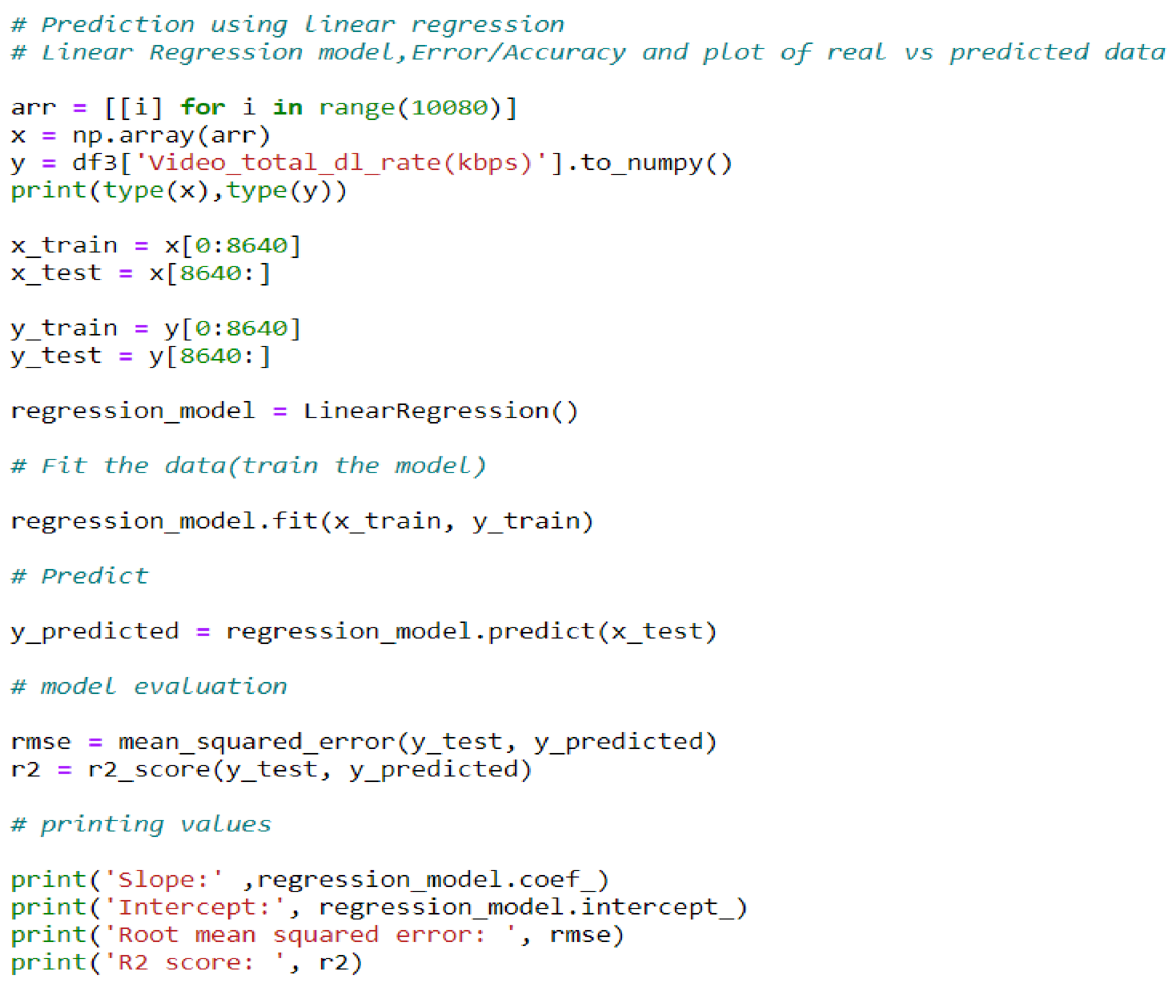
Fig 6.3 Shown that Scattered graph plot of real data vs predicted data using LSTM. As you see that Video Download speed is in positive y-axis.

7. Application of Linear Regression

7.1 Linear regression Model Training and Model evaluation

For Linear Regression, we have to pre-process the data. As it has no requirements of dependent and independent datasets, we directly input the data of download speed data. We fit the data into the Linear regression model.

Code of training model using Linear regression



```
# Prediction using linear regression
# Linear Regression model,Error/Accuracy and plot of real vs predicted data

arr = [[i] for i in range(10080)]
x = np.array(arr)
y = df3['Video_total_dl_rate(kbps)'].to_numpy()
print(type(x),type(y))

x_train = x[0:8640]
x_test = x[8640:]

y_train = y[0:8640]
y_test = y[8640:]

regression_model = LinearRegression()

# Fit the data(train the model)

regression_model.fit(x_train, y_train)

# Predict

y_predicted = regression_model.predict(x_test)

# model evaluation

rmse = mean_squared_error(y_test, y_predicted)
r2 = r2_score(y_test, y_predicted)

# printing values

print('Slope:' ,regression_model.coef_)
print('Intercept:', regression_model.intercept_)
print('Root mean squared error: ', rmse)
print('R2 score: ', r2)
```

7.2 Graphical Analysis of Data

The comparison between the actual and the predicted graphs for download speed and signal strength using Linear Regression is given below. The red graphs is the predicted graph while the blue graph is the actual one. It can be easily seen that the error margin is quite high while the graph predicted is not propery aligned with the real data graph.

Code

```
# plotting values

# data points
# plt.plot(x,y)
# plt.show()

plt.figure(figsize=(6.4,6.0),dpi=100)
plt.plot(x_test,y_test)
plt.plot(x_test,y_predicted,color='r')
plt.title('Video Total Download Speed')
plt.xlabel('Time_bin(Gap of 1 min.)')
plt.ylabel('Download_speed(kbps)')
plt.legend()
```

Output

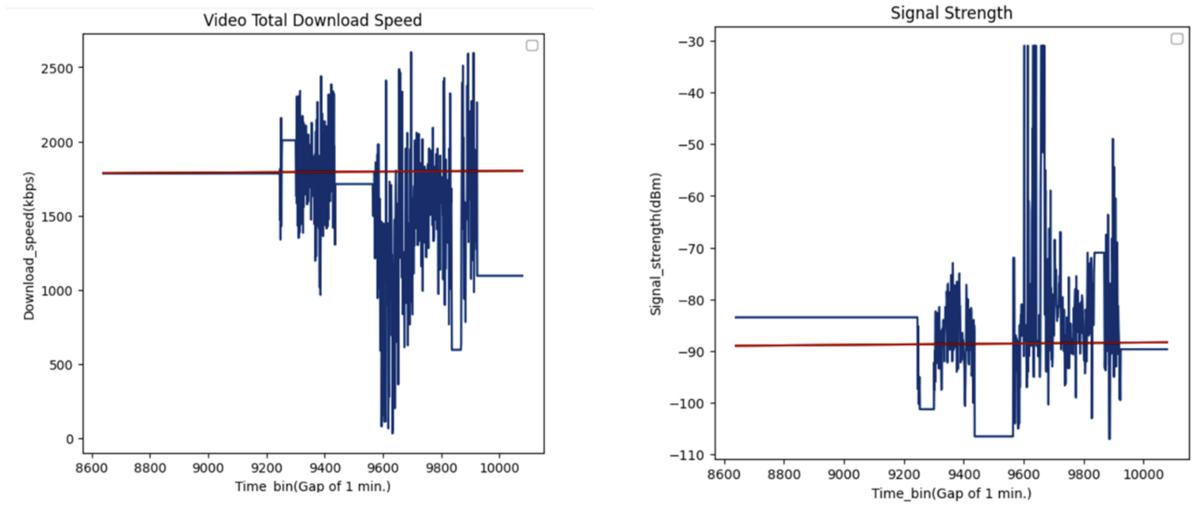


Fig 7.1 Shown that normal graph plot of real data vs predicted data using Linear regression. As you see that signal strength is in negative y-axis and Download speed on positive y-axis

8. Error analysis

8.1 Error analysis of LSTM and Linear regression

The error and graphical scores for Linear Regression is given below with error such as RMSE and R2 score.The errors can be seen being very high.

Parameters (Linear Regression)	Download Speed	Signal Strength
Slope	0.0107	0.00046742
Intercept	1693.267	-93.03
RMSE	181872.51	119.99
R2 score	-0.260	-0.0511

The errors obtained for LSTM are listed below. We tried to find out 3 types of errors such as MAE,RMSE and MASE .The errors can be seen to be reduced to a large extent.

Parameters (LSTM)	Download Speed	Signal Strength
MAE	104.22	2.477
RMSE	215.64	6.06
MASE	6.81	3.66

The main reason for huge difference in RMSE(root mean squared error) between LSTM and Linear regression is that LSTM can not only process single data points, but also entire sequences of data. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, and our prediction model is also based on time-series basic.

In contrast , We used Linear regression for prediction which is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables(or independent variables). In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data.

And in our Dataset ,"Download speed rate" column is independent of all given column in dataset like "ID" ; "Test Time" ; "AppVersion Name" ; "IMEI" ; "UE Model" ; Address etc.
As you known that we predict that download speed rate of video which is independent of all given column.

9.Conclusion and Future Aspects

9.1 Conclusion

We have successfully implemented the LSTM model in our project for singal strength and download speed prediction. The RMSE is also reduced to a large extent in the due course of time. Over 50 epochs, the RMSE levels can be reduced further. The parameters taken were in small numbers. With data in bigger numbers, the RMSE will be further reduced. Practical application of this model still has many blockers and restrictions. One of the major factor is the weather, which has a larger impact on the network strength and also the connectivity of the handset. For practical application, many more factors have to be kept in mind which has to be given in input accordingly with the independent dataset of time, which can also be said as the future scope of this project. We have also learned about two models in detail and in which condtions one prevails over the other. In the future, to make our algorithm more practical, we can add every parameter which effects the network strength and then to implement in as a notification sender in all the mobile OS available in our daily market.

9.2 Future Scope

The future goals are as follows:

1) To make a predictive model which can predict the dimensions of the network such as speed, connectivity in real time analysis as done in Stock exchange market prediction.

2) To come up with more large no of dataset and predicted our channel capacity more and more accuretely.

3) A notification feature will be added to the model which will notify the users about the dimensions of the network according to which the user can change his/her schedule.

4) Other factors relating to the consumer experience of using mobile services such as price, traffic management policies, data allowances, customer service, billing etc. can also be considered.

5) Analysis of mobile broadband services delivered to other devices, the performance of mobile virtual network operators (MNVOs) may be examined in future.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[2] Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[3] Roderforte. , Zingo. Zhang, and Yong. Li, “Using lstm and neural network methods for traffic flow prediction,” in *Association of Automation (YAC), Youth Academic Annual Conference of*. IEEE, 2016

[4] Chunxiao Jiang, Haijun Zhang, Yong and Lajos Hanzo. Machine learning paradigms for next generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, 2017.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[6] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[7] Atlas Iyer, and Iiten. Stoica, “Celliq: real-time cellular network analytics at scale,” in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, 2015, pp. 309–322.

[8] Maltsev, Pudeyev, Bolotin et al., “Channel modeling and characterization,” MiWEBA FP7-ICT-608637/V1.0, 2014.

[9] Yong Li, and Bong Juang, “Power of deep learning for channel estimation and signal detection in OFDM systems,” *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117,

[10] Goodfellow, Bengio, Courville, and Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[11] S. Ramanathan and H. Perry, "Method and system for evaluating user-perceived network performance," ed: Google Patents, 2000.

[12] Tensorflow ,keras and scikit-learn documentation “https://www.tensorflow.org/resources/learn-ml ,

“https://keras.io/guides/ , “https://scikit-learn.org/stable/tutorial/index.html