Time series forecasting is predicting target variable according to time frame. For that I choose G-research crypto forecasting dataset from Kaggle. As there are 14 currencies data in the dataset, I will be using the data of only one currency I-e bitcoin currency data

So, in the dataset data is represented by timestamp (Unix time) column which is number of seconds that have elapsed since the Unix epoch, excluding leap seconds. The Unix epoch is 00:00:00 UTC on 1 January 1970. And the column “Close” will be our target variable which we will try to predict.

We will focus on columns I-e “Close” and “timestamp” because neural prophet have specific requirement regarding dataframe that is data should have two columns one is date\_time and other is the target variable that is to e predicted , so firstly looking at timestamp column we see that is given in minutely wise I-e each row is 60 seconds apart but there are certain rows where difference between consecutive rows is greater than 60 I-e it is either 120 or 180 or more so we must deal with such missing data

A picture containing schematic

Description automatically generated

There are different approaches for it like we can divide the data based on largest gap but then we would be compromising on our data I-e we would be skipping a lot of data which won’t be based on 60 seconds interval but on largest interval like 5 minutes or more, the approach I will be adopting is to first I will be filling the gaps with 60 seconds and filling the null values row with previous values i-e pad



A screenshot of a computer screen

Description automatically generated with low confidence

then selecting my data based on hourly manner as dataset is quite large so I want to minimize it as I am using it will only for demonstration if I was using the data for prediction then I would have chosen 60 seconds gap approach as I would have retained as much data as possible

Graphical user interface, text, application

Description automatically generated with medium confidence

Then comes looking for trends which is increasing or decreasing of value in our data and seasonality which is repeating short-term cycle in our data by doing that we can get a sense how our data behave depending on situations like trends in data by seeing how data changes with time and yearly seasonality by seeing how data cycles on yearly basis, monthly seasonality by seeing how data cycles on monthly basis and so on. We can look for data behavior on holidays like Christmas season etc. by doing so we get to know if data is somewhat close to what we expect it to be.

To get understanding of trend and seasonality, I used **stats.model** seasonal decompose feature and plot it, which gives us trends and seasonal plot of data. If seasonality is constant through trend, then will be additive and if it grows with trend then it will be multiplicative. In our case it is clearly additive as doesn’t grow with trend.

Timeline

Description automatically generated

Neural Prophet is open-source library for time series forecasting developed by Facebook. NeuralProphet follows the sklearn model API. We create an instance of the NeuralProphet class and then call its fit and predict methods.

The input to Neural Prophet is always a dataframe with two columns: **‘ds’** and **‘y’**. The **‘ds’** (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH: MM: SS for a timestamp. The **‘y’** column must be numeric and represents the measurement we wish to forecast.

I split the data into train and test set with ratio of 9:1.

I made a function **get\_features** to take dataframe and change it accordingly to neural prophet desire dataframe.

I also made the function **mean\_absolute\_percentage\_error** (mape)to get model predictive behavior value based on mean absolute percentage error.

And then used the default parameters to train the model on train data.

Then I individually predicted the model on train data and test data. Prediction on train data gave **mape** value of 37.62% and prediction on test data gave an **mape** value of 43.1%.