Declaration

I hereby affirm that I am the sole author of the report I have submitted. Information used in the study must be identified as a source, whether direct or indirect. I am aware that the report's digital form will be checked for evidence of improper use and to establish whether the entire report or any portions of it might be considered plagiarized. After comparing it to the available sources, I accept that my work will be added to the database. Where it will likewise stay following the review to enable comparison with the potential report submitted, any future rights to replication or usage shall be prohibited. This report has never been submitted to a board of examination and has never been published.

Abstract

The hunt for methods to forecast a stock's future value employing deep learning models and Fourier analysis motivated this work. Machine learning has recently emerged as a technique for studying financial text data. One of the most popular and valuable parts of finance is stock market forecasting. Time series analysis is the most well-known and vital procedure for reaching this goal. The objective of this study is to acquire historical data on stock and employ Fourier function regression combined with the use of Deep Neural Network models to predict future values of the target variable using that model. Since a wide variety of Fourier functions exist that can be used in the analysis, we tested our work for different Fourier functions (only sine terms, only cosine terms, or both sine and cosine terms) and of various degrees. An exhaustive analysis of these Fourier functions for predicting Fourier parameters showed that employing Fourier functions with only sine or cosine terms gave accurate predictions even for varying degrees. On the other hand, a similar behaviour was not observed with Fourier functions with both sine and cosine terms, which gave reasonably accurate predictions for low degrees only. Finally, a Flask API instance was used to deploy the model to allow real-time calculation of Fourier parameters. The results of this deep learning model show that it can make reasonable predictions about future pricing based on actual data.

Acknowledgement

Contents

Introduction

The term "stock market" describes open marketplaces for issuing, acquiring, and exchanging stocks that trade over the counter or on a stock exchange. Fractional ownership in a firm is represented by stocks, also known as equities, and the stock market is a marketplace that investors use to purchase and sell ownership of such investible assets. Being able to access public money swiftly allows businesses to grow. Hence a well-run stock market is thought to be essential to economic growth.

The market is unstable and frequently unpredictable, as any one of us could have anticipated. Researchers have been experimenting with time-series data to forecast future values for years, with stock valuation forecasting being the most challenging and lucrative use. Market movement, however, depends on a variety of factors, only a small subset of which can be measured, including historical stock data, trade volume, and current pricing. The intrinsic value, assets, quarterly performance, recent investments, and strategy of a firm, for example, all impact the traders' confidence in the company, which in turn affects the price of its shares.

In recent years, Machine Learning has been very successful in achieving human-like performance levels and reliability in various fields of human activity. In many tasks, like Vision and Natural Language Processing, which were thought to be beyond human automation, Machine Learning techniques have now achieved near-human levels of accuracy, in some cases performing even better than humans **[1].** The possible domain of applications where Machine Learning can make significant contributions is increasing daily. Naturally, various Machine Learning models, from statistical techniques to more sophisticated Deep Neural Network architectures are now being successfully applied to the domains of market analysis, which will be the central theme of our work.

This paper uses the mathematical technique of Fourier analysis to describe the development and deployment of a Stock price prediction model. It uses Fourier Function Regression to build the model with the past 22 years of stock data of the S&P 500 index (Standard and Poor’s 500) for this stock price prediction. The main target of this project is to predict the average stock price for the next ten days.

We will first apply pre-processing and feature extraction to the raw S&P 500 stock price data. We will then use Fourier function regression on a sliding window of 10 days in the

past to obtain a set of fitted Fourier parameters. After normalizing with Robust Scaler, the data will be appropriately split into train and test sets. Later, we will construct a deep neural network model for training and testing. The prediction will follow the model fitting with specific epochs and batch size. We use the Mean Absolute Error (MAE) and r2 score metrics to evaluate the correctness of the model's predictions. An API for obtaining the Fourier function parameters, given the type and degree of the Fourier function, will be subsequently deployed using Flask.

Theoretical Background

STOCK MARKET

A stock is a sort of investment that represents a proportionate ownership interest in the issuing company **[2].** Stocks are sometimes referred to as shares or equity. Every day, shares and other financial instruments are bought and sold on the stock market, which is a public market. A share of a company's ownership is represented by each one, and the S&P 500 index comprises the shares of the 500 largest American firms.
Datasets that are used to forecast stock prices typically include the Open, Close, High, Low, and Volume. The OPEN price is the initial price of any listed stock at the start of a trading day on an exchange. The stock's best and lowest values on The HIGH and LOW prices on that day are well-known. Traders frequently employ these stats. To calculate the volatility of the stock. The CLOSE price of a stock is the price at which it trades at the end of the trading day. Volume is the total number of securities, contracts, or stocks exchanged over all markets in a specific time frame. Changed CLOSE prices are taken as representing the stock's true value and show the stock's value following dividend payments.

EPS, or earnings per share, is a commonly used metric for assessing a company's profitability. The Price to Earnings Ratio (P/E) measures how much a company's shares are currently worth in relation to its earnings per share (EPS) **[3].**

The stock market is a novel issue for researchers, academics, traders, investors, and businesses due to its extreme volatility. It is now possible to predict the market to some extent thanks to improvements in processing capacity and methodology.

S&P 500 index

The joint venture S&P Dow Jones Indices' registered trademark is the S&P 500, often known as the Standard & Poor's 500. The 500 largest U.S. corporations make up this stock index, which is generally regarded as the most accurate gauge of the state of the market for American equities as a whole. **[4]**

From a different perspective, the S&P 500 index serves as a statistical gauge of the performance of the 500 largest stocks in the United States. The S&P 500 serves as a common standard against which the performance of a portfolio can be measured in this situation.

Market capitalization is weighted in the S&P 500 index (share price times the number of shares outstanding). This implies that the degree of influence a firm has over the performance of the index depends on its valuation. Not just 1/500th of the index is represented by each listed company. The S&P 500 index is more impacted by large corporations like Apple and Amazon than by comparably smaller firms like Macy's and Harley-Davidson.

One crucial issue is that, despite the fact that there are 500 significant enterprises, the valuations differ widely. The market capitalizations of several of the index's biggest companies exceed $1 trillion. The lowest S&P 500 companies, with market capitalization between $6 billion and $7 billion, are more than 200 times smaller than this.

Throughout the trading day, the S&P 500 index's value swings continuously depending on performance-weighted market data for the underlying companies.

STOCK MARKET PREDICTION

Stock market prediction is an endeavour to forecast the future value of business stock or other financial instruments traded on an exchange. A stock's future price prediction that is accurate could result in a sizeable profit. One of the most challenging duties is to predict and analyze the stock market. This is due to a multitude of factors, such as market volatility and a variety of many additional dependent and independent factors that affect a particular item's market value stock. These variables make it very challenging for any stock market expert to accurately forecast the rise and fall of the market.

However, with the advent of Machine Learning and its powerful algorithms, current market research and stock market prediction developments have started to incorporate such methods in their analysis of stock market data **[5].** Many people have conducted stock price predicting studies due to the stock market's consistent growth over the past few decades. They made an effort to evaluate and project price and movement changes on the stock market.

The world economy, politics, governmental policies, natural or man-made calamities, investor behaviour, and other factors all have an impact on the price of a particular stock. It is one of the most difficult tasks in time series forecasting. Time series data can be reliably predicted using traditional statistical methods as well as more modern Deep Learning approaches like Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM).

MACHINE LEARNING

Software programs may anticipate outcomes more correctly with the use of machine learning (ML), a type of artificial intelligence (AI), without needing to be explicitly told to do so. Machine learning algorithms use previous data as input to anticipate new output values.

Recommendation engines commonly employ machine learning. Other frequent uses include face detection, fraud detection, spam filtering, malware threat detection and predictive maintenance.

Machine learning is important because it enables the creation of new products and gives organizations an understanding of consumer behaviour trends and operative business patterns. Machine learning is crucial to the operations of many of today's leading companies, like Facebook and Google. Machine learning has become a major point of competitive difference for many firms.

A typical method to categorize conventional machine learning is the process through which a prediction-making algorithm learns to increase its accuracy. The four main approaches are supervised learning, unsupervised learning, reinforcement learning, and semi-supervised learning. The sort of algorithm that data scientists employ depends on the type of data that they want to forecast.

- **Supervised Learning**: In supervised learning, data scientists describe the variables they want the algorithm to look for connections between and provide the algorithms with labeled training data. The algorithm's input and output are both described.
- **Unsupervised Learning**: Algorithms trained on unlabeled data are used in this sort of machine learning. The algorithm searches through data sets in search of any significant relationships. Both the input data that algorithms use to train and the predictions or suggestions they produce are predefined.
- **Semi-supervised Learning**: Semi-supervised learning is a method of machine learning that combines the two categories mentioned above. An algorithm may be fed mostly labeled training data by data scientists, but the algorithm is allowed to explore the data on its own and come to its own conclusions about the data set.
- **Reinforcement Learning**: Data scientists frequently use reinforcement learning to instruct a computer to carry out a multi-step procedure for which there are set rules. An algorithm is programmed by data scientists to fulfill a goal, and they provide it with positive or negative feedback as it determines how to do so. However, the algorithm typically chooses the course of action on its own.

Time Series Analysis

TIME SERIES

A time series is described as a collection of data points arranged chronologically. Daily, monthly, or even annual time order options are available. A time series, in particular, enables one to see the causes that affect certain variables across time. To examine how a certain asset, security, or economic indicator evolves over time, time series analysis can be helpful. Both fundamental analysis and technical analysis employ forecasting techniques based on time series.

Analyzing a Time Series

Time Series forecasting is the process of using a statistical model to predict future values of a time series based on past results **[6].**

- Predicting data with time series analysis involves estimation of future data using known previous data by a substantial model.
- Its aim is to determine and understand patterns in changing data, where these patterns represent components of time series such as a cycle, trend, or irregular movements.
- To produce accurate forecasts of future sales, GDP, and global temperatures, the data model must take into consideration these trends when these elements are present in a time series.
- Many other types of applications can make use of time series analysis, such as stock market analysis, sales forecasting, yield prediction, financial analysis, and forecasting.
- Autoregressive (AR), Integrated (I), Moving Average (MA), and some other combinations such as Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) are commonly time series models.
- We can also make use of mathematical techniques like Fourier analysis to study time series data. These, when combined with the power of Deep Neural Networks, can be used to make effective future predictions.

Machine Learning Models For Time-series Forecasting

ARIMA Model

A model called ARIMA, or autoregressive integrated moving average, is defined by three order parameters: (p, d, q).AR(p)

- **AR(ρ) Autoregression –** Automatic Regression The dependent connection between current and historical observations is used as the basis for a regression model known as autoregression. An auto-regressive (AR(p)) component is included when the regression equation for the time series includes past values.
- **I(*d*) Integration –** employs differencing of data to make the time series stable (by removing an observation from the previous time step observation). A series' current values are subtracted from its prior values d times in the process of difference.
- **MA(*q*) Moving Average –** a model that utilizes the connection between a delayed observation and a moving average residual error. A moving average component integrates prior error components and serves as the model's error representation. The order q indicates how many terms must be included in the model.

The ARIMA model performs better with smaller datasets.

SARIMA Model

In order to represent the time series, we use a SARIMA model. Seasonal AutoRegressive Integrated Moving Average is known as SARIMA. It is made up of the AR and MA models. Three variables define the model:

- d = associated with initial differencing.
- p = the AR part's hierarchy.
- q = order of the moving average part.

The ACF plot, which displays the autocorrelations that quantify the link between an observation and its preceding one, may be used to calculate the value of p. The number of transformations required to make the time series stationary may be used to derive the value of d, which is the order of integration. The PACF plot may be used to calculate the value of q **[7].**

The Dickey-Fuller test, which can detect whether a time series is stationary or not, may be used to calculate the value of d. We may make use of the statsmodels library's adfuller class. We create a method called test_stationarity() that, if the time series is positive, returns True; otherwise, it returns False.As many times as the time series becomes stationary, we alter it using the diff() method.

We may draw the ACF and PACF graphs in order to determine the values of p and q. The statsmodels library's plot acf() and plot pacf() methods can be used. The value of p corresponds to the highest value outside of the confidence interval on the ACF graph.

Vector Autoregressive Model (VAR)

A straightforward multivariate time series model, the vector autoregressive model links recent observations of one variable to earlier observations of that variable as well as to other variables in the system. Because they allow for feedback between the model's variables, VAR techniques differ from univariate autoregressive models. Due to the framework they offer for reaching important modelling goals, VAR models have long been a favourite in the fields of finance and econometrics. VAR models are primarily used for data description, forecasting, drawing structural conclusions, and evaluating policy **[8]**.

Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) is a modified version of the RNN architecture. It solves the problem of vanishing gradients effectively and enables neural networks to capture longer-range dependencies considerably in historical data **[9]**. LSTM performs better when working with a lot of data, which makes it very useful in state-of-the-art applications like speech recognition and synthesis.

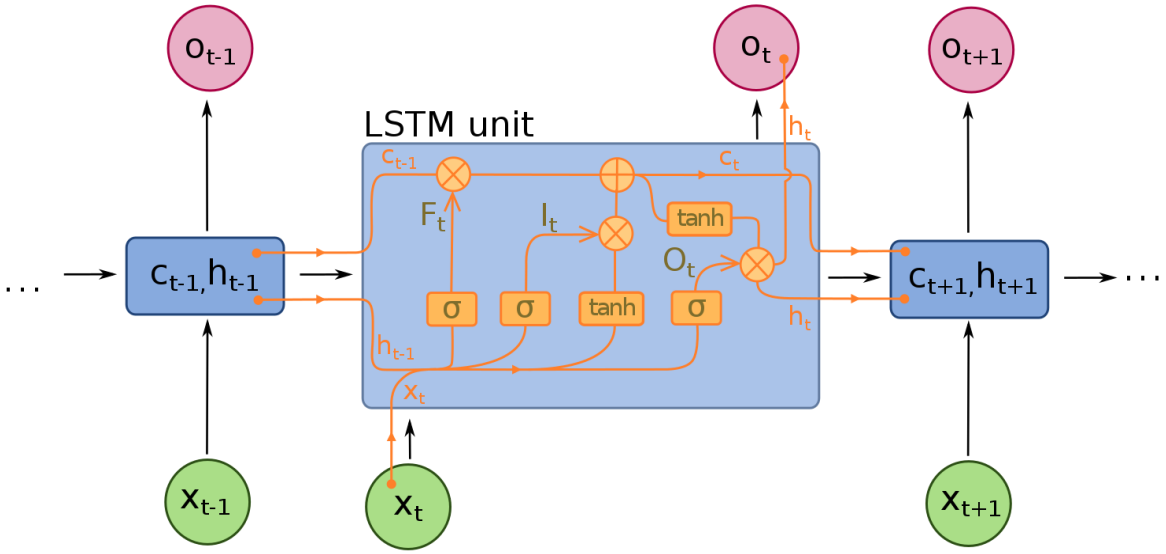


Fig 1 : Basic structure of a LSTM Model

Fourier Function Regression

A Fourier transform (FT) is a mathematical transformation that breaks down functions that rely on either time or space into functions that depend on either time or spatial frequency. The Fourier

transform takes a time signal and expresses it in terms of the frequencies of the waves that make up that signal **[10]**.

The Fourier transform is a central feature in many science and engineering domains. For example, it finds application in the solution of equations for the flow of heat, for the diffraction of electromagnetic radiation, and for the analysis of electrical circuits. The Fourier transform can be used to interpolate functions and smooth signals. For example, in the processing of pixelated images, the high spatial frequency edges of pixels can easily be removed with the aid of a two-dimensional Fourier transform **[11]**.

The Fourier transform can be extended to apply to the problem of time-series forecasting by assuming a Fourier function of a particular type and degree and fitting the assumed function on feature values obtained by taking sliding windows of a fixed number days in the past. This produces a set of Fourier function parameters for each such sliding window. These can be obtained for the model features and passed into the model (instead of the actual feature values) to predict the corresponding Fourier parameters for the target feature, which can finally be used to obtain the predicted values. We employ this approach in our work and show that it proves to be very effective in stock price analysis.

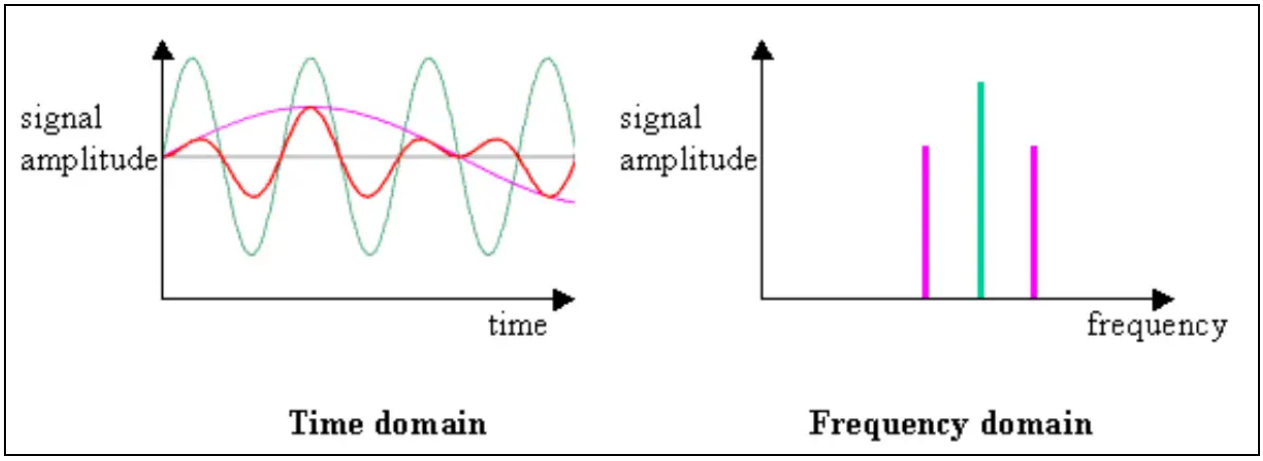


Fig 2 : Fourier series transform showing conversion of component frequencies in time domain to frequency domain [12]

Methodology

The main target of this project is to predict the Fourier parameters of the average price for the next 10 days of an S&P 500 index stock. The process of predicting these features step by step is shown below :-

3.1 Importing the required Libraries

Importing the necessary libraries is the first step in preprocessing S&P 500 index data. Additional libraries are needed for deploying and viewing the LSTM model output. The primary libraries that were essential to this deep learning model are:

- **Pandas:** For the purpose of manipulating and analyzing data, the Python programming language has a software package called pandas. It contains data structures and procedures specifically for working with time series and numerical tables. It is free software distributed under the BSD license's three clauses. The word is taken from "panel data," a phrase used in econometrics to refer to data sets that contain observations for the same persons throughout a range of time periods. Python data analysis is a play on words in the name of the thing.

- **MatPlotlib:** This ubiquitous data visualization toolkit makes the construction of two-dimensional diagrams and graphs easier. Because it offers an object-oriented API for

embedding plots into programs, the charting package Matplotlib is particularly helpful in data science projects.

- **SciPy:** Open-source software for mathematics, science, and engineering is called SciPy, which is pronounced "Sigh Pie." The NumPy library, which offers simple and quick N-dimensional array manipulation, is a prerequisite for the SciPy library. The SciPy library offers a variety of user-friendly and effective numerical routines, such as algorithms for numerical integration and optimization. It is designed to interact with NumPy arrays. Together, they are quick to install, free, and compatible with all common operating systems. Some of the top scientists and engineers in the world depend on NumPy and SciPy because they are simple to use but have sufficient power.

- **NumPy:** The cornerstone Python module for scientific computing is called NumPy. A multidimensional array object, various derived objects (like masked arrays and matrices), and a variety of routines for quick operations on arrays are provided by this Python library. These operations include discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more.

- **SciKit-Learn:** For projects based on Python-based data science, this is the industry standard. Scikit is a group of SciPy Stack packages created to handle particular tasks. Data scientists frequently use machine learning and data mining for clustering, regression, model selection, dimensionality reduction, and classification. The fact that Scikit runs well and has comprehensive documentation is another benefit.

- **TensorFlow:** A well-known Python framework for deep learning and machine learning is TensorFlow. It's the perfect tool for many tasks, including voice and object recognition. It aids in the creation of artificial neural networks that can handle numerous data sets. The library has a number of layer-helpers (tf-learn, tf-slim, skflow) that make it even more useful.

- **Keras:** A Python interface for artificial neural networks is provided by the open-source software package known as Keras. The TensorFlow library interface is provided by Keras. TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML were just a few of the backends that Keras supported up until version 2.3. One and only TensorFlow is supported as of version 2.4. Its user-friendliness, modularity, and extensibility are its main design goals as it aims to facilitate quick experimentation with deep neural networks.

In addition, we also use libraries like flask and flask_restful during API deployment.

3.2 Dataset Preparation

The raw stock data for any company can be obtained from Yahoo Finance as part of the data preparation. When constructed, all the necessary stock data features will be included in the dataset. The S&P 500 index's Open, High, Low, and Close Price are some of the stock features used in this dataset. These will be used to calculate the Average Price for the current period and the Average Price for the next 10 days. In addition, we also make use of the Volume of transactions as a feature for model training.

3.3 Data Preprocessing

We first make use of the Pandas library to load the S&P 500 dataset, which is stored in the .csv (comma separated value) format, to a Pandas data frame, as shown in the picture below. The input dataset includes S&P 500 index data for the period from January 3, 2000, to April 14, 2022. The input data has 5606 rows and 9 columns. The dataset will be displayed as :-

	id	uuid	dcp	opcp	hpcp	lpcp	cpcp	acpcp	vtcp
0	1	NaN	2000-01-03	1,469.25	1,478.00	1,438.36	1,455.22	1,455.22	931,800,000
1	2	NaN	2000-01-04	1,455.22	1,455.22	1,397.43	1,399.42	1,399.42	1,009,000,000
2	3	NaN	2000-01-05	1,399.42	1,413.27	1,377.68	1,402.11	1,402.11	1,085,500,000
3	4	NaN	2000-01-06	1,402.11	1,411.90	1,392.10	1,403.45	1,403.45	1,092,300,000
4	5	NaN	2000-01-07	1,403.45	1,441.47	1,400.73	1,441.47	1,441.47	1,225,200,000
5	6	NaN	2000-01-10	1,441.47	1,464.36	1,441.47	1,457.60	1,457.60	1,064,800,000
6	7	NaN	2000-01-11	1,457.60	1,458.66	1,434.42	1,438.56	1,438.56	1,014,000,000
7	8	NaN	2000-01-12	1,438.56	1,442.60	1,427.08	1,432.25	1,432.25	974,600,000
8	9	NaN	2000-01-13	1,432.25	1,454.20	1,432.25	1,449.68	1,449.68	1,030,400,000
9	10	NaN	2000-01-14	1,449.68	1,473.00	1,449.68	1,465.15	1,465.15	1,085,900,000

Table 1 : Format of input data

We first do basic pre-processing by removing commas from values in each column and converting them into floating-point values. Next, we calculate the average price for the current period by taking the average of Open, High, Low, and Close prices for that day. Next, we

calculate the average price for the next 10 days by taking the values of the average price for each of the next 10 days and computing their average.

3.3.1 Scaling the Data

The range of data features is normalized via feature scaling, sometimes referred to as data normalisation. A major issue with most ML models is that they are very sensitive to the scale of input data, in that they only work well when the input lies between a small range of values. Since the range of input values for ML algorithms can be relatively wide, feature scaling becomes a vital step while running them. The data is transformed into an array of values with zero mean and unit variance after normalization.

Data scaling aids in decreasing computation costs and reduces memory use. By scaling down, we can also increase precision because the data is not dispersed widely.

The basic equation for normalizing data is :-

$$x^{'} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where,

X_{min} = the minimum value of data,
X_{max} = the maximum value of data,
X' = the normalised value

This type of scaling can be implemented using Scikit-learn's StandardScaler class, which normalizes the data to zero mean and unit variance.

Because Deep Neural Networks are sensitive to outliers, we can use the Sci-Kit-learn library's RobustScaler class, a modified version of the StandardScaler class.

RobustScaler uses the interquartile range so that it is robust to outliers. Therefore its formula is as follows: **[13]**

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

where,

$Q_1(x)$ = the lower quartile (25% quartile),
$Q_3(x)$ = the upper quartile (75% quartile)

3.3.2 Creating and Splitting of Train and Test Data

We must divide the dataset into train and test sets before feeding it to the model. We will therefore perform the train-test split on data after feature scaling. A wide variety of methods, including Random Seed and Cross Validation, are employed for separating train and test data. However, the data in time series follows a definite order and cannot be randomly shuffled. Hence, we simply divided the dataset according to the needs of the problem.

We used various train-test split ratios to check the effectiveness of the model and finally made use of a 80% train and 20% test split on the input data for best results, which was later used for deployment.

3.3.3 Timesteps

Time steps must be created to provide short-term past data for time series analysis, which in turn allows any model to infer the data's long-term dependencies. Because projections for the following 10 days are taken into account in this project, for each feature used in the analysis, we took a sliding window of 10 days in the past. For each such sliding window, we obtain a set of 10 feature values, which will then be fitted with a Fourier function of a particular type and degree to obtain a set of Fourier function parameters.

3.3.4 Selecting Features and Setting Target Variable

An effective input feature to represent the daily variations in stock prices in terms of longer time scales is the average price for the current period, which can be computed by taking the average of the Open, High, Low and Close prices. Since we are also interested in monitoring the volume of stocks traded per day, the transaction volume of stocks per day is also an essential input feature. Lastly, in line with our motive to make a prediction regarding the future variation of stock prices, we define another feature called the average stock price for the next 10 days, which is computed by taking the average of the daily average stock prices for the next 10 days.

The central idea behind applying Fourier analysis to the given time-series data is to first take a date and for each of the above mentioned features, construct a sliding window of feature values for 10 days in the past. The next step would be to fit a Fourier function of a particular type and degree on the 10 data points so obtained, and obtain the corresponding Fourier function parameters. This procedure is repeated for each of the three aforementioned features to give 3 sets of Fourier function parameters, which will now be used to construct the input data for the model. **[14][15]**

The datasets X_train, Y_train, X_test and Y_test are then created for each of these 3 features by separating the last 1000 input samples to be the test data and using the rest for train data.

```
avg_price_train = normalized_params_3degree[:-1010]
avg_price_test = normalized_params_3degree[-1010:-10]
vol_train = normalized_params_3degree_vol[:-1010]
vol_test = normalized_params_3degree_vol[-1010:-10]
avg_price_next_train = normalized_params_3degree_next[:-1000]
avg_price_next_test = normalized_params_3degree_next[-1000:]
```

Fig 3 : Performing train-test split on the input data

They follow the shape (n_x, n_f), where :-

n_x :- no. of data samples,
n_f :- no. of Fourier function parameters

The Fourier parameters for each feature are then scaled to a normal distribution using the RobustScaler class of sklearn, which in addition to normalizing the values to zero mean and unit variance, also provides added robustness to outliers.

```
# normalize the calculated Fourier parameters to be passed as input
scaler = RobustScaler()
normalized_params_kdegree = scaler.fit_transform(params_kdegree)
normalized_params_kdegree_vol = scaler.fit_transform(params_kdegree_vol)
normalized_params_kdegree_next = scaler.fit_transform(params_kdegree_next)
```

Fig 4 : Scaling the input data

3.4.1 Creating the Model

We will now construct a model architecture, which takes in two input features :-

- Fourier function parameters for the average price current period,
- Fourier function parameters for the transaction volume of stocks

and predicts as output,

- Fourier function parameters for the average price for the next 10 days.

The model was created using different types of Fourier functions, which can be distinguished on the basis of two characteristics -

1. Type :-
    - only sine terms,
    - only cosine terms,
    - Both sine and cosine terms
2. Degree :-
    Can be any positive integer

```
# Creating Fourier Function
def make_func(ftype, numarg):
    def func(x, *a):
        n = int((numarg + 1)/2)
        s = 0
        for i in range(n):
            if i == 0:
                s = a[i]/2
            else:
                if ftype == 'cos':
                    s += a[i]*np.cos(i*x)
                elif ftype == 'sin':
                    s += a[i]*np.sin(i*x)
                elif ftype == 'cos_sin':
                    s += a[i]*np.cos(i*x) + a[i+n-1]*np.sin(i*x)
        return s
    return func
```

Fig 5 : Defining the Fourier function

The overall structure of the model is as shown below :-

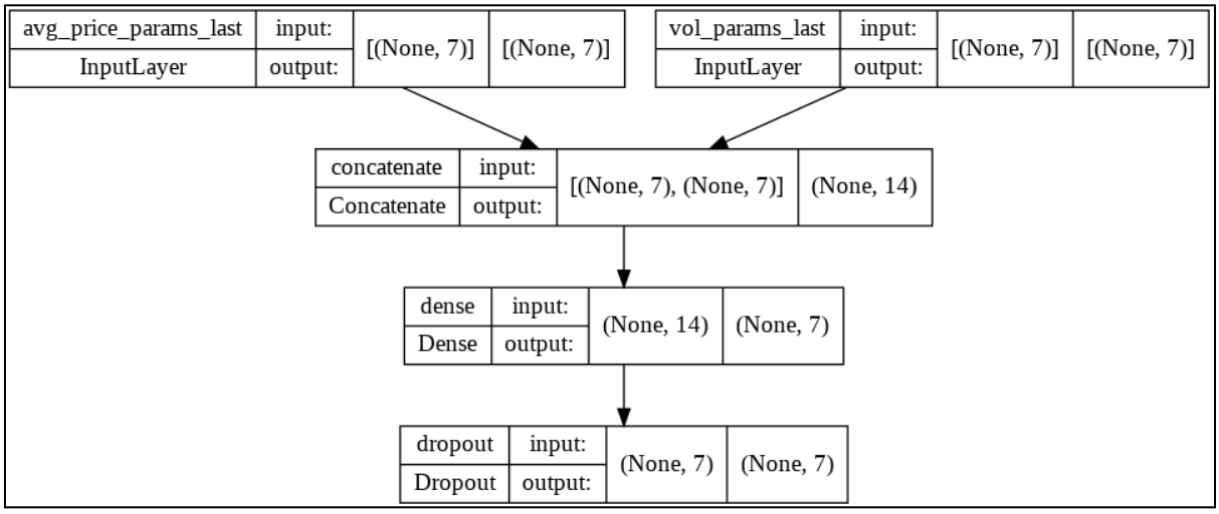


Fig 6 : Model diagram

```
avg_price_params_last = keras.Input(shape=(2*k+1,), name="avg_price_params_last")
vol_params_last = keras.Input(shape=(2*k+1,), name="vol_params_last")
inputs = layers.concatenate([avg_price_params_last, vol_params_last])
outputs = layers.Dense(2*k+1)(inputs)
model = keras.Model(inputs = [avg_price_params_last, vol_params_last], outputs = outputs)
```

Fig 7 : Model layers

Since the model takes in 2 sets of input features, we make use of the Keras Functional API to link multiple inputs together **[16][17]**. We will first concatenate the 2 input features into a single feature of length (2 * no. of Fourier parameters), and pass it as model input. A Dense layer with shape (None, no. of Fourier parameters) is used to construct the final model. We use Adam Optimizer to compile the model, with a learning rate = 3e-3, and employ Mean Absolute Error as the loss function. This was found to be the most ideal combination for the above DNN model.

Later, we use the fit function to train the model described below on the
training data for 30 epochs with a 128-batch size on the training data. These combination of values were found to give best results after extensive hyperparameter tuning.

```
model.fit(
    {"avg_price_params_last": avg_price_train, "vol_params_last": vol_train},
    avg_price_next_train,
    epochs=30,
    batch_size=128,
    validation_split=0.2,
    shuffle = False
)
```

Fig 8 : Model training parameters

3.4.2 Prediction

After training the model with train data, the next step is to obtain the prediction using this trained model for the Test data. After predicting test data, it should then be inverse-scaled to regain the normal values of the output feature. Y_hat denotes the predictions of the model.

The prediction model along with reshape and inverse transform are as shown in figure.

```
pred_test = model.predict({"avg_price_params_last": avg_price_test, "vol_params_last": vol_test})
pred_test_final = scaler.inverse_transform(pred_test)
```

Fig 9 : Model prediction

After predictions, the next step is to calculate the mean absolute error (MAE) and R2 score in order to calculate the efficiency of the model.

3.5 Results

Mean Absolute Error (MAE):

The Mean Absolute Error (MAE) is precisely the average of the absolute values of errors between predicted and actual values **[18]**. The sign of these discrepancies/errors is disregarded to prevent cancellations between positive and negative numbers. If the sign weren't ignored, the predicted MAE would probably be much smaller than the actual gap between the model and the data.

The MAE is computed as :-

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

R2 Score:

The coefficient of determination (R2) in a model of simple or multiple linear regression is the ratio of the explained variation (sum of squares attributable to regression) to the overall variance (total sum of squares total SS (TSS)). The ability of a linear regression model to predict the variation in a dependent variable is indicated by the coefficient of determination, often known as R-squared **[19]**.

The effectiveness of a linear regression model is assessed using the coefficient of determination, often known as the R2 score. What can be predicted from the input independent variable is the level of variation in the output dependent characteristic. Depending on the ratio of total deviation of outcomes given by the model, it is used to assess how effectively observed results are replicated by the model.

$$R^2 = 1 - \frac{\Sigma(y_i - \hat{y_i})^2}{\Sigma(y_i - \bar{y_i})^2}$$

It is equivalent to the square of the correlation coefficient.

Effect of degree and type of Fourier functions :-

The effect of using Fourier functions of various types and degrees is as shown below :-

Degree	MAE	r2 score	Layers	Training
3	29.033	0.9903	Dense(2k+1)	lr = 3e-3, batch_size = 128, no decay
4	372.044	0.5603	Dense(2k+1)	lr = 3e-3, batch_size = 128, no decay

Table 2 : Error values for both sines and cosines

Degree	MAE	r2 score	Layers	Training
3	14.04158	0.9994	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
4	18.3037	0.9963	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
5	22.9351	0.9818	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
6	584.698	-10.016	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
7	6357.4199	-4021.9589	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay

Table 3 : Error values for only cosines

Degree	MAE	r2 score	Layers	Training
3	8.57409	0.9998	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
4	14.4456	0.9986	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
5	15.6685	0.9965	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
6	35.8811	0.8788	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay
7	184.9279	-0.412	Dense(k+1)	lr = 3e-3, batch_size = 128, no decay

Table 4 : Table values for only sines

The common range of R-squared values is between 0 to 1. It needs to be nearly equal to 1 to have effective outcomes. The resultant R-squared score for this model was found to be close to 0.99 (considering different types and degrees of Fourier functions). Therefore, based on this score, it is suggested that this model may be used for deployment and additional training. Consequently, using all of these values and charts. It is claimed that this model is effective. Therefore, this model can be used for deployment purposes.

Plot:

Fig 10 : Plot of actual data (average price for current period vs date)

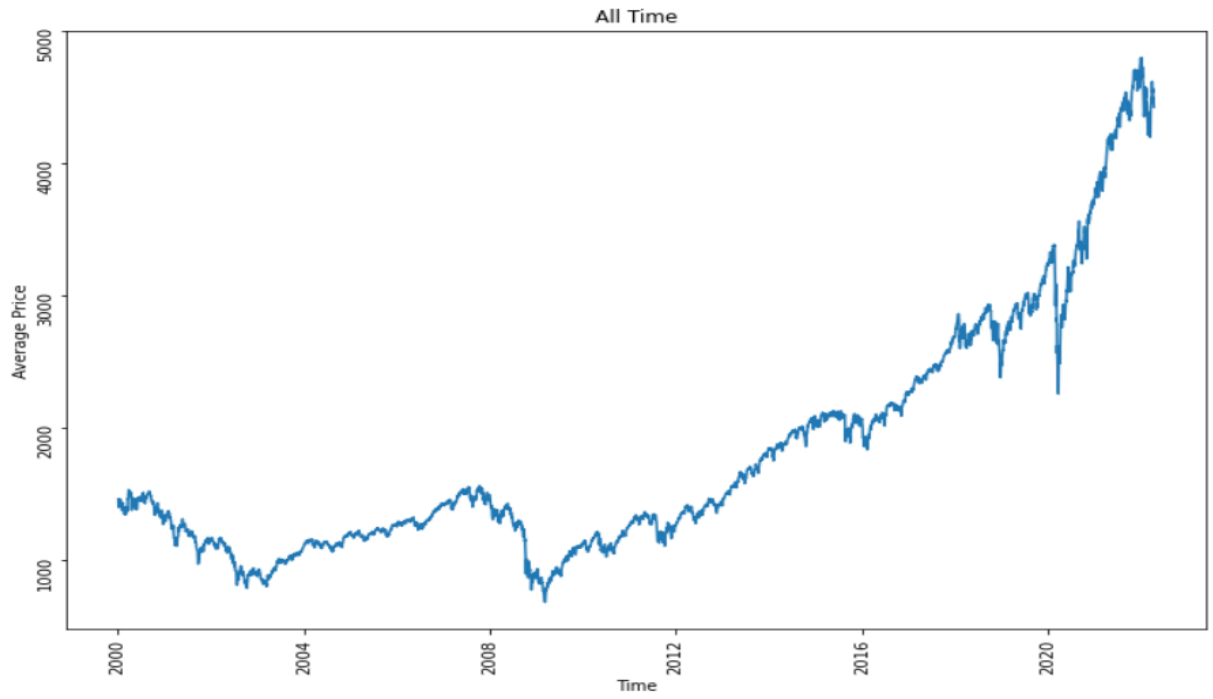
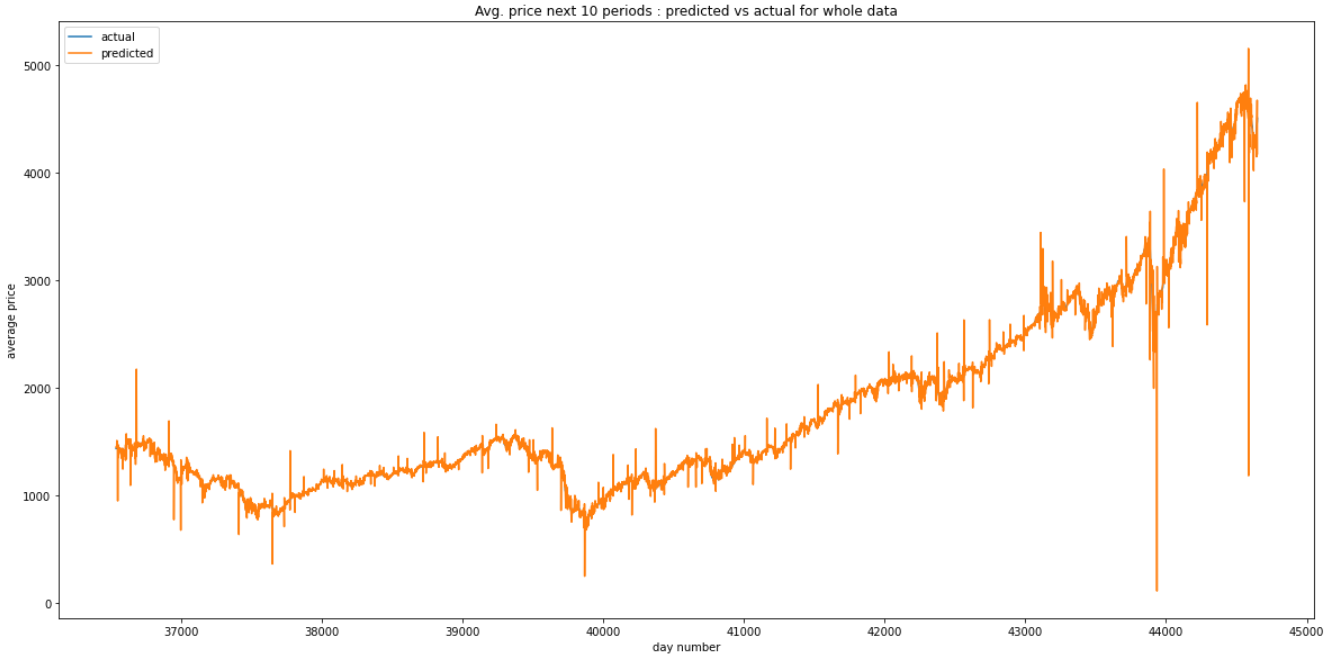


Fig 11 : Plot of actual vs predicted values for average price current period (type = only sines, degree = 3)



The S&P 500's Actual vs. Predicted Value Plot is depicted here. The almost fully overlapping actual and projected lines show how effective the model is.

4. Creating the API

In order to create an interface to obtain predicted values of Fourier parameters for the target variable, we made use of the Flask REST API to create an API Method Connector. Given the type and degree of the Fourier function, one can make use of API GET calls to obtain the predicted Fourier parameters.

The dependencies for the code can be installed by running the command :-

"pip3 install -r requirements.txt"

in the code folder. This will install locally all necessary libraries needed to run the program.

```
numpy
matplotlib
pandas
datetime
scipy
sklearn
tensorflow
keras
flask
flask_restful
```

Fig 12 : requirements.txt

Now run the code file “spp_s&p500_2000_2021_apn10pvcpcpr_fourier.py” to start the Flask API locally on your device.

```
 * Serving Flask app 'spp_s&p500_2000_2021_apn10pvcpcpr_fourier' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
 * Restarting with stat
```

Fig 13 : Running Flask API

For testing if the API runs correctly, we make use of the Postman app, which can be installed as per individual device requirements **[20]**.

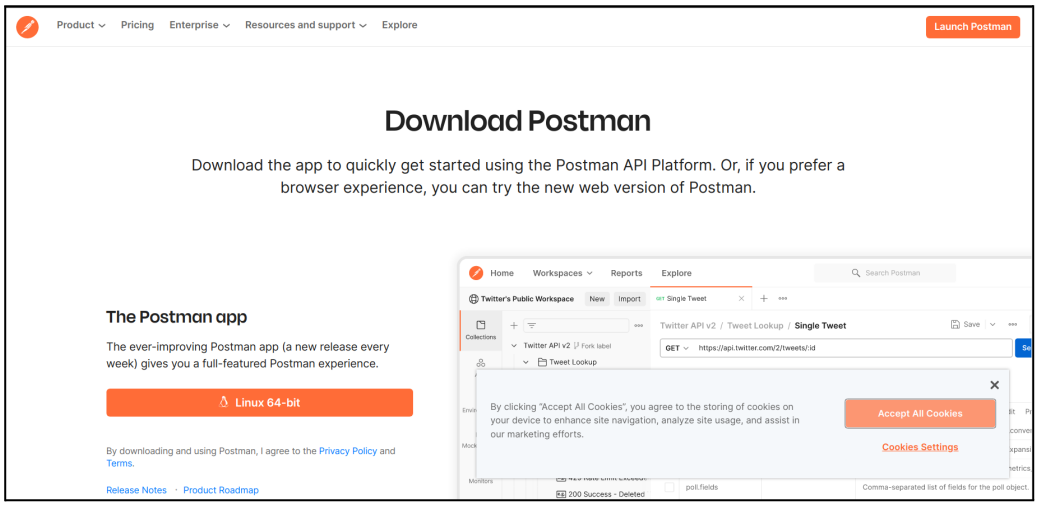


Fig 14 : Downloading Postman

Add the API url specified in the terminal with the address "/fourier" and pass two arguments "ftype" and "deg" to the API as follows :-

1. Type :-
    - only sine terms ("sin") ,
    - only cosine terms ("cos"),
    - Both sine and cosine terms ("cos_sin")
2. Degree :-

    Can be any positive integer.

Clicking on "Send" passes a GET API request to the API Method connector, which gives the set of predicted Fourier function parameters as output. A typical use of the API with Fourier function type "sin" and degree "3" gives :-

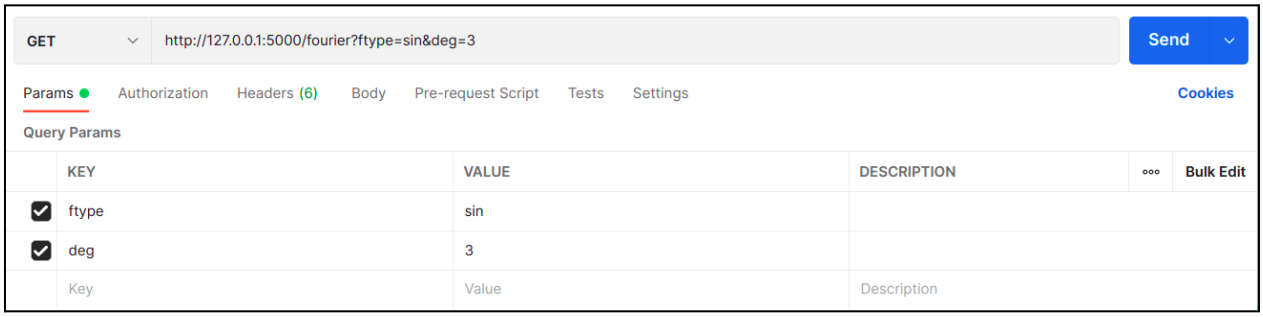


Fig 15 : API input

The output for this API request is saved in the "response.json" file.

Directory Structure :-

- "*spp_s&p500_2000_2021_apn10pvcpcpr_fourier.py*" – code file
- "*raw_features_data.csv*" – input data file
- "*requirements.txt*" – file containing necessary requirements
- "Degree vs Error.xlsx" - contains error values for different scenarios
- "*response.json*" –

    sample file containing API response for
    - ftype = "sin"
    - deg = 3

5. Conclusion

We employ the method of Fourier function regression to predict variations in stock market prices and show that it proves to be very reliable in making accurate predictions. The efficacy of the model demonstrates the strong dependence of the variation of the future stock market prices on the current trends in stock prices as well as the daily transaction volume of stocks. It can predict the average stock price for the next 10 days if it has all of the attribute information for that day's forecast. This also proves to be very helpful to traders who have already completed a next-day trade by the time the stock market closes for the day. The output values, which are the Fourier parameters for the average price for the next 10 days, can be used to obtain a reliable estimate of the future trends in stock prices, which can help to provide a quick overview of the stock's ups and downs over the ensuing 10 days. With the highest value for the upcoming 10 days, an investor can plan to sell to make money, and with the lowest value for the upcoming 10 days, investors and traders can plan to buy to acquire stocks at a lower cost. Finally, based on these anticipated outcomes, traders and investors can choose to purchase or sell a stock depending on the anticipated price for the following 10 days.

In the present work, at the deployment stage, the input which is a dataset should be set ready before using it to get the predictions. Hence, the updated dataset can be added to the code directory to run an updated set of predictions, in line with more recent trends. This model, which was created using a combination of Fourier function regression with deep neural network models, provides accurate and reliable estimate of the stock price variations. I suggest experimenting with more sophisticated deep neural network model architectures may help improve the precision and accuracy of the analysis.

Bibliography

1) Applications of ML :- https://www.infoq.com/articles/state-art-automl/
2) Finance blog :- https://reangfinance.blogspot.com/
3) What is PE ratio :- https://www.businessinsider.com/personal-finance/what-is-pe-ratio
4) S&P 500 index :- https://in.investing.com/indices/us-spx-500
5) Stock Market prediction using ML :- https://www.upgrad.com/blog/stock-market-prediction-using-machine-learning/
6) Time series forecasting :- https://www.alibabacloud.com/topic-center/tech/19tggrvkima7-time-series-forecasting-alibaba-cloud
7) How to use the SARIMA model :- https://towardsdatascience.com/how-to-model-a-time-series-through-a-sarima-model-e7587f85929c
8) Intro to VAR models :- https://www.aptech.com/blog/introduction-to-the-fundamentals-of-vector-autoregressive-models
9) Understanding LSTMs :- https://colah.github.io/posts/2015-08-Understanding-LSTMs
10) Fourier transform :- https://en.wikipedia.org/wiki/Fourier_transform
11) Fourier transform applications :- https://www.sciencedirect.com/topics/chemistry/fourier-transform
12) Fourier transform image :- https://www.motioncontroltips.com/how-are-fast-fourier-transforms-used-in-vibration-analysis/
13) RobustScaler :- https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml
14) Adding Fourier terms to regression :- https://towardsdatascience.com/how-to-add-fourier-terms-to-your-regression-seasonality-analysis-using-python-scipy-99a94d3ae51
15) Analyzing seasonality with Fourier Transforms :- https://towardsdatascience.com/analyzing-seasonality-with-fourier-transforms-using-python-scipy-bb46945a23d3
16) The Keras Functional API :- https://keras.io/guides/functional_api
17) Predicting mixed targets with neural networks :- https://towardsdatascience.com/predicting-mixed-targets-with-neural-networks-and-keras-1dc754ce0c98
18) Mean Absolute Error :- https://link.springer.com/10.1007%2F978-0-387-30164-8_525
19) R2 score :- https://www.investopedia.com/terms/r/r-squared.asp
20) Postman API :- https://www.postman.com/