

# Assignment

# ML as a Service

---

2

DRISHYA LAL CHUKE  
Student ID: 25076922

Oct 7, 2024

Github Username	DrishyaChuke Drishyachuke98@gmail.com
Github Repos	Experiment Repo: <a href="https://github.com/DrishyaChuke/adv_mla_at2">https://github.com/DrishyaChuke/adv_mla_at2</a> Package Repo: <a href="https://github.com/DrishyaChuke/my_krml_25076922">https://github.com/DrishyaChuke/my_krml_25076922</a> API Repo: <a href="https://github.com/DrishyaChuke/api_at2">https://github.com/DrishyaChuke/api_at2</a> StreamLit Repo: <a href="https://github.com/DrishyaChuke/app_at2">https://github.com/DrishyaChuke/app_at2</a>
URLs	Backend: N/A Frontend: <a href="https://app-at2.onrender.com">https://app-at2.onrender.com</a>

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

## Table of Contents

1.	Executive Summary	2
2.	Business Understanding	3
a.	Business Use Cases	3
3.	Data Understanding	4
4.	Data Preparation	5
5.	Modeling	6
a.	Approach 1 : Baseline Model Using Linear Regression	6
b.	Approach 2 : Predictive Model Using LightGBM	6
c.	Approach 3: Forecasting Model Using Prophet	7
d.	Approach 4: Predictive Model Using Random Forest	8
6.	Evaluation	9
a.	Evaluation Metrics	9
b.	Results and Analysis	9
c.	Business Impact and Benefits	10
d.	Challenges	10
e.	Data Privacy and Ethical Concerns	11
7.	Deployment	12
8.	Conclusion	15

## 1. Executive Summary

This project focuses on developing and deploying machine learning models for an American retailer with 10 stores across California, Texas, and Wisconsin. The main objective is to create two models: a **predictive model** to estimate sales revenue for specific items in stores on given dates, and a **forecasting model** to predict total sales across all stores for the next 7 days.

The **predictive model** utilizes the LightGBM algorithm, providing insights for inventory and demand management, while the **forecasting model** leverages the Prophet library for time-series forecasting to support operational planning. Both models aim to optimize inventory, enhance profitability, and improve overall resource allocation.

The entire workflow—data preparation, model development, evaluation, and deployment—was handled through a structured process. The models were deployed using **FastAPI** and **Streamlit**, with Render as the hosting platform, ensuring accessibility via a user-friendly web interface. The code, experiment records, and deployment details were managed in a GitHub repository to facilitate collaboration and reproducibility.

The outcomes include increased **accuracy in sales predictions**, improved **operational efficiency**, and the availability of real-time predictions via an accessible **web application** for business stakeholders.



## 2. Business Understanding

### a. Business Use Cases

The project aims to address two key use cases for an American retailer with 10 stores spread across California, Texas, and Wisconsin. The first use case involves **predicting sales revenue** for a specific item in each store on a given date. This information is crucial for **inventory management** and **pricing strategy**, enabling the retailer to optimize stock levels and reduce costs. The second use case is **forecasting total sales revenue** across all stores for the next 7 days, providing insights for **resource planning** and ensuring efficient allocation of staff and inventory.

### b. Key Objectives

1. Develop a **predictive model** to estimate sales revenue for specific items and dates, thereby aiding in data-driven inventory and demand management.
2. Create a **time-series forecasting model** to predict sales for the next 7 days, allowing for better planning of store operations.
3. **Stakeholders:** Store managers, supply chain teams, and business decision-makers will benefit from improved decision-making through accurate sales forecasts, leading to optimized operational efficiency and reduced wastage.

These models are expected to improve the retailer's ability to manage inventory, reduce stockouts or overstocking situations, and ultimately **enhance profitability**.



### 3. Data Understanding

**Dataset Overview:** The dataset consists of multiple files that provide comprehensive information about the retailer's operations. These include:

1. **Training Data:** Contains historical sales data for specific items in stores across California, Texas, and Wisconsin.
2. **Evaluation Data:** Used for model validation and performance assessment.
3. **Calendar Data:** Includes relevant date information, helping capture seasonality and time-related features.
4. **Events Data:** Provides information about holidays or special events that could impact sales.
5. **Items Weekly Price Data:** Contains the weekly selling price of items, which helps in understanding the price sensitivity of sales.

**Features Overview:** The main features used in the analysis include:

- **Store and Item Identifiers:** Unique IDs for stores and items that help in distinguishing sales data for different combinations.
- **Time-related Features:** Date, month, year, day of the week, and event information to capture seasonal effects.
- **Sales Revenue:** The target variable used for prediction and forecasting.
- **Item Price:** Weekly price data that aids in understanding the relationship between price changes and sales.

**Data Limitations:** The dataset required was divided into different dataset, which needed to be merged. So, looking into our business question, we pick the relevant feature and create the datasets that the models require. Also, the dataset is huge which might hamper the function for some models since my setup might not handle this volume of data to run efficiently.



## 4. Data Preparation

For Model only training data was used for modelling as the data is huge. As we proceed with data preparation and function to reduce memory usage was called from my package for efficient running of the kernel.

- Data Reshaping:
  - The sales data in training dataset is in a wide format, where each day is represented by a separate column. This may need to be reshaped into a long format where each row represents the sales for a specific day. Example: Convert the daily columns (d\_1, d\_2, ..., d\_1541) into a single date and sales column for each item-store combination.
  - Reshaping the sales data. We'll use **pd.melt()** to transform the daily sales columns into a long format.
- Data Merging:
  - Merging the calendar and calendar\_events datasets with the sales data to include dates and event information. This will allow you to capture trends like spikes in sales during events.
  - Merging the item prices dataset to include price information for each item-store-week combination.
  - Now that the sales data is reshaped, we can merge it with the calendar.csv to replace the d column (e.g., d\_1, d\_2) with actual dates.
- After the merge, certain number of rows on 'sell\_price' features has null which was considered as missing values. So, all the null value was converted to 0.
- Now that number of sales and sell price of all the product was derived, total price was calculated ( sales \* sell price = total price).
- From date feature we generated features like day of week, month and year for compatibility of our dataset to run in our predictive model.
- Forecasting Model: For forecasting model, only date and total sales feature was taken for time-series analysis.
- Predictive Model: For predictive model Id features like item id, store id and time features like day of week, month and year were taken as the model will predict the sales revenue for a given item in a specific store at a given date.



## 5. Modeling

### Overview:

The modeling phase involved creating two different machine learning models: a predictive model to estimate sales revenue for specific items in stores on a given date, and a forecasting model to predict total sales for the next 7 days. Different approaches and algorithms were explored to identify the best-performing model for each task.

### a. Approach 1 : Baseline Model Using Linear Regression

- **Algorithm:**

- A **Linear Regression** model was used as the baseline predictive model to estimate sales revenue for specific items in stores. This simple regression model helps establish a benchmark for model performance.

- **Model Description:**

- The features used included store\_id, item\_id, day\_of\_week, month, and year. The **numerical features** (total\_sales, day\_of\_week, month, year) were scaled using StandardScaler, while the **categorical features** (store\_id and item\_id) were label-encoded to prepare the data for training.

- **Training Process:**

- The linear regression model was trained on the scaled and encoded dataset to provide a quick and interpretable baseline. The goal was to understand how well a simple model could perform with the given features.

- **Evaluation:**

- The baseline model's performance was evaluated using **Root Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)**. This provided a benchmark against which more complex models (LightGBM, Random Forest) could be compared.

### b. Approach 2 : Predictive Model Using LightGBM

- **Algorithm:**

- **LightGBM** was chosen for the predictive model due to its ability to handle large datasets efficiently and its support for categorical variables. LightGBM is a gradient boosting framework that provides fast training, supports missing values, and handles large-scale data effectively.
- **Model Description:**
  - The predictive model aims to estimate sales revenue for a given item in a specific store on a given date. The features used include store\_id, item\_id, day\_of\_week, month, and year.
- **Hyperparameter Tuning:**
  - The hyperparameters were tuned to optimize the model's performance, including num\_leaves, learning\_rate, and n\_estimators. A combination of **random search** and **cross-validation** was used to find the best values.
- **Training Process:**
  - The training data was split into training and validation sets. The LightGBM model was then trained on the training set, with early stopping applied to prevent overfitting. The model evaluation metrics included **Root Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)** to measure prediction accuracy.

### c. Approach 3: Forecasting Model Using Prophet

- **Algorithm:**
  - **Prophet** was used for the time-series forecasting model. Prophet is an open-source library developed by Facebook, designed for business forecasting applications where trends and seasonality are present. It is particularly effective for handling missing data, outliers, and changing trends.
- **Model Description:**
  - The forecasting model aimed to predict the total sales across all stores for the next 7 days. The features used for this model were historical total\_sales data aggregated daily.
- **Training Process:**



- The Prophet model was trained on historical aggregated sales data, with date as the ds (timestamp) and y as the target (total sales). Prophet automatically decomposes the data into trend, seasonal, and holiday components, which makes it highly suitable for capturing periodic patterns in sales data.
- **Hyperparameter Tuning:**
  - Hyperparameters such as seasonality\_mode (additive vs. multiplicative) and changepoint\_prior\_scale were tuned to adjust how sensitive the model is to trend changes.

#### d. Approach 4: Predictive Model Using Random Forest

- **Algorithm:**
  - **Random Forest** was also considered for the predictive model to estimate sales revenue for specific items in stores. Random Forest is an ensemble learning method that works well with tabular data and handles non-linearity effectively.
- **Training Challenges:**
  - During training, the **Random Forest** algorithm struggled with the size of the dataset, resulting in extensive training time without producing an output. Due to the dataset's large size, Random Forest became computationally expensive, and no results were obtained even after a considerable amount of time.
- **Reason for Discontinuation:**
  - Given the large volume of data and the time complexity of Random Forest, this approach was not feasible for real-time or even offline prediction use cases. Hence, the model was not pursued further, and LightGBM was preferred due to its efficiency and speed.



## 6. Evaluation

### Model Comparison:

- **Baseline Model:** The linear regression model served as a simple and interpretable benchmark. The results provided insights into the limitations of a linear approach and indicated that a more complex model was needed for better accuracy.
- **LightGBM:** Demonstrated strong performance for the predictive task due to its efficiency and ability to handle feature interactions.
- **Prophet:** Excelled at time-series forecasting for the next 7 days, effectively capturing trends and seasonality.
- **Random Forest:** Training was computationally expensive and infeasible for the large dataset, leading to discontinuation.

### a. Evaluation Metrics

The evaluation of the predictive models was carried out using **Root Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)**. These metrics were chosen as they provide insights into the average magnitude of the prediction errors (MAE) and the variability of errors (RMSE). A lower value indicates better performance, and comparing the baseline model with LightGBM helps assess the improvements achieved.

### b. Results and Analysis

#### Baseline Model (Linear Regression)

- **Training Metrics:**
  - **RMSE:** 0.9980
  - **MAE:** 0.4775

#### Analysis:

The baseline model's training metrics indicate that the model was able to provide a reasonable initial approximation of the sales revenue. However, the relatively high RMSE and MAE highlight its inability to capture complex relationships and interactions among the features, which are often necessary in a retail sales context where factors like seasonality, promotions, and external events can significantly impact sales.

#### LightGBM Predictive Model

- **Validation Metrics:**
  - **RMSE:** 6.3694
  - **MAE:** 3.0269
- **Test Metrics:**
  - **RMSE:** 6.3937
  - **MAE:** 3.0248

#### **Analysis:**

The LightGBM model shows a significant increase in RMSE and MAE values compared to the baseline model, primarily due to the complexity of the data and the prediction task at hand. However, it outperforms the baseline in terms of flexibility, handling large datasets, and capturing non-linear relationships.


- **Generalization:** The difference between the **Validation RMSE** and **Test RMSE** is very small, indicating that the LightGBM model is not overfitting and generalizes well to unseen data.
- **Performance Improvement:** Compared to the baseline model, LightGBM captures more intricate feature interactions, resulting in improved accuracy in predicting the actual sales revenue.

### **c. Business Impact and Benefits**

- **Baseline Model:** While the baseline model provided a quick initial benchmark, it was unable to capture complex interactions and exhibited a higher error rate. The inability of linear regression to handle the dataset's complexity limited its usefulness for precise inventory and demand planning.
- **LightGBM Model:** The **higher accuracy** of the LightGBM model in both validation and testing phases makes it more suitable for decision-making in inventory and resource allocation. By providing better predictive performance, the model can help optimize stock levels, reduce waste, and ensure that customer demand is adequately met.

### **d. Challenges**

- **Complexity of Sales Data:** The dataset's size and complexity made it difficult for traditional models like linear regression to perform well. The non-linear relationships and categorical features required a more sophisticated algorithm to achieve meaningful predictions.

- 
- **Random Forest Limitations:** Random Forest was initially considered but proved infeasible due to the high computational demands and extended training time on the large dataset, further validating the choice of LightGBM.

#### e. Data Privacy and Ethical Concerns

- **Privacy of Data:** The dataset used contained sales information and store identifiers, with no sensitive customer data involved, reducing privacy concerns. The project adhered to data protection standards, ensuring no Personally Identifiable Information (PII) was used in modeling or analysis.
- **Fairness:** Models were evaluated to ensure that store and item features were treated fairly, without bias toward specific stores or regions. The use of categorical encodings helped achieve this goal.



## 7. Deployment

### Deployment Overview:

The deployment process involved integrating the models developed for sales forecasting and sales prediction into a usable application for business stakeholders. **FastAPI** and **Streamlit** were used as the main tools to provide interactive endpoints and a user-friendly web interface, while **Render** was chosen as the deployment platform to host the application.

### Deployment Workflow:

#### 1. Model Integration:

- The **predictive model** (LightGBM) and the **time-series forecasting model** (Prophet) were saved in .pkl format after training.
- These models were loaded into the application using **joblib** to serve predictions and forecasts when required.

#### 2. API Endpoints with FastAPI:

- **FastAPI** was used to create RESTful endpoints to facilitate interaction with the models:
  - `/health/`: A health check endpoint that confirms the API is up and running.
  - `/sales/national/`: An endpoint for forecasting total sales for the next 7 days using Prophet.
  - `/sales/stores/items/`: An endpoint to predict sales for a specific store and item on a given date using LightGBM.
- These endpoints ensured seamless integration and allowed stakeholders to interact with the models programmatically, making predictions for planning purposes.

#### 3. Web Interface Using Streamlit:

- **Streamlit** was used to provide an easy-to-use graphical interface for non-technical users to interact with the models.
- Users could input relevant data (e.g., store ID, item ID, and date) and get predictions directly through the web application, improving accessibility.

#### 4. Deployment on Render:

- **Render** was used as the cloud platform for hosting the web application. Render provided an easy setup for Docker-based deployment and enabled automatic redeployment with each update pushed to the GitHub repository.
- The Docker image, containing all necessary dependencies for running FastAPI and Streamlit, was built locally and then published to Docker Hub. Render pulled the Docker image and hosted it to ensure scalability and reliability.

#### Challenges and Recommendations:

- **API Deployment:** Faced Issue on the Docker while pushing in docker hub, kept failing to push the image due to which render deployment failed for FastAPI.i
- **Library Issues:** During deployment on Render, there were some library compatibility issues, such as missing system dependencies (libgomp.so.1) required by LightGBM. This was resolved by updating the Dockerfile to include the necessary system packages (libgomp1).
- **Dependency Management:** Managing dependencies across multiple platforms (local vs. Render) required careful inclusion of all Python packages in requirements.txt and installation of additional system libraries in the Docker container.
- **Deployment Recommendations:**
  - **Use Version Control:** Consistent use of GitHub for version control helped maintain reproducibility and provided easy integration with Render.
  - **Automation of Deployment:** Setting up automatic redeployment with each GitHub push ensured that updates were immediately reflected in the production environment without manual intervention.
  - **Environment Variables:** Use environment variables to manage secrets and sensitive configuration settings during deployment for enhanced security.

#### Deployment Impact:

- The deployed application allowed **store managers and analysts** to interact with the models via a web interface to make informed decisions.
- Predictions could be accessed in real-time, facilitating **inventory management, sales planning, and resource allocation**.



## 8. Conclusion

### Summary of Key Findings:

The project successfully developed and deployed two models for the American retailer: a **predictive model** for item-level sales estimation and a **forecasting model** for predicting overall sales across stores. The predictive model, implemented using **LightGBM**, provided more accurate and flexible results compared to the baseline **Linear Regression** model, making it well-suited for optimizing inventory and demand management. The **Prophet** model demonstrated its effectiveness in capturing seasonality and trends, providing reliable sales forecasts for the next 7 days to aid in operational planning.

### Project Success:

- The **LightGBM model** significantly outperformed the baseline model, highlighting the importance of using advanced machine learning algorithms to capture the complexities of sales data.
- Deploying the models with **FastAPI** and **Streamlit** ensured easy accessibility for stakeholders through both REST APIs and an interactive web interface.
- Hosting on **Render** provided a scalable and reliable solution for deploying machine learning models to production, making real-time predictions possible for the business.


### Challenges Overcome:

- **Handling Large Data:** The dataset's size was a challenge for models like **Random Forest**, which struggled to train efficiently. This validated the choice of **LightGBM** for scalability and speed.
- **Deployment Issues:** Compatibility issues with missing system libraries (libgomp.so.1) during deployment were addressed by updating the Dockerfile to include necessary dependencies.

### Future Work and Recommendations:

- **Enhance Feature Engineering:** Further exploration of feature engineering, such as adding lagged features or rolling averages, could improve the model's accuracy in predicting sales.
- **Deploy Model Monitoring:** Implement a model monitoring system to track model performance over time and ensure accuracy in changing market conditions.



- 
- **Scalable Infrastructure:** Consider using container orchestration tools like Kubernetes if the prediction demands increase significantly, providing better load balancing and scaling options.

In conclusion, the project successfully met its objectives by delivering a robust, accessible, and scalable solution for sales forecasting and prediction, ultimately supporting the retailer in enhancing **profitability**, **operational efficiency**, and **decision-making**.

