BITCOIN PRICE PREDICTION USING GCP AND DEEP LEARNING

Bitcoin is the first digital decentralized cryptocurrency that has shown a significant increase in market capitalization in recent years. The objective of this project is to determine the predictable price direction of Bitcoin in USD by deep learning techniques and google cloud platform.

Design and Analysis

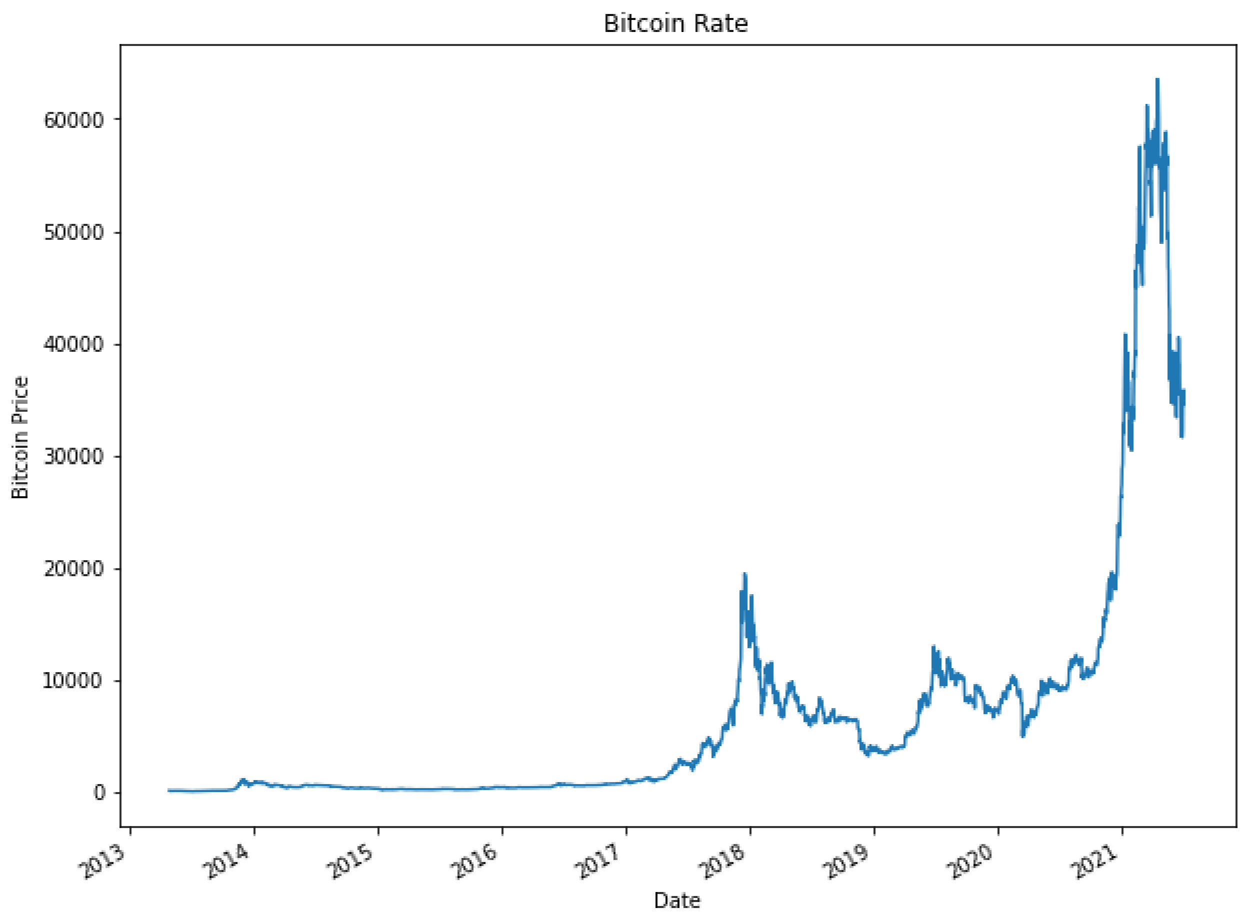
The dataset that we have, looks like following :

D	E	F	G	H	I	J
Date	High	Low	Open	Close	Volume	Marketcap
2013-04-29 23:59:59	147.4880066	134	134.4440002	144.5399933	0	1603768865
2013-04-30 23:59:59	146.9299927	134.0500031	144	139	0	1542813125
2013-05-01 23:59:59	139.8899994	107.7200012	139	116.9899979	0	1298954594
2013-05-02 23:59:59	125.5999985	92.2818985	116.3799973	105.2099991	0	1168517495
2013-05-03 23:59:59	108.1279984	79.09999847	106.25	97.75	0	1085995169
2013-05-04 23:59:59	115	92.5	98.09999847	112.5	0	1250316563
2013-05-05 23:59:59	118.8000031	107.1429977	112.9000015	115.9100037	0	1288693176
2013-05-06 23:59:59	124.663002	106.6399994	115.9800034	112.3000031	0	1249023060
2013-05-07 23:59:59	113.4440002	97.69999695	112.25	111.5	0	1240593600
2013-05-08 23:59:59	115.7799988	109.5999985	109.5999985	113.5660019	0	1264049202
2013-05-09 23:59:59	113.4599991	109.2600021	113.1999969	112.6699982	0	1254535382
2013-05-10 23:59:59	122	111.5510025	112.7990036	117.1999969	0	1305479080
2013-05-11 23:59:59	118.6790009	113.0100021	117.6999969	115.2429962	0	1284207489
2013-05-12 23:59:59	117.4489975	113.4349976	115.6399994	115	0	1281982625
2013-05-13 23:59:59	118.6989975	114.5	114.8199997	117.9800034	0	1315710011
2013-05-14 23:59:59	119.8000031	110.25	117.9800034	111.5	0	1243874488
2013-05-15 23:59:59	115.8099976	103.5	111.4000015	114.2200012	0	1274623813
2013-05-16 23:59:59	118.7600021	112.1999969	114.2200012	118.7600021	0	1325726787
2013-05-17 23:59:59	125.3000031	116.5709991	118.2099991	123.0149994	0	1373723882
2013-05-18 23:59:59	125.25	122.3000031	123.5	123.4980011	0	1379574546

- Date : date of observation
- Open : Opening price on the given day

- High : Highest price on the given day
- Low : Lowest price on the given day
- Close : Closing price on the given day
- Volume : Volume of transactions on the given day
- Market Cap : Market capitalization in USD

For building the model, only Closing price of bitcoin would be considered. The following figure represents the Bitcoin Closing Price from 2013 to 2021.



Long Short-Term Memory networks, or **LSTMs** for short, can be applied to time series forecasting. There are many types of LSTM models that can be used for each specific type of time series forecasting problem.

LSTM models for univariate time series forecasting :

1. Vanilla LSTM
2. Stacked LSTM
3. Bidirectional LSTM
4. CNN LSTM
5. ConvLSTM

Data Preprocessing

- ➢ LSTM models are sensitive to the scale of the data. So we apply MinMax scaler.

- ➢ Splitting the dataset into 65% training set and 35% testing set.

- ➢ Before a univariate series can be modeled, it must be prepared.

The LSTM model will learn a function that maps a sequence of past observations as input to an output observation. As such, the sequence of observations must be transformed into multiple examples from which the LSTM can learn.

For Example,

Consider a given univariate sequence:

[10, 20, 30, 40, 50, 60, 70, 80, 90]

We can divide the sequence into multiple input/output patterns called samples, where three time steps are used as input and one time step is used as output.

X	y
10, 20, 30	40
20, 30, 40	50
30, 40, 50	60

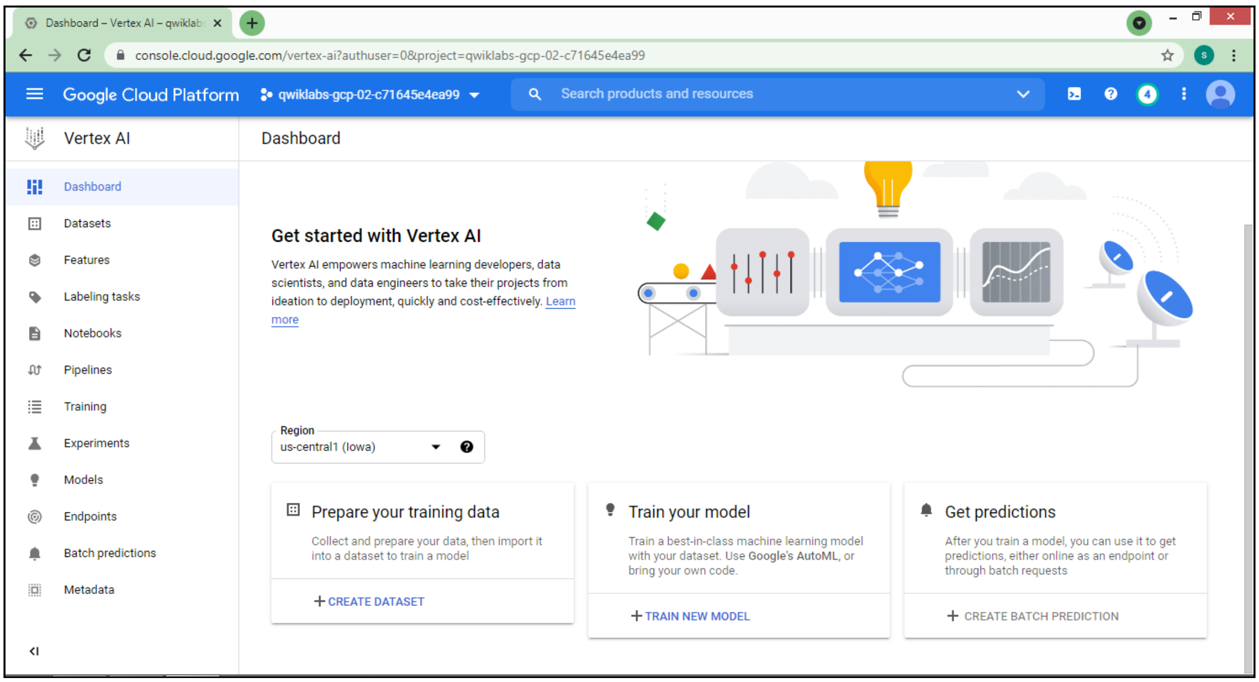
…

In our case, time_step = 100 which means number of input features equal to 100.

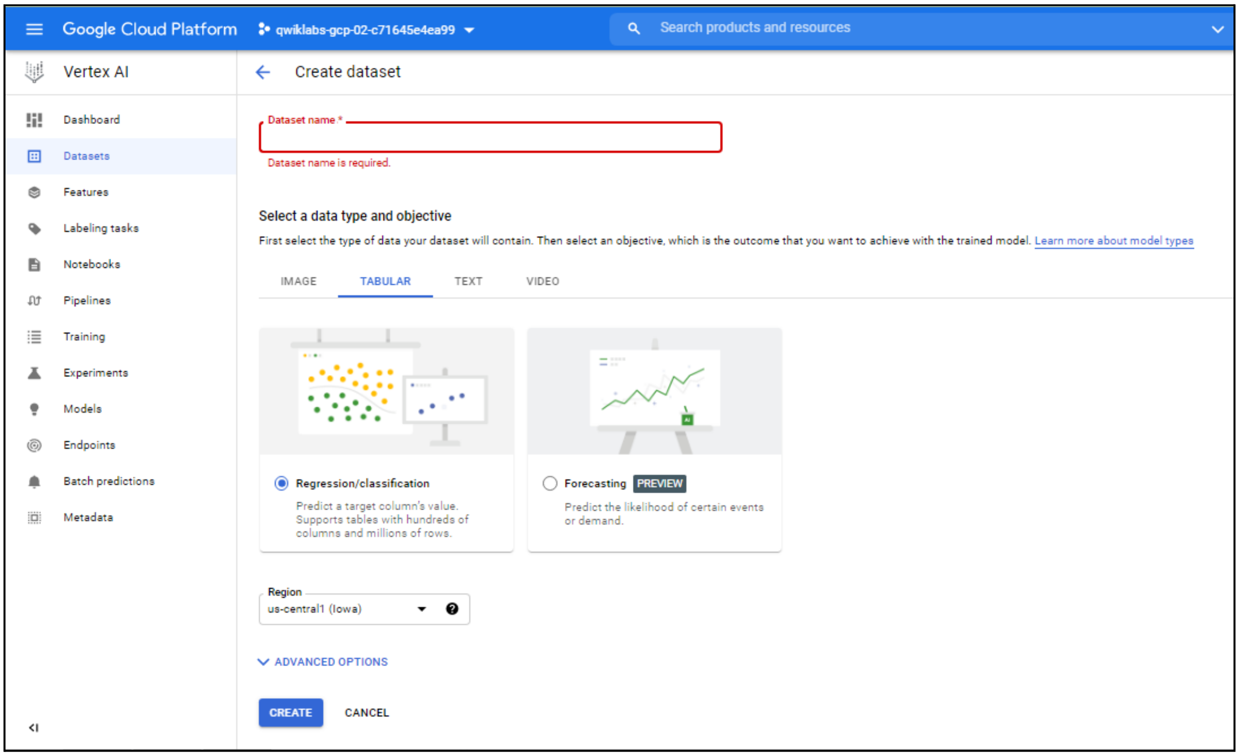
Implementation

For this project, we are going to use the **Stacked LSTM** model because its performance is better than the other LSTM models and ARIMA model.

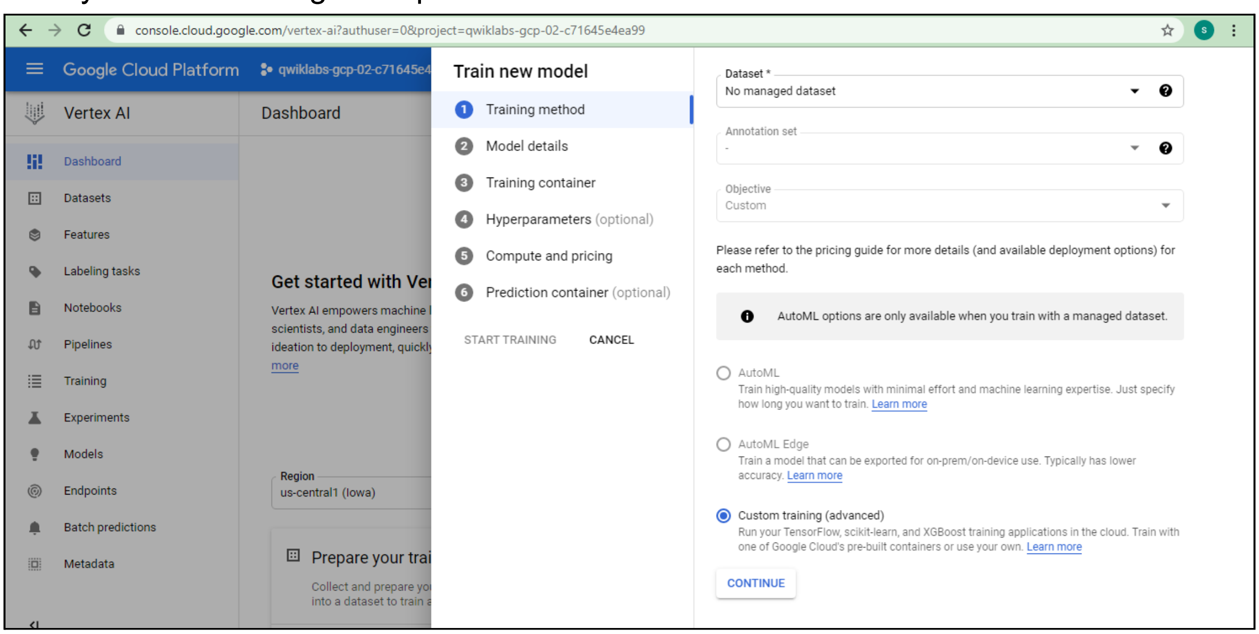
Here is the web UI of the Vertex AI



You can create dataset



Train your model and get the prediction



For training and testing the model, we are going to use custom code.

Importing Libraries

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

### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

Defining Model

The model expects the input shape to be three-dimensional with [samples, timesteps, features], therefore, we must reshape the single input sample before making the prediction.

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

Multiple hidden LSTM layers can be stacked one on top of another in what is referred to as a Stacked LSTM model.
An LSTM layer requires a three-dimensional input and LSTMs by default will produce a two-dimensional output as an interpretation from the end of the sequence. We can address this by having the LSTM output a value for each time step in the input data by setting the **`return_sequences=True`** argument on the layer. This allows us to have 3D output from hidden LSTM layer as input to the next.

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(n_steps,
n_features)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

Key in the definition is the shape of the input; that is what the model expects as input for each sample in terms of the number of time steps and the number of features.
We are working with a univariate series, so the number of features is one, for one variable.
The number of time steps as input is the number we chose when preparing our dataset.
The shape of the input for each sample is specified in the **input_shape** argument on the definition of the first hidden layer.

In this case, we define a model with 50 LSTM units in the hidden layers and an output layer that predicts a single numerical value.

The model is fit using the efficient Adam version of stochastic gradient descent and optimized using the mean squared error, or ‘mse‘ loss function.

Once the model is defined, we can fit it on the training dataset.

```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=
100,batch_size=64,verbose=1)
```

After the model is fit, we can use it to make a prediction.

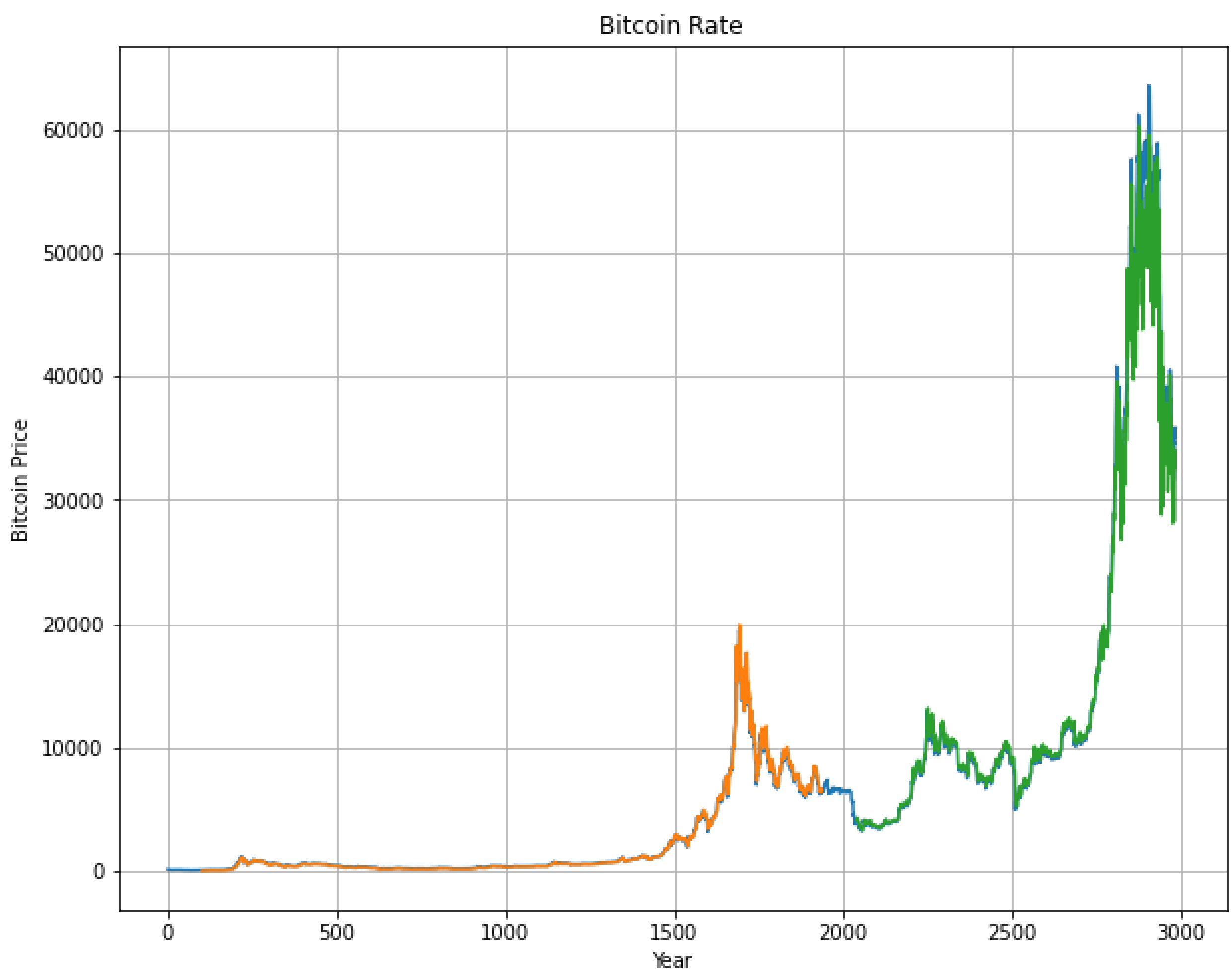
```
train_predict= model.predict(X_train)

test_predict= model.predict(X_test)
```

We can predict the next value in the sequence by providing the input. Here we want to predict the Bitcoin price rate from July 2021 to June 2022.
Hence we first take the last 100 days bitcoin price from the test data and prepare our dataset. We will use this dataset to predict the price for next 365 days.

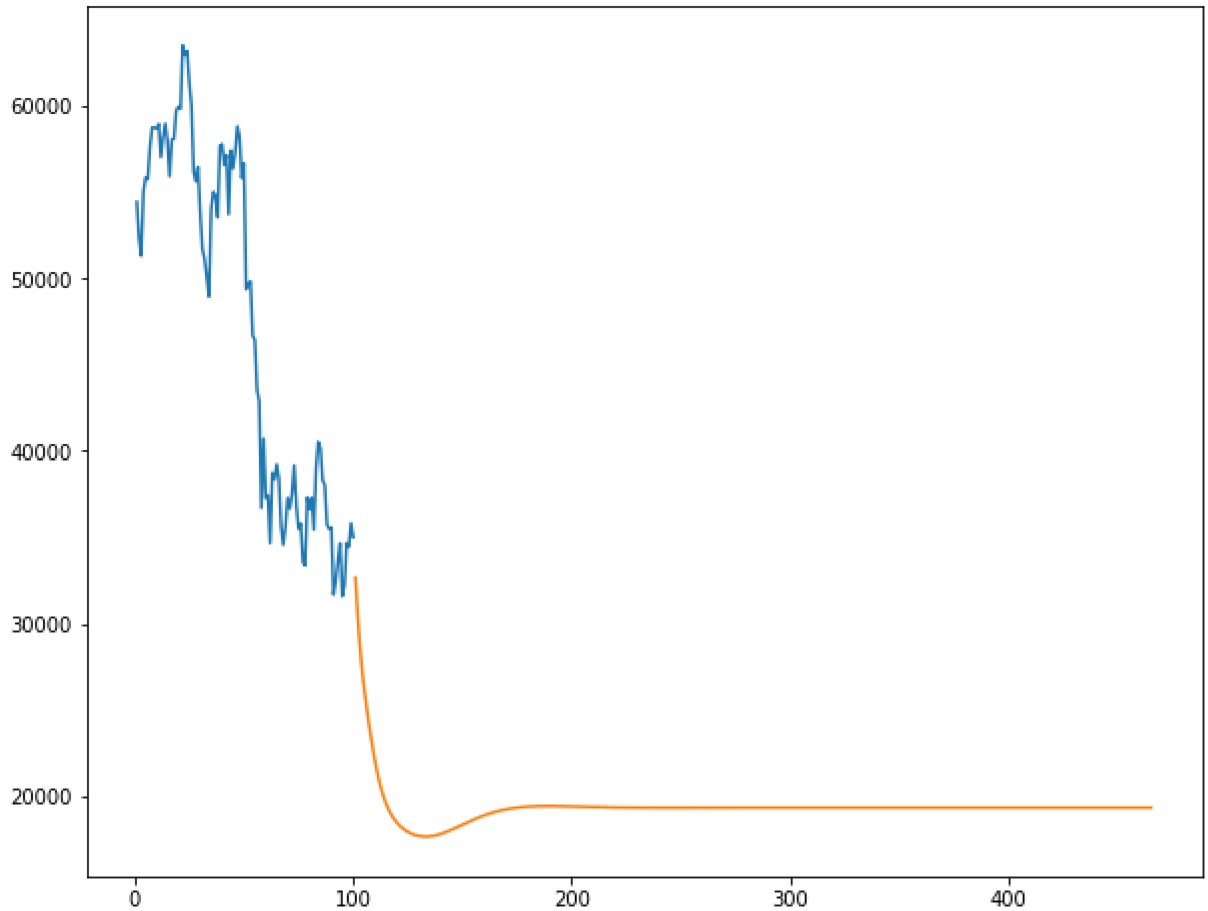
Result

The following figure represents the performance of the model.

Blue Line : represents actual data
Orange Line : represents prediction on the training data
Green Line : represents prediction on the test data



The prediction of the model for the next 365 days ie from July 2021 to June 2022 is as follows:



References

[1]https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

[2]https://towardsdatascience.com/giving-vertex-ai-the-new-unified-ml-platform-on-google-cloud-a-spin-35e0f3852f25