Time Series Forecasting

The goal of this project was to use GitHub to track created and closed issues of given repositories for the past 2 years along with time series forecasting using Tensorflow/LSTM, FB Prophet, and StatModel, and then deploying it to the Google Cloud platform.

What is Time Series?

- Time Series is a collection of data points indexed based on the time they were collected. Most often, the data is recorded at regular time intervals.

- What makes Time Series data special? Forecasting future Time Series values is a quite common problem in practice.

Predicting the weather for the next week, the price of Bitcoins tomorrow, the number of your sales during Christmas, and future heart failure are common examples.

FB Prophet:

Facebook developed Prophet FB as an algorithm for the in-house prediction of time series values for different business applications. Therefore, it is specially designed for the prediction of business time series.

TensorFlow/LSTM:

LSTM stands for Long short-term memory.

LSTM cells are used in recurrent neural networks that learn to predict the future from sequences of variable lengths. Note that recurrent neural networks work with any kind of sequential data and, unlike ARIMA and Prophet, are not restricted to time series. The main idea behind LSTM cells is to learn the important parts of the sequence seen so far and forget the less important ones. This is achieved by the so-called gates, i.e., functions that have different learning objectives such as a compact representation of the time series seen so far how to combine new input with the past representation of the series what to forget about the series what to output as a prediction for the next time step.

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

Statmodels:

StatsModel is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests and statistical data exploration. An extensive list of result statistics is available for each estimator. The results are tested against existing statistical packages to ensure that they are correct.

statsmodels.tsa contains model classes and functions that are useful for time series analysis.

Walkthrough of the project:

In order to fulfill the requirements, I am implementing time series forecasting to retrieve data from the GitHub repositories for the previous two years. Next, I am using Docker and GCloud to deploy microservices, using Python and Flask for the backend, and finally using React and Javascript for the front end, thus forming the 3 necessary microservices.

React

React retrieves GitHub-generated and closed problems for a certain repository and uses high charts to display the corresponding bar charts; in this case, I also created components for line charts and stack bars.

It then also displays the images of the forecasted data for the given GitHub repository and images are retrieved from GCP storage

Lastly, React is making a fetch API call to the Flask microservice.

Flask

Following that, Flask takes the repository name from the API's body (i.e., from React) and retrieves the opened and closed problems for the specified repository during the previous two years.

Additionally, it is also fetching the author_name and all other information for the created and closed issues.

It is then using the data obtained from GitHub and passing it as an input request in the POST body to the LSTM microservice which is then predicting and forecasting the data as mentioned below under LSTM-forecast.

The response obtained from the LSTM microservice is also returned back to the client.

LSTM-forecast

It is the implementation to accept the GitHub data from the Flask microservice and will forecast the data for 30 days based on the past 2 years.

It is also plotting the required graphs using Matplot lib, and this graph will be stored as images in Google Cloud storage.

The image URL is then returned back to the flask microservice.

Facebook/Prophet:

The prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend and typically handles outliers well.

Libraries to be imported:
import pandas as pd
from prophet import Prophet

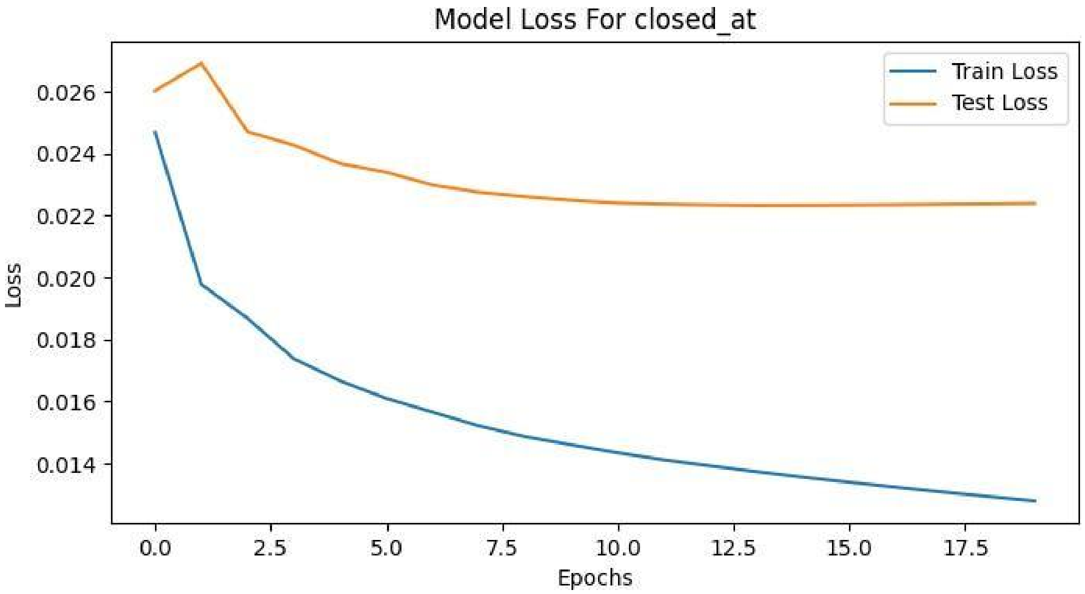
After getting the data from GitHub, we use pandas to read the file. Then we are formatting the data in two columns: “ds” and “y”. Then we fit the model by instantiating a new Prophet object. Any settings to the forecasting procedure are passed into the constructor. Then you call its fit method and pass in the historical data frame. Predictions are then made on a data frame with a column ds containing the dates for which a prediction is to be made.

StatModel:

Imported the necessary libraries, such as pandas and the specific modules from statsmodels.tsa (time series analysis). After retrieving the GitHub data, we did formatting it into the required input format, typically with columns for dates ("ds") and corresponding values ("y"). Then, by instantiating a suitable StatModel object and passing the historical data, we could fit the model and make predictions for future time points.

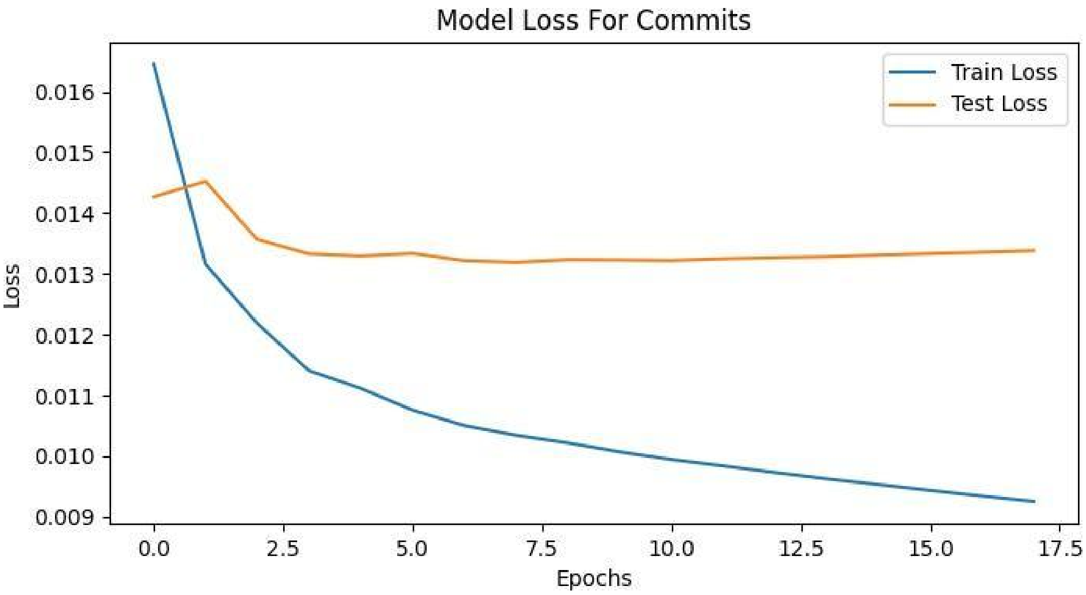
Comparative Analysis:

Model Loss:



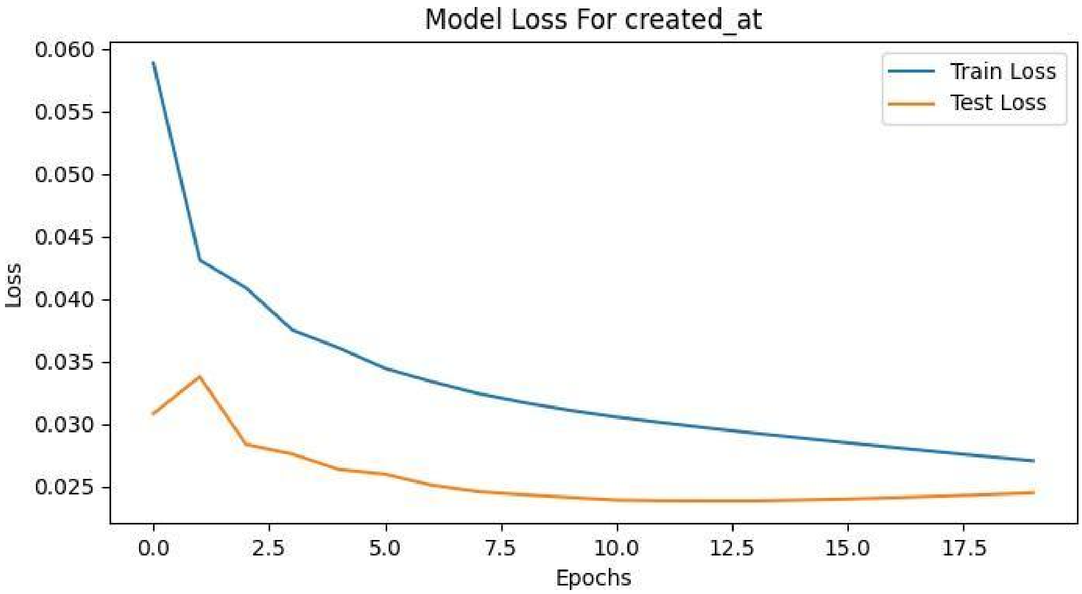
1.a Model_loss_closed_at_angular.png

Here (Image 1. a) I am doing a comparison between the experimental results obtained for model loss, Train loss, and Test loss, as we are observing initially Epochs loss is high initially for both but afterward, it reduces for Train data, Loss data it reached a peak initially, then drops but tends to be high throughout. Overall the situation is Overfit.



1.b Model_loss_commits_angular.png

Similarly, the Model Loss for commits (1. b), is an overfit situation as the difference between Train-Test Loss is significant and Train loss remains largely low,

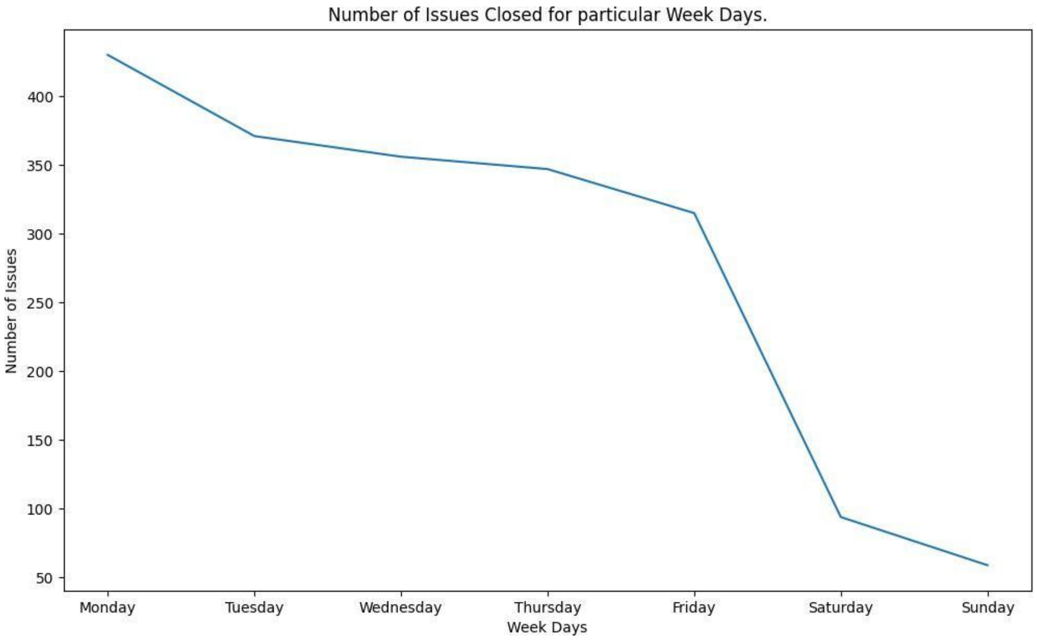


1.c Model_loss_created_at_angular.png

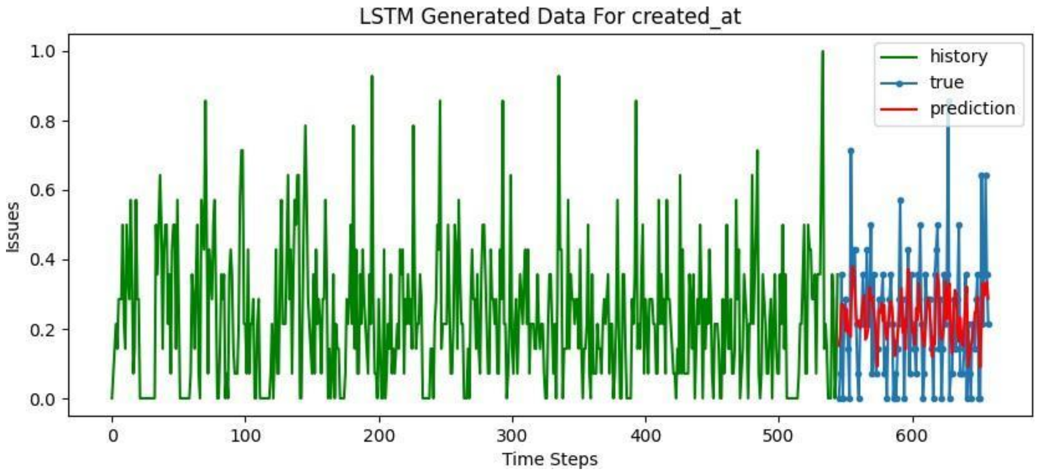
Model loss for Created(1.c) at data is the best fit, as epoch losses are similar after the initial drop for both Train and Test data

Time series forecasting using Tensorflow/LSTM, FB Prophet, and StatModel

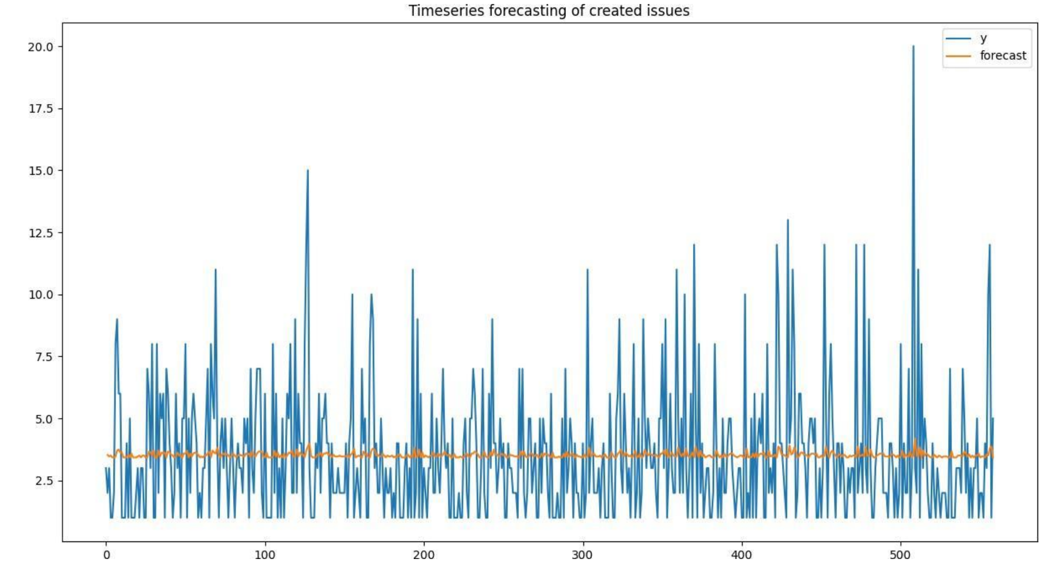
1. The day of the week the maximum number of issues created



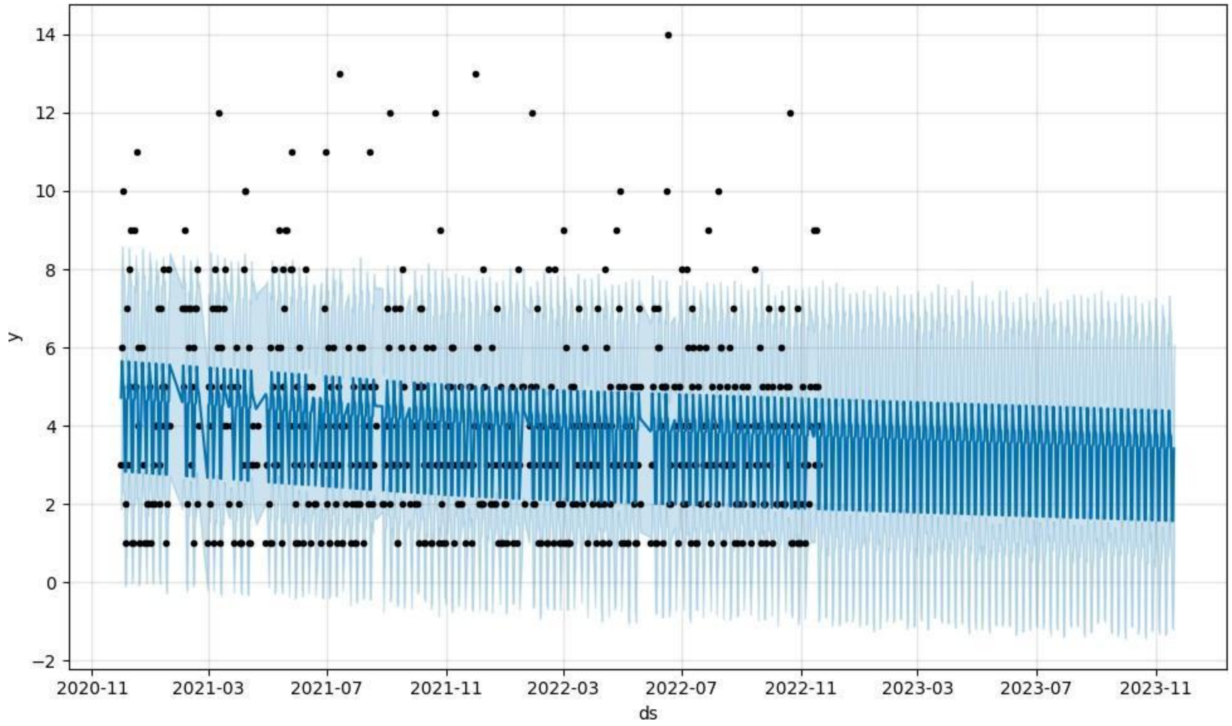
I am using to_period ('d') to set the frequency for extracting the number of issues, and will be able to infer from the graphs for the day of the week the maximum number of the issue are created, and similarly for each repository, So for LSTM below are the graph plotted, and the red plot showcases the time forecasting through LSTM and observing whether the forecasting is the best fit



Secondly taking the graph for StatsModel, for the created issues, the forecast is consistent but does not consider the distortion.



Thirdly FB Prophet, it has the best forecasting out of all as we are observing below, it has got the most accuracy and best bin width out of the three.



So overall per the experimental results, we can conclude and recommend that FB Prophet has got the best time series forecasting.