Predicting Sales Using Time Series Forecasting

1st Akansha Jajodia
Computer Engineering Department
San Jose State University
San Jose, United States
akansha.jajodia@sjsu.edu

2nd Raaga Pranitha Kolla
Computer Engineering Department
San Jose State University
San Jose, United States
raagapranitha.kolla@sjsu.edu

3rd Saranya Visvanathan
Computer Engineering Department
San Jose State University
San Jose, United States
saranya.visvanathan@sjsu.edu

***Abstract*— Forecasting is the process of prediction about the future events which helps to act proactively. It has become the need of every industry from e-commerce, retail, storage, fashion and so on, to understand the business needs and meet the demand. In this paper, the process undertaken to build models capable of predicting future sales of given items is explained. The historical information of walmart sales is used to build machine learning models using time series algorithms to predict the next 28 days of sales. Later, the trained models are evaluated and its prediction performance is analysed. The time series algorithms used are ARIMA, LightGBM, FBProphet.**

Keywords: prediction, ARIMA, LGBM, FBProphet, SARIMAX, time series, sales forecasting

I. INTRODUCTION

Currently, the problems based on the time has been the common one. Time series forecasting provides better performance in many applications like sales prediction, weather prediction, health prediction, etc. It utilizes the past information and helps to provide better predictions on future horizons. Also, understanding and implementing the time series forecasting is one of the expected skills for any machine learning engineer. With that motivation, we decided to focus on sales prediction which would happen in the period of 28 days at an e-commerce store (Walmart) based on the historical information of the past sales data. We intend to use time series forecasting to predict sales.

Once the time series data is chosen, it is explored to understand its components. Later, the data is analysed for any missing values or improper format, then it is processed accordingly to be in the format required by the algorithms used. If needed, new columns are created using the existing information. Then, the data is split into two parts: training data and validation data. The training data is used to train the machine learning models and the trained model is evaluated on the validation data to analyse its performance. Finally, the trained models are used to make predictions of the future sales (28 days).

The structure of the paper is as follows. The section II background talks about the research which has already been done in this field to gain knowledge on how to approach the problem. The section IV explains the technical approach in detail to solve the problem, which is followed by the section IV, which specifies the tools used to build the project. Finally, the section VI summarizes the project.

II. BACKGROUND

Sales prediction becomes important in almost all industries such as retail, e-commerce, food, etc. This helps to make an informed decision on the businesses which could generate profit. Time series data holds historical information, which needs to be processed without disrupting the ordering in the data. This paper [1] explains the methodologies involved in building machine learning models for time series forecasting. Also, it helps to understand the concepts of feature engineering and model evaluation while using time series data.

This paper [2] explains how ARIMA removes stationarity in the data by adding a difference [2] step to satisfy its expectation of stationary data. It also explains the two variations of time series forecasting such as short lived and long lived. **Short lived forecasting** involves making predictions for a smaller time ranging from the next hour upto next two years. On the other hand, **Long lived forecasting** makes predictions over longer time ranging from the next 5 years or a decade. This paper [3] explains the usage of random forests, moving average for short lived forecasting. And, the preference of using advance networks for long lived forecasting is explained in this paper [4].

The paper [5] describes a hybrid combination of using ARIMA with neural networks to forecast time series data. The non-linearity in the data is handled by Artificial Neural Networks better than the standard machine learning models. So, this paper [5] makes analysis on the hybrid approach of using self organizing fuzzy neural network along with ARIMA on different datasets, where it makes predictions only using ARIMA model, then only with neural networks and later the hybrid approach combining both the models. The experiments resulted in the hybrid approach outperformed the rest of the approaches.

This paper discusses the usage of Facebook Prophet model for sales prediction. The model they built was tested against real world data, and they found out that it gave a good result in generating monthly and quarterly forecasts. The authors in paper [6] performs forecasting on Bitcoin dataset which contains data of almost 24 months. The model selection was done using threefold splitting techniques. ARIMA and Prophet results were compared and they found out that prophet works better than ARIMA with their dataset.

Since we worked on the M5 competition data, the results of the competition are used while choosing the algorithm for

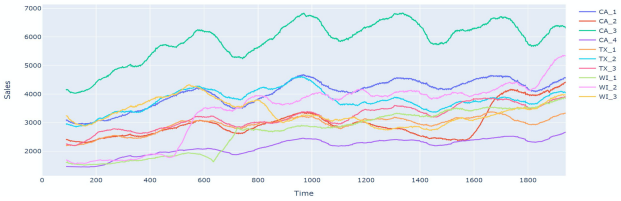

Fig. 1. Increasing trend in the sales among all stores in the dataset

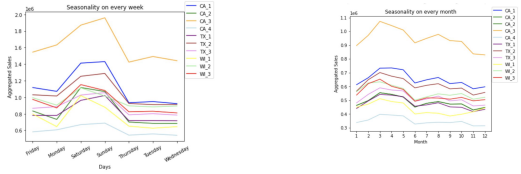

Fig. 2. Showing seasonality among different stores on every of week and every month in a year

time series forecasting. The discussion of M5 competition results on the YouTube channel "International Institute of Forecasters" explains the key findings, and it shows the top four winning models are based on LightGBM. This paper [7] helped to understand the usage of LightGBM for model training.

All these understandings are collectively used to implement the pipeline of the time series forecasting model with learning algorithms such as ARIMA, LightBGM, FBprophet.

III. DATASET DESCRIPTION

We used the dataset collected from M5 competition [8]. It has sales data for 1941 days. And, it is discrete and continuous data. This data details about sales of multiple items aggregating them into different departments, categories, stores and geographical areas. The data belongs to the Walmart store for 3 different states: California, Texas and Wisconsin. The three csv files are used namely: calendar.csv, sell_prices.csv, sales_train_evaluation.csv.

- **calendar.csv:** The main feature in this file specifies the days where there is a special event example Thanksgiving, Christmas, etc
- **sell_prices.csv:** This dataset gives the details of sales regarding to each store and category.
- **sales_train_evaluation.csv:** This is the main dataset which contains information of all the sales from day_1 to day_1941 (including the last 28 days sales which needs to be predicted)

IV. TECHNICAL APPROACH

A. Exploratory data analysis

Exploratory data analysis for time series datasets is different compared to other datasets. There might be correlation between past data to the current one that we are going to predict. The main components for time series data analysis are trend, seasonality and stationarity.

- **Trend:** This specifies whether the data has an increasing or decreasing trend in the amount of sales in the data. By doing EDA we found that there is a trend seen in the dataset among different stores which can be seen in 1.
- **Seasonality:** Seasonality specifies there is a pattern that repeats over a period of time. By doing EDA we found that the dataset has a seasonality which repeats every week and a seasonality which appears in some months in the year. 2 specifies the seasonality among different stores in a week and over a year.

Index	CA_1	CA_2	CA_3	CA_4	TX_1	TX_2	TX_3	WI_1	WI_2	WI_3	d
d_1	4337	3494	4739	1925	2595	3892	3030	2704	2258	4038	d_1
d_2	4155	3046	4827	1777	2687	3697	3006	2194	1922	4198	d_2
d_3	2816	2121	3785	1386	1822	2731	2225	1562	2018	3317	d_3
d_4	3051	2324	4232	1440	2258	2954	2169	1351	2022	3211	d_4
d_5	2630	1942	3817	1536	1694	2492	1726	2	1175	2132	d_5
d_6	3276	2288	4369	1589	2734	3439	2833	2049	2244	4590	d_6
d_7	3450	2620	4703	1469	1691	2588	1947	2815	2232	4486	d_7
d_8	5437	3729	5456	1988	2820	3772	2848	3248	2643	5991	d_8
d_9	4340	2957	5581	1818	2887	3857	2832	1674	2140	4850	d_9
d_10	3157	2218	4912	1535	2174	2932	2213	1355	1836	3240	d_10
d_11	2995	2123	4447	1368	1607	2628	1989	1335	1526	3053	d_11
d_12	2710	1901	4544	1195	2149	2699	1922	1410	1846	3312	d_12
d_13	2928	2436	4406	1434	1895	2515	1699	1349	1785	3309	d_13
d_14	3078	2584	4380	1312	2256	2883	2063	1946	2015	3928	d_14
d_15	4316	3455	5187	1652	2944	4000	3137	2834	2442	5066	d_15
d_16	4364	3563	5780	1933	3432	4779	3514	2012	2336	4677	d_16
d_17	2757	1801	3303	1163	1821	2679	2119	1522	1612	2907	d_17
d_18	2430	1797	3758	1511	2217	2927	2316	1363	1980	3771	d_18
d_19	2272	1741	3641	1190	1962	2606	2038	1300	1628	3065	d_19
d_20	2009	1614	3099	1113	1821	2449	2142	1459	1678	2934	d_20
d_21	3015	2214	3320	1476	1951	2615	2145	1872	1813	3300	d_21
d_22	4553	3080	4694	1543	2571	3705	2874	2596	2014	4059	d_22
d_23	3966	3087	4794	1618	2810	3845	3134	1410	1992	3022	d_23

Fig. 3. Sample Dataframe after aggregation

- **Stationarity:**Stationarity specifies if the data has a constant mean and standard deviation at any point of time in the dataset. For machine learning models to work correctly the stationarity in the data needs to be removed. Even in our dataset also there is a stationarity pattern observed in the dataset and some preprocessing steps are done in order to handle the stationarity.

B. Building Forecasting model

1) ARIMA: ARIMA stands for Auto Regressive Integrated Moving Average. 'AR' stands for auto regression, it is a simple model that uses previous observations to predict sales at a particular time stamp 't'. MA stands for moving average,it is done by calculating the average value until that time stamp 't' and dividing by the total number of timestamps. 'I' represents Integrated is used to remove stationarity from the dataset. In short ARIMA model works best for data where its mean and variance remains constant at any point of time to get better accuracy.

Data preprocessing:

a) Remove noise: As the data we are dealing is time series we need to check for the time series components like seasonality, trend and stationarity. The trend component is removed by denoising the dataset using wavelet denoising. So that the mean of data remains same, and the sudden hike in sale prices such as during Christmas and thanks giving does not effect the model's prediction.

b) Aggregate sales: First the data is joined with calender.csv file to get the information about important event days. Later the sale prices are aggregated for every store, state, department and category individually so that these dataframes can be used to predict the aggregated sales in each type of aggregation. The sample aggregation can be seen in Fig 3

c) Getting ARIMA parameters: The ARIMA parameters are p,q and d

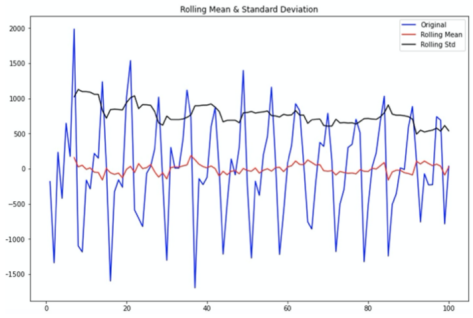

Fig. 4. After shifting lag features by factor of 1 stationarity in data

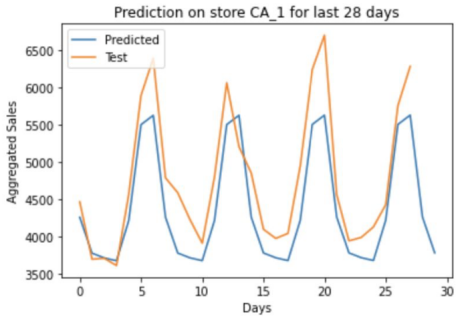

Fig. 5. Sample ARIMA aggregated sales prediction on store CA_1

- In order for the ARIMA model to work, stationarity in the data needs to be removed. To achieve this we tried different parameters by shifting the lags of the data by factor of 1, 7 and 8. After shifting the lags of the data, the dickey-fuller test is run to check the null-hypothesis. Null hypothesis determines whether there is a statistical relationship or significance between the variables in the data. By using the null-hypothesis we can check whether the stationarity exists or not in the data.The 4 specifies that the after shifting lag features by 1 the stationarity in the data is removed. Thus making the rolling mean and rolling standard deviation of the data more or less same through out the data and making the integrated parameter(d) as 1. This can be seen in Fig 4. But

  after removing the stationarity, there still seems to be a pattern left in the dataset which repeats after every week, so in order to accommodate this into the model, SARIMAX an extension of ARIMA which allows the seasonality in the data to be still there is used with seasonality parameter to be 7.
- To get the auto-regressive parameter we are using the 't-1' timestamp data to predict sales at timestamp 't' thus, making the parameter for auto-regression(p) to be equal to 1.
- The Moving Average parameter(q) is taken as 1 to consider the average until 't-1' timestamp to predict sales at timestamp 't'.

Train-test split: The sales-train-evaluation.csv contains details of sales over a period of 1941 days. For purpose of training the model, the last 28 days are split into test, the 28 days prior to test are split into validation and the remaining 1885 days are split into training data.

Model Training: The SARIMAX model is imported from statsmodels library. The train dataset after aggregation is built using the parameters obtained from preprocessing and making seasonality parameter as 7. First the data is aggregated into total sales ac cross different stores in all states namely CA_1, CA_2, CA_2, CA_3, CA_4, TX_1, TX_2, TX_3, WI_1, WI_2, WI_3 and predictions are made on test set for all stores. Next the data is aggregated into total sales across all the categories in the dataset namely HOUSEHOLD, FOODS and HOBBIES and the prediction on test set are made for all categories. Next the the data is aggregated into different departments like HOBBIES_1, HOBBIES_2, HOUSEHOLD_1, HOUSEHOLD_2, FOODS_1, FOODS_2, FOODS_2 and the predictions on each department is made on the test set. Next the sales prices of all the 30,490 items across all stores is dividing into train, validation and test is made and the trained with the same parameters.

Model Performance: The RMSE for sale prices on test data when aggregated storewise is 0.693. Sample prediction can be seen in Fig 5. RMSE for sale prices on test data when aggregated department wise is 0.94. RMSE for sale prices on test data when aggregated for each category is 1.08 The for sale prices on test data when predicted for each item is 1.06.

2) LightBGM:

Data preprocessing: The following lists the steps followed for preparing the initial data for training

- For training the models, the data is converted into pandas dataframe. After conversion, the columns are of type "object". These columns are converted into appropriate data types as per the values it stores.
- There are no missing values and only null values in the data. These values are converted to zero for numeric type columns and to empty string for columns with strings as values.
- The unnecessary or repeated columns are dropped.
- Models expect numerical values as input. So, categorical attributes are converted into numerical attributes.
- Multiple pandas dataframe are merged into a single dataframe.
- **Melting:** The dataframe is reshaped by converting multiple time series into a single time series IV-B.2. All 1941 sales columns are converted into two columns: "d" representing the days and "sales" representing each day's sales. Basically, the wide table is converted into a long one.

Feature Engineering: The classic features of the time series such as lag and rolling statistics features are created.

- **Lag features:** The past values are known as lags. For a

```
print(df_sales.shape)
df_sales[["item_id","d","sales"]].head(10)
```

(58327370, 8)

	item_id	d	sales
0	HOBBIES_1_001	d_1	0
1	HOBBIES_1_002	d_1	0
2	HOBBIES_1_003	d_1	0
3	HOBBIES_1_004	d_1	0
4	HOBBIES_1_005	d_1	0
5	HOBBIES_1_006	d_1	0
6	HOBBIES_1_007	d_1	0
7	HOBBIES_1_008	d_1	12
8	HOBBIES_1_009	d_1	2
9	HOBBIES_1_010	d_1	0

Fig. 6. Dataframe after melting: d and sales columns created

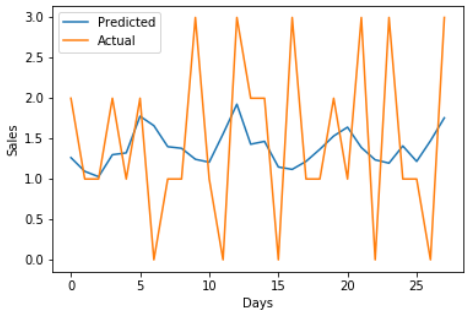

Fig. 7. Actual Vs Predicted sales by LGBM model

time t, lags are t-1, t-2 and so on. This is a new column created by shifting values of an existing column to a particular range n, where n can be any integer within row size, assuming past values as future sales.

- **Rolling statistics features:** These features are created by performing some statistical calculations (mean, median, sum, maximum, minimum, etc.,) on the existing columns. For example, a rolling mean feature is created by taking the mean of the values of the sliding window of size n, where n can be any integer within row size.

Train test split: In time series forecasting, splitting the data into the training and validation is different. Since, the time series depends on the ordering in the data, it should not be randomly shuffled that would disrupt the order. So, for this project, the data is splitted based on location. And, the last 28 days of sales is splitted as the validation data.

Building model: Once the data is preprocessed and the features are engineered, the training data is used to train the forecasting model using LightBGM (LGBM) V-A algorithm. LGBM has various parameters, some of them are explained below.

- **colsample_bytree:** It defines the number of features to be sampled for building a tree.
- **force_row_wise:** When the number of data points is large, and the total number of bins assigned is relatively small, this value should be set to "True". There is another option force_col_wise. If no value is given, the system would decide modelling choice as either row wise or column wise based on the number of the data points and the bin size. LGBM compresses memory by assigning the feature values to the certain number of bins, that defines the bin size. LGBM exploits many strategies to keep it light, this is one of them.
- **num_leaves:** It refers to the maximum number of leaves in a tree.
- **learning_rate:** This controls the speed of the model training. The higher learning rate makes huge steps while minimizing the error and the vice versa. This value should be greater than zero.
- **metric:** It is used to evaluate the performance of the trained model. The metric used is RMSE IV-C.1.
- **objective:** It refers the loss function which is used for optimization to minimize the error.
- **seed:** To avoid different splits in each experiment. For a particular seed value, the data split for each tree would be similar even in different runs.
- **num_boost_round:** It defines the number of iterations for the model training.
- **early_stopping_rounds:** It is a hard stop on the training process if there is not any significant improvement in error observed for the given number of rounds.
- **verbose_eval:** It defines the amount of information about the training process that can be printed.

Each parameter combination is used to train a model, and after training, it is evaluated on validation data using evaluation metrics IV-C.1 to find the parameters which predicts the sales accurately.

LGBM model provides predict function to make predictions on the unknown data, which is used to predict the sales. This figure 7 shows the plot of actual and predicted sales for 28 days.

3) FBProphet:

Data preprocessing: The following steps have been performed as a part of data preprocessing for Prophet model.

- The unnecessary columns from the sales_train_evaluation such as item_id, dept_id, cat_id, store_id, state_id are removed.
- Prophet models expects the input to be in ds and y format, where ds is datestamp and it should follow yyyy-mm-dd hh:mm:ss format and y is the value to be predicted. So, both dataframes are merged together to get the required input.

Train test split: Here, the same method is followed in the above algorithm to split the dataset into training and validation.

Building model: Once, the data is in required format, it is trained against FBProphet model to make prediction. In this project, we have passed some parameters to improve the efficiency of the model. Some of the parameters are:

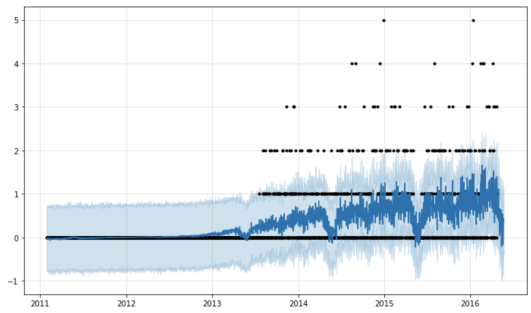

Fig. 8. Prediction results

- **growth:** The growth parameter is used for trend. In this project, linear growth parameter is selected.
- **seasonality_mode:** This parameter shows how the forecasts should be combined with the seasonality components. The default value of this parameter is set to additive seasonality. For our project, we have used multiplicative seasonality because the seasonality is not constant rather it grows with the trend.
- **interval_width:** For this project, it is set to 0.8 which is the default value.
- **fourier_order:** This parameters is one of the seasonality parameters and tells you the number of Fourier components each seasonality is composed of. In our project, we have set this to 20.

FBprophet provides predict function which is used to make predictions. This function returns a new dataframe which includes columns about the predicted date, value of trend, upper and lower values of seasonalities (daily, weekly and yearly). It also provides us with values of predictions with upper and lower bound among other values.

C. Model evaluation

1) Evaluation metrics: The trained model is evaluated on the test data to analyse the model performance on sales prediction. The evaluation metric used is RMSE, which is explained below. Root Mean Squared Error (RMSE) RMSE finds the prediction error by calculating the mean of aggregation of the squared difference between the predicted and the actual values of all items. The higher the model's prediction varies from the actual values, the higher it gets penalized by squaring the error. So, this metric helps to understand the difference of the model's prediction to the actual sales on average.

$$RMSE = \sqrt{\sum_{i=1}^{N}(x_i - x_i^-)^2/N} \quad (1)$$

2) Result Analysis: After training the models with different algorithms, each model's future sales prediction (for 28 days) is evaluated using RMSE. **Aggregation level:** Unit sales of all the products is predicted for 28 days by aggregating all the stores and states. The table I lists out the evaluation errors of all the models.

TABLE I
MODEL EVALUATION RESULTS BASED ON RMSE

Models	***RMSE errors***
ARIMA	1.06
LightBGM	2.2
FBProphet	0.61

The considered time series sales data has trends and seasonalities inbuilt to it. FBProphet handles time series components effectively with fourier transform, and that resulted in the better model performance compared to the other models based on the model error. The other models also provided the decent performance on the sales prediction. The quite higher error indicates the possibility of parameter tuning, which might help to improve the prediction performance.

V. TOOLS USED

A. Light Gradient Boosted Machines (LBGM)

It is an open source gradient boosting framework. It uses the boosting technique of ensemble learning. In boosting, a set of models learn sequentially based on the feedback from the previous models, thereby providing strong overall prediction. LGBM encompasses a bunch of trees whose individual results would be averaged over to get the final prediction result. It uses gradient descent as an optimization function to minimize the prediction errors. Moreover, it is referred as light, because of its high speed and ability to handle large scale of data. For this project, **lgbm open source implementation** is used.

B. Facebook Prophet

Facebook prophet is a forecasting open source tool developed by Facebook to do time series based prediction. Prophet is a method built on the basis of an additive model for forecasting time series data where nonlinear patterns match with annually, weekly, and regular seasonality, plus holiday impact. It works better with time series that have clear seasonal effects and historical data from many seasons. Prophet is immune to missing data and pattern changes, and usually manages outliers well [9]. In [9], authors mentioned that Prophet is used in many applications across Facebook to generate accurate forecasts and performs better than any other method in most cases. In this project, we have used Facebook prophet to predict sales of Walmart for next 28 days given the historical data.

C. ARIMA

ARIMA is one of the most effective machine learning model to predict time-series forecasting tasks. In order to run the model smoothly some dependent libraries need to be imported as well. The dependent libraries are python >3.6, Numpy>1.15, Scipy>1.2, Pandas>0.23, patsy>0.5.1 and Cython>0.29. All the dependent libraries can also be installed using anaconda environment or pip. In order for the ARIMA model to accommodate seasonality the extension of

ARIMA which is known as SARIMAX is also imported form statsmodels library itself.

D. Pandas

Pandas is one of the highly used library in the field of Machine Learning. It is an open source library and allows us to read data from various file formats like csv, json, etc into easily readable pandas dataframe. It gives the ability to do data manipulation such as merging files, reshaping, etc.

E. Numpy

Numpy another open source library stands for Numeric Python, gives us the flexibility to work with n-dimensional matrix operations.

F. Visualization libraries

We have used two powerful libraries matplotlib and seaborn which is built on top of matplotlib for visualizing our analysis done on the data.

G. Statsmodels

This library provides classes and functions for different statistical models. The ARIMA and SARIMAX models are imported from the statsmodels library.

All the libraries are open source, and can be installed from anaconda environment or pip.

VI. CONCLUSION

Building the time series forecasting model follows the steps which vary from the traditional machine learning models training. This project is intended to understand the components of the time series data and the ways to build models that make predictions about future sales. As everyone in the team was new to the time series data, it was challenging initially to understand the way it has been used to build the models. Every model requires the data to be processed in a certain way before giving it for training.

So, because of the time constraint and the time series forecasting complexity, instead of trying all the available algorithms, each team member planned to focus on one algorithm to understand all the stages of building a time series forecasting model using that learning algorithm. To do that, the time series algorithms such as ARIMA, LightBGM and FBProphet are chosen based on the results of the M5 competition [8].

A detailed study has been done on these algorithms, and the models are compared. Based on the calculated error, FBProphet provided the better performance with minimum error when compared with the other models. we tried the models individually but the error can be minimized by using the combination of models where two or three models used together to make the final prediction by averaging their individual values. Also, the model can be trained with altering the parameters with different values for minimizing the error.

This project helped us to understand the steps involved in building the time series forecasting models to make predictions about the future. These predictions can be used to make better decisions which would benefit the business.

VII. INDIVIDUAL CONTRIBUTION

The table II specifies the contribution of each team member to the project.

TABLE II
CONTRIBUTIONS

Task	***Team Members***
Exploratory Data Analysis	All
Data Preprocessing, Feature Engineering	All(Each model needed different preprocessing)
LightGBM	Saranya Visvanathan
ARIMA	Raaga Pranitha Kolla
FBProphet	Akansha Jajodia
Slides and Demo	All
Report	All

VIII. SUPPLEMENTARY MATERIALS

All the code implementations of the project are uploaded in the github. And, each file's description can be seen in readme file.

A. Github project link

https://github.com/raagapranitha/cmpe257-M5Forecasting

REFERENCES

[1] K. Ramírez-Amaro and J. C. Chimal-Eguía, "Machine learning tools to time series forecasting," in *2007 Sixth Mexican International Conference on Artificial Intelligence, Special Session (MICAI)*, 2007, pp. 91–101.

[2] S. Mehrmolaei and M. R. Keyvanpour, "Time series forecasting using improved arima," in *2016 Artificial Intelligence and Robotics (IRANOPEN)*, 2016, pp. 92–97.

[3] A. Lahouar and J. Ben Hadj Slama, "Random forests model for one day ahead load forecasting," in *IREC2015 The Sixth International Renewable Energy Congress*, 2015, pp. 1–6.

[4] Guanqun Dong, K. Fataliyev, and L. Wang, "One-step and multi-step ahead stock prediction using backpropagation neural networks," in *2013 9th International Conference on Information, Communications Signal Processing*, 2013, pp. 1–5.

[5] S. McDonald, S. Coleman, T. M. McGinnity, and Y. Li, "A hybrid forecasting approach using arima models and self-organising fuzzy neural networks for capital markets," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–7.

[6] I. Yenidoğan, A. Çayir, O. Kozan, T. Dağ, and Arslan, "Bitcoin forecasting using arima and prophet," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, 2018, pp. 621–624.

[7] Y. Cao and L. Gui, "Multi-step wind power forecasting model using lstm networks, similar time series and lightgbm," in *2018 5th International Conference on Systems and Informatics (ICSAI)*, 2018, pp. 192–197.

[8] "The m5 competition," Nov 2020. [Online]. Available: https://mofc.unic.ac.cy/m5-competition/

[9] B. S. J. Taylor and B. Letham, "Prophet: forecasting at scale," Sep 2018. [Online]. Available: https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/