Assignment 2

ML as a Service

Rushab Shah
Student ID: 14351985

06/10/2023

36120 - Advanced Machine Learning Application
Master of Data Science and Innovation
University of Technology of Sydney

https://quiet-savannah-28493-e3545a2f3e86.herokuapp.com/
https://github.com/rushman95/timeseries-sales-forecast

Table of Contents

1. Executive Summary

This report details the development and deployment of 2 machine learning models to forecast revenue. The first model aims to predict the revenue of an item sold at a store on a given date while the second model forecasts the total revenue across all stores and items for the next 7 days from the input date. Having tested several different algorithms, the final models to be selected were a CatBoost model for the item-store prediction and a Prophet forecasting model. These models were then deployed as an API on the Heroku platform and can be accessed via the link on the cover page.

2. Business Understanding

a. Business Use Cases

An American retailer operating 10 stores across California (CA), Texas (TX), and Wisconsin (WI), sells items from three different categories: hobbies, foods, and household.

The company faces challenges in accurately predicting sales revenue for specific items in individual stores on given dates, due to the complexity of the sales data. Being able to predict sales revenue would help the retailer to optimize inventory management, staffing, and marketing efforts to maximize revenue while minimizing costs.

The retailer also seeks to forecast the total sales revenue, across all items and stores, for a given date. Accurate sales revenue forecasting is crucial for supply chain management and financial decision-making. Fluctuating demand patterns, regional variations, and market dynamics make the trend difficult to capture, however accurate forecasts are needed to ensure sufficient inventory levels and efficient operations. By implementing forecasting models, the retailer can align its resources optimally and meet customer demand with precision.

b. Key Objectives

The project aims to develop to machine learning models:

- A model to accurately predict sales revenue for a specific item in a specific store on a given date.
- A model to accurately forecast the total sales revenue across all items and stores for 7 days, based on a given date input.

The final models will be deployed as an API, with Heroku, so that they can be tested or applied by any interested stakeholders, in a format that does not require much technical experience. The models can be used to enhance decision making by supply chain teams, inventory teams, finance teams and executive leadership.

3. Data Understanding

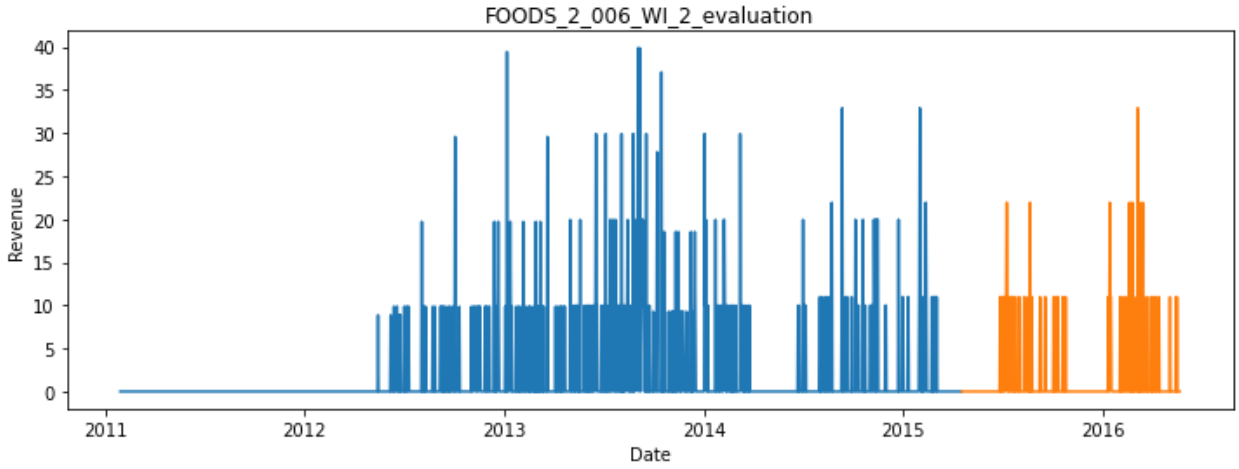
The dataset used in this project was composed from 5 csv files. A brief description of each file has been provided below:

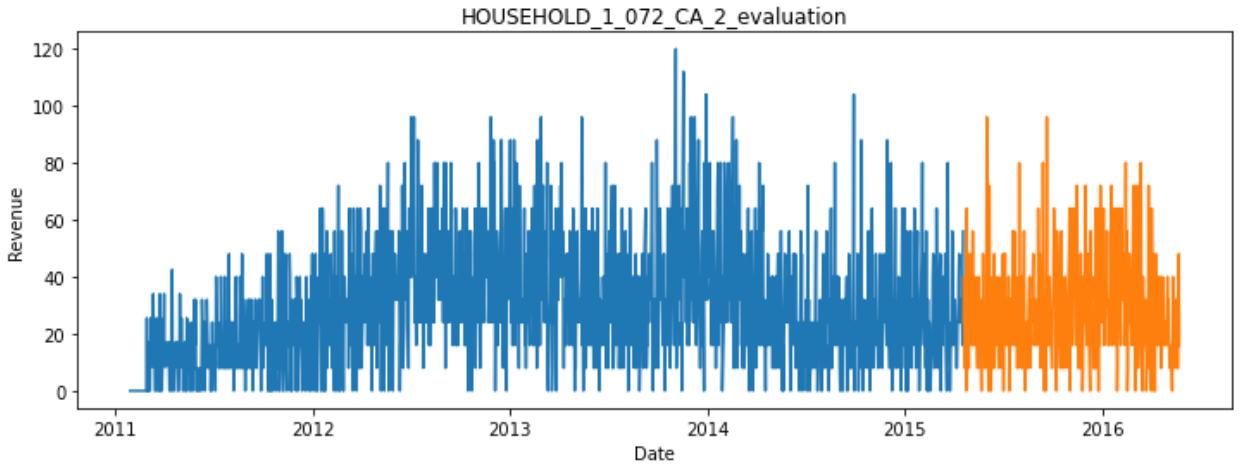
- Sales_train: It contained the training data, provided in a wide format with each row representing the quantity of daily sales of each unique product at each store.
- Sales_test: It contained testing data, also provided in a wide format with the same index as the training data (not provided) as the quantity of daily sales. The date range for the test data immediately followed the training data.
- Item_sell_prices: Weekly sell price of each item at each store.
- Calendar: Contained date mapping for sales_train and sales_test column names.
- Calendar_events: List of calendar events that could affect daily item sales.

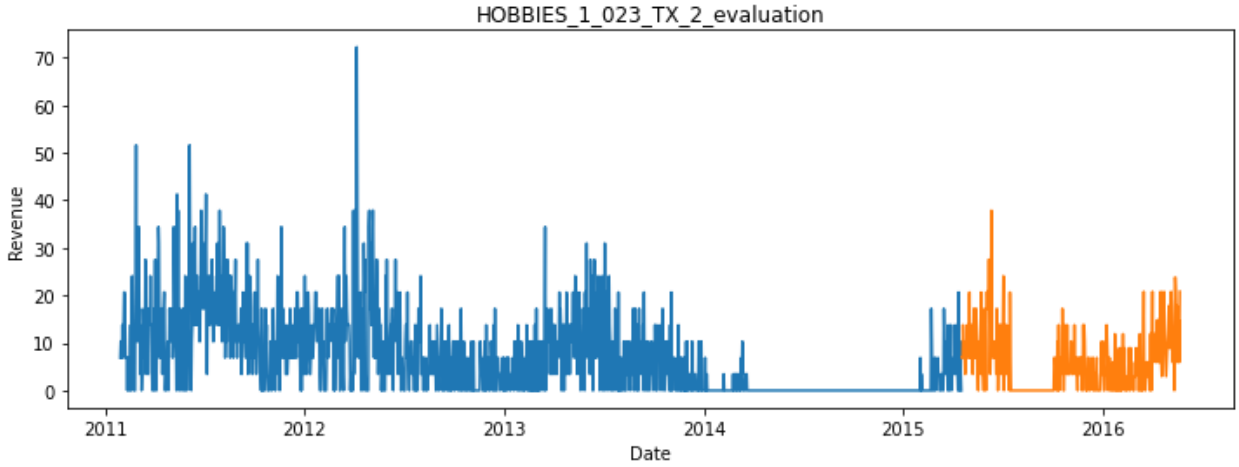
The dataset had a total of 30490 rows present, representing the total number of unique item-store combinations. No missing or duplicate values were found in the training and testing data, however some item prices were missing on some dates where no items were sold but did not affect the revenue.

In terms of Categories: FOODS had the most items (14370), followed by HOUSEHOLD (10470) and finally HOBBIES (5650). Each store had 3049 products, which meant that each item was present in each store.

The revenue has been plotted for several random item-store combinations across the entire time period to gain a better understanding of the trends present in the data:







The items all show very erratic patterns with a lot of daily fluctuations, some of which can be perceived as noise, which may negatively affect the performance of the models. Some items (like the first and third plot) have time periods where there were no sales suggesting that the product may have been taken off the market.

4. Data Preparation

Since the dataset was largely clean, minimal preprocessing was required. The initial step involved joining the tables and calculating revenue. The split between the training and testing data was preserved, as it had already been established. In the case of the training data, it was immediately transformed. However, for the testing data, an initial step involved combining it with the relevant ID columns. Subsequently, dates, selling prices, and calendar events were incorporated into both the training and test data frames to create a comprehensive representation of all sales transactions. Daily revenue figures were then computed for each item at each store.

Item-Store Revenue Prediction:

For the task of predicting item-store revenue, the target variable was straightforwardly the calculated revenue. Feature selection was primarily based on the deployment solution, emphasizing date-related and categorical variables. Year, month, day of the week, and day of the year were extracted from the date and introduced as numeric features for the model. Since the model assumes ordinality of these features, standardization was deemed unnecessary. In terms of categorical features, the ID features (item_id, cat_id, dept_id, store_id, state_id) were one-hot encoded within the model pipeline. Additionally, calendar events and their respective categories were included as categorical features, each tied to specific dates.

It's noteworthy that the model only required item, store, and date inputs, limiting its ability to utilize features like price for forecasting. Due to storage constraints in the free tier of heroku-postgres, which provided only 10,000 rows, price data could not be retained to generate contextual price-related features. Features such as price lags, rolling windows, and categorical means (mean price sold for a category across/within stores) could have enhanced the model's grasp of price fluctuations, but these could not be accommodated.

Total Revenue Forecast:

In the case of total revenue forecasting, the target variable represented the total revenue across all stores and items for each day. Since a time series model was employed that solely required the date and target, no further feature engineering was deemed necessary in this context.

5. Modeling

a. Item-Store Revenue Prediction

To predict item-store revenue, various regression-based algorithms were explored using a predefined training dataset and their performance evaluated on the test data. The algorithms considered included Linear Regression, XGBoost, and CatBoost. The feature selection process encompassed all engineered date features and categorical features linked to each item-store ID and calendar events. Several models were trained both with and without calendar event features. A sci-kit learn pipeline was employed to one-hot encode the categorical features and merge them with numeric features, preparing the dataset for model ingestion.

The use of an on-demand model service was also contemplated, recognizing that time series models might excel in capturing fluctuations. However, this approach was not tested due to the unavailability of a feasible method for storing the required data (at no cost).

Ultimately, the chosen model was CatBoost (without the calendar features), with the learning rate set to 0.224.

b. Total Revenue Forecast

For total revenue forecasting, both ARIMA and Prophet models were assessed, with Prophet emerging as the more user-friendly choice, particularly for handling holidays, leading us to select it as the final model.

In the case of the ARIMA model, an auto ARIMA approach was employed to identify the optimal parameters for order (autoregressive lags, order of differences, and moving average) and seasonality order. This allowed for customization and effective trend capturing. However, it struggled to capture fluctuations caused by events. To address this, a SARIMAX model could have been used, but it would have required additional preprocessing and the specification of holidays for the test set, introducing complexity.

In contrast, the Prophet model seamlessly fitted the data without the need for specific parameter tuning. It effortlessly incorporated calendar events into the holidays argument, effectively capturing the trend and displaying the added advantage of detecting event-related surges, particularly around the new year. Notably, the Prophet model did not necessitate the specification of holidays in the test set, streamlining the forecasting process.

6. Evaluation

a. Evaluation Metrics

The metrics used to evaluate the time series models were the coefficient of determination (R2), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

1. R-squared (R^2):

THE coefficient of determination, measures the proportion of the variance in the dependent variable (target) that is explained by the independent variables (features) in the model. R^2 provides an overall assessment of model fit but should be supplemented with other metrics, especially when dealing with time-dependent data because it does not consider the temporal dependencies.

2. Root Mean Squared Error (RMSE):

RMSE measures the square root of the average squared differences between predicted and actual values, in the same units as the target variable, making it interpretable and comparable. In time series analysis, RMSE is a useful metric because it quantifies the model's prediction accuracy while considering the magnitude of errors.

3. Mean Absolute Error (MAE):

MAE measures the average absolute differences between predicted and actual values, also in the same units as the target variable. MAE is less sensitive to extreme outliers since it does not square the errors. MAE is useful for understanding prediction accuracy while being less affected by extreme errors, making it a robust choice for assessing models on time series data.

The Mean Absolute Percentage Error was also calculated, but cannot be used because it is extremely large, probably as a result of the huge dataset size (47 million rows).

b. Results and Analysis

The results for each model have been discussed below:

1. Item - Store Prediction:

Evaluation metrics for each model on the test set data have been summarized in the table below:

Model	R^2	RMSE	MAE
Linear Regression	0.215	10.010	4.325
XGBoost	0.343	9.162	4.212
ADABoost	-0.140	12.060	5.311
CatBoost	0.442	8.440	4.078
CatBoost (without events)	0.456	8.333	4.069

In this scenario, a noticeable divergence in algorithm performance becomes evident. CatBoost stands out prominently, surpassing all other models, particularly in terms of R2, where it achieved the highest score. XGBoost follows as the second-best performer. Additionally, CatBoost exhibited lower MAE and RMSE values compared to all other models, indicating a superior accuracy level. What's intriguing is that CatBoost, even without the inclusion of calendar events, managed to perform marginally better. However, it's important to note that despite these achievements, the overall model fit remained relatively modest, with none of the models achieving a coefficient of determination greater than 0.50. This can be attributed to the inherent unpredictability often associated with time series data.

CatBoost's remarkable performance can be attributed to its proficiency in handling categorical data, a trait that it consistently demonstrated in this context. Due to the lengthy training periods of approximately one hour for each model, the optimization of hyperparameters was somewhat limited.

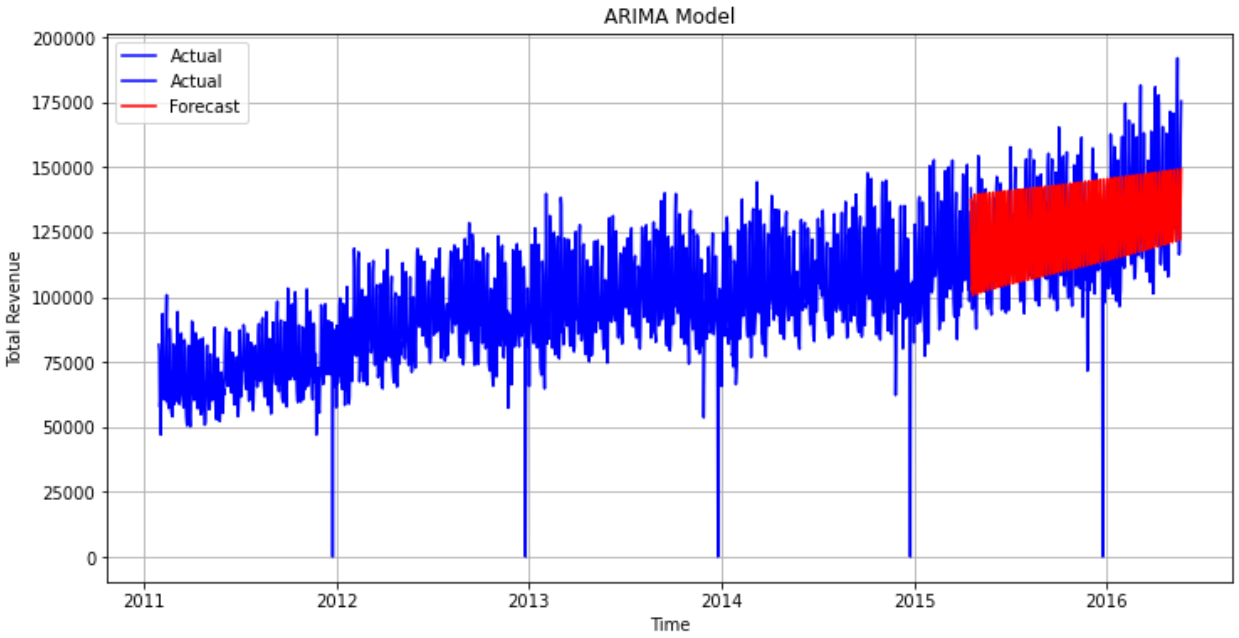
2. **Total Revenue Forecast:**

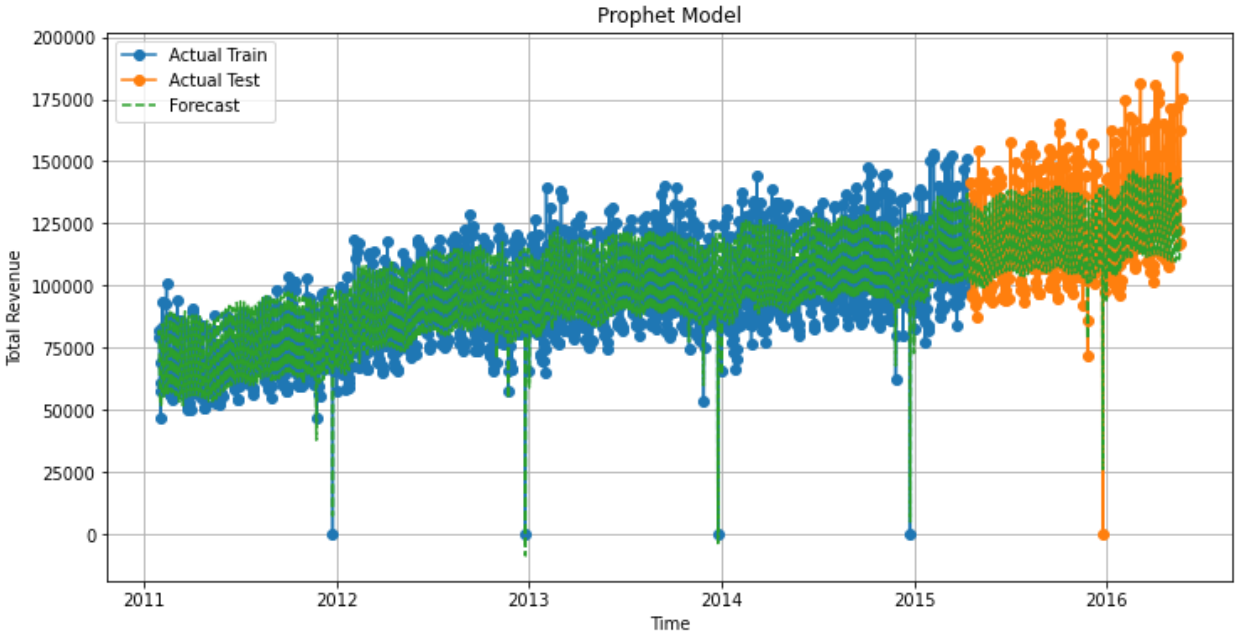
Evaluation metrics for the total revenue forecasting models on the test set are displayed below.

Model	R^2	RMSE	MAE	MAPE
ARIMA	0.481	15503.326	11460.539	10.005
Prophet	0.683	12123.469	9248.817	2.190

The Prophet model demonstrated superior performance over the ARIMA model across all evaluation metrics, with a noteworthy advantage observed in the mean absolute percentage error. These findings signify an enhancement in the coefficient of determination, indicating that the time series models excel in capturing the underlying data trends.

For a better understanding of each model’s performance, the following graphs were plotted.





The Prophet model exhibits a notable ability to detect event-related trends, particularly in proximity to the onset of each new year. In contrast, the ARIMA model did not incorporate any holidays or events explicitly. While both models effectively capture the prevailing trend, neither model identified the exceptional surge in 2016, likely due to its deviation from the relatively consistent growth observed in preceding years. The prophet model was selected for the final deployment.

c. Business Impact and Benefits

The project has effectively met its business objectives by delivering two models: one for predicting item-store revenue and another for forecasting total weekly revenue based on a specific date. However, it's crucial to approach the predictions from both models with caution due to the relatively low coefficient of determination.

To enhance the project's performance and capabilities, the implementation of a database for data storage is recommended. This database can be utilized to construct additional features, such as categorical means and price lags. These features would enable the model to better understand the impact of price fluctuations on revenue. Moreover, having a database in place would facilitate real-time access to this valuable information, further optimizing the model's predictive abilities.

d. Data Privacy and Ethical Concerns

There were several privacy and ethical concerns that were considered for this project as sensitive business data is concerned.

1. Data Privacy: The project involves the use of sales transaction data, but customer data had already been removed to protect customer privacy. The sales data had also been completely deidentified in terms of the names of the items sold and specific store locations so no sensitive information was revealed that could be traced back to the business.
2. Bias and Fairness: Before deployment, the model needs to take bias and fairness into account. Negative or inaccurate predictions can result in poor decision making leading to losses for the business, which in turn could affect staff and operations. Any issues should be promptly addressed.
3. Safeguarding Trade Secrets: Sensitive data such as the prices of the different products should be protected, as revealing them may lead to unfair disadvantages for the company. The deidentified products help with this, however all data should be protected.

7. Deployment

The API has been deployed on the Heroku platform using a Docker file containing container building instructions. You can easily access and interact with the API through a web browser. It was developed using FastAPI, which has facilitated the creation of an intuitive interface for the API, featuring four key endpoints:

- '/' (GET): Displaying a brief description of the project objectives, list of endpoints, expected input parameters and output format of the model, link to the Github repo related to this project

`https://quiet-savannah-28493-e3545a2f3e86.herokuapp.com/`

- '/health/' (GET): Returning status code 200 with a string with a welcome message of your choice

`https://quiet-savannah-28493-e3545a2f3e86.herokuapp.com/health/`

- '/sales/national/' (GET): Returning next 7 days sales volume forecast for an input date

`https://quiet-savannah-28493-e3545a2f3e86.herokuapp.com/sales/national?date=YYYY-MM-DD`

- '/sales/stores/items/' (GET): Returning predicted sales volume for an input item, store and date

`https://quiet-savannah-28493-e3545a2f3e86.herokuapp.com/sales/stores/item?item_id=ITEM_ID&store_id=STORE_ID&date=YYYY-MM-DD`

The endpoints in each URL have input arguments highlighted in green. Please ensure that the input arguments are provided in the correct case and date format. Comprehensive documentation for the application can be found here:

`https://quiet-savannah-28493-e3545a2f3e86.herokuapp.com/docs/`

A major benefit of deploying the application on Heroku is its effortless integration with Github, simplifying both the initial setup and subsequent updates, making the process quick and straightforward.

8. Conclusion

In conclusion, the completion of this project yielded valuable insights and findings. The CatBoost regression model emerged as the top performer, surpassing all other models by a considerable margin, even with a reduced feature set. This underscores its effectiveness in predicting item-store revenue on specific dates, possibly due to its adept handling of the highly categorical nature of the data.

Notably, time series models were not explored for this task due to deployment constraints related to database storage limitations, making it impractical to manage a multitude of different models.

Regarding total revenue forecasting, the Prophet model demonstrated superior performance when compared to the ARIMA model, despite running without any customized hyperparameters. Additionally, the Prophet model effortlessly accommodated calendar events, requiring minimal preprocessing efforts, thus facilitating model construction and testing.

However, it's essential to exercise caution as not all evaluation metrics aligned in favor of model performance, with the coefficient of determination indicating a suboptimal fit. Consequently, prudent usage of the model is recommended. Further enhancements can be achieved by establishing an Extract, Transform, Load (ETL) pipeline for continuous data updates, thereby augmenting its predictive capabilities.

Lastly, the successful deployment of the API on Heroku is a significant milestone, offering accessibility through the link provided on the report's cover page. Maintenance and updates are streamlined through straightforward pushes to the associated Github repository, ensuring the model remains up-to-date and relevant.

9. References

Catboost. (n.d.). *CatBoost*. CatBoost. Retrieved October 6, 2023, from https://catboost.ai/en/docs/

Heroku. (2023). *Documentation*. Heroku Dev Center. Retrieved October 6, 2023, from https://devcenter.heroku.com/categories/reference

Lendave, V. (2021, November 1). *A Guide to Different Evaluation Metrics for Time Series Forecasting Models*. Analytics India Magazine. Retrieved October 6, 2023, from https://analyticsindiamag.com/a-guide-to-different-evaluation-metrics-for-time-series-forecasting-models/

META. (n.d.). *Quick Start | Prophet*. Meta Open Source. Retrieved October 6, 2023, from https://facebook.github.io/prophet/docs/quick_start.html