Assignment 2
ML as a Service

Saumya Bhutani
Student ID: 14360820

10th Oct 2023

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

Table of Contents

1. Executive Summary

This project is centred around developing a robust predictive and forecasting machine learning model to predict sales revenue accurately. The significance of this endeavour lies in its potential to revolutionize the way retail businesses operate.

The problem at hand is to create two models:

- A predictive model that can provide accurate sales revenue predictions for an item at a particular store, on any given date.
- A forecasting model that on a particular date, forecasts revenue predictions for the next 7 days.

These projects were initiated in response to the challenges faced by retail businesses, which include inventory management, demand forecasting, pricing optimization, store performance evaluation, and effective marketing campaigns.

Outcomes Achieved and results of the project.

Through this project, a significant milestone has been achieved with the development of a predictive model and a forecasting model that promises transformative changes in retail business operations such as

- Inventory Management: Accurate revenue predictions help optimize inventory, preventing overstock and understock scenarios.
- Demand Forecasting: The model informs production, distribution, and marketing decisions through precise sales forecasts.
- Pricing Strategies: Understanding sales trends allows us to fine-tune pricing for better revenue and margins.
- Store Performance: Insights into top-selling items in different regions improve store performance.
- Marketing Campaigns: The model guides effective marketing campaigns targeting specific items or categories.

2. Business Understanding

a. Business Use Cases

1. Inventory Optimization: The project's primary application is in inventory management. Accurate sales revenue predictions enable the business to optimize inventory levels, ensuring the right quantity of each item is stocked to meet demand while minimizing overstock or understock situations.

2. Demand Forecasting: The project aids in demand forecasting for individual items. By predicting future sales accurately, the business can make informed decisions about production, distribution, and marketing strategies, resulting in efficient resource allocation.

3. Pricing Strategies: The project helps in adjusting pricing strategies. Understanding sales trends for specific items allows the business to tailor pricing for higher revenue and improved profit margins.

The project was motivated by several challenges and opportunities:

- Inventory Challenges: Overstocking and understocking can tie up capital and lead to lost sales. Accurate sales predictions offer an opportunity to optimize inventory and prevent these issues.
- Demand Uncertainty: Fluctuations in demand can lead to resource inefficiencies. The project addresses this challenge by providing reliable demand forecasts.
- Competitive Pricing: Pricing is a critical aspect of revenue generation. Understanding sales trends allows for competitive pricing strategies.
- Marketing Effectiveness: Effective marketing campaigns are essential for business growth. Tailored promotions based on accurate predictions improve marketing campaign effectiveness.

b. Key Objectives

- Accurate Revenue Predictions: Develop precise models for sales revenue forecasting to support business functions.

- Optimized Inventory: Achieve inventory optimization by reducing overstock and understock situations.
- Effective Demand Forecasting: Provide efficient demand forecasting for items to plan production, distribution, and marketing.
- Improved Pricing: Enhance pricing strategies by understanding sales trends and maximizing revenue.

Identification of Key stakeholders

- Business Executives: Require accurate revenue predictions for resource allocation and business planning.
- Inventory Managers: Need optimized inventory management to reduce costs and stockouts.
- Production Managers: Rely on demand forecasts for efficient resource allocation and waste reduction.
- Marketing Teams: Require sales insights for effective campaigns.
- Store Managers: Benefit from performance evaluations for local adaptations.
    -

By harnessing ML techniques and historical data, the project addresses diverse stakeholder needs, ultimately enhancing overall business operations.

- Accurate Predictions: ML models provide precise forecasts for informed decision-making.
- Optimized Operations: Accurate predictions optimize various aspects of operations.
- Efficient Planning: ML models assist in efficient resource planning.
- Enhanced Strategies: ML insights lead to improved strategies.
- Tailored Approaches: ML models allow for customized approaches.
- Data-Driven Decisions: ML guides data-driven decision-making.

3. Data Understanding

The datasets provided of an American retailer having 10 stores across 3 different states: California (CA), Texas (TX) and Wisconsin (WI).

Categories of items sold: hobbies, foods and household.

There are 5 sets of datasets provided:

Training set

It consists of columns: item_id, dept_id, cat_id, store_id, state_id and days in wide format.

```
First 5 rows:
                              id        item_id    dept_id   cat_id store_id  \
0  HOBBIES_1_001_CA_1_evaluation  HOBBIES_1_001  HOBBIES_1  HOBBIES     CA_1
1  HOBBIES_1_002_CA_1_evaluation  HOBBIES_1_002  HOBBIES_1  HOBBIES     CA_1
2  HOBBIES_1_003_CA_1_evaluation  HOBBIES_1_003  HOBBIES_1  HOBBIES     CA_1
3  HOBBIES_1_004_CA_1_evaluation  HOBBIES_1_004  HOBBIES_1  HOBBIES     CA_1
4  HOBBIES_1_005_CA_1_evaluation  HOBBIES_1_005  HOBBIES_1  HOBBIES     CA_1

  state_id  d_1  d_2  d_3  d_4  ...  d_1532  d_1533  d_1534  d_1535  d_1536  \
0       CA    0    0    0    0  ...       1       1       1       0       1
1       CA    0    0    0    0  ...       0       0       0       0       0
2       CA    0    0    0    0  ...       0       0       1       0       0
3       CA    0    0    0    0  ...       8       2       0       8       2
4       CA    0    0    0    0  ...       2       0       1       3       2

   d_1537  d_1538  d_1539  d_1540  d_1541
0       0       1       0       0       1
1       0       0       0       1       0
2       0       0       0       0       0
```

Fig1. Training set

Calendar

It consists of columns: date, wm_yr_wk and d.

```
First 5 rows:
         date  wm_yr_wk    d
0  2011-01-29     11101  d_1
1  2011-01-30     11101  d_2
2  2011-01-31     11101  d_3
3  2011-02-01     11101  d_4
4  2011-02-02     11101  d_5
```

Fig2. Calendar dataset

Price per item

It consists of columns: store_id, item_id, wm_yr_wk and sell_price.

```
First 5 rows:
  store_id        item_id  wm_yr_wk  sell_price
0     CA_1  HOBBIES_1_001     11325        9.58
1     CA_1  HOBBIES_1_001     11326        9.58
2     CA_1  HOBBIES_1_001     11327        8.26
3     CA_1  HOBBIES_1_001     11328        8.26
4     CA_1  HOBBIES_1_001     11329        8.26
```

Fig3. item price dataset

Events set

It consists of columns: date, event_name, and event_type.

```
First 5 rows:
         date     event_name event_type
0  2011-02-06      SuperBowl   Sporting
1  2011-02-14  ValentinesDay   Cultural
2  2011-02-21  PresidentsDay   National
3  2011-03-09      LentStart  Religious
4  2011-03-16      LentWeek2  Religious
```

Fig4. Events set

Note: Test set includes only days in wide format.

After merging the datasets:

```
Info:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 47107050 entries, 0 to 47107049
Data columns (total 11 columns):
 #   Column      Dtype
---  ------      -----
 0   store_id    object
 1   cat_id      object
 2   item_id     object
 3   Sales       int64
 4   date        datetime64[ns]
 5   wm_yr_wk    int64
 6   sell_price  float64
 7   revenue     float64
 8   event_name  object
 9   event_type  object
 10  event       object
dtypes: datetime64[ns](1), float64(2), int64(2), object(6)
memory usage: 4.2+ GB
None

Shape:
(47107050, 11)
```

Fig5. Final merged dataset

Note -The resulting dataset which does not include test data has the Year: 2011 to 2015

```
1540 days
4.219178082191781 years
```

Fig6. final merged data stats

4. Data Preparation

Data Merging Steps

Predictive Model:

1. Transforming Date Columns: I began working on four datasets (except test data)In the train set, I transposed the date columns to make them more interpretable. This involved converting date strings into datetime objects and extracting relevant date information.

2. Incorporating Date Details: A left join between the train and calendar sets was performed to include additional details about the date of item sales, such as the day and week.

3. Merging with Weekly Sell Price Data: The previously joined dataset was joined with the item_weekly_sell_price dataset, using the week as the common identifier to incorporate information about the selling price of items.

4. Calculating Revenue: To calculate revenue, I multiplied the quantity sold by the sell price. This step was essential for determining the monetary value of sales.

5. Adding Event Information:A new column labelled 'event' was added to the merged dataset. This column contains values 'Yes' and 'No' to indicate whether an event occurred on a specific date, based on the events dataset. This information was to understand how events impact sales.

```
Info:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 47107050 entries, 0 to 47107049
Data columns (total 11 columns):
 #   Column      Dtype
---  ------      -----
 0   store_id    object
 1   cat_id      object
 2   item_id     object
 3   Sales       int64
 4   date        datetime64[ns]
 5   wm_yr_wk    int64
 6   sell_price  float64
 7   revenue     float64
 8   event_name  object
 9   event_type  object
 10  event       object
dtypes: datetime64[ns](1), float64(2), int64(2), object(6)
memory usage: 4.2+ GB
None
```

Fig7. Final merged set

Forecasting Model:

1. Initially, from the five datasets: train, test, calendar, events, and item_weekly_sell_price.I began by concatenating the columns from the train and test sets.

2. In the train dataset, I transposed the date columns to provide daily information on items sold.

3. Using the day information from the calendar dataset, I performed a left join between the train and calendar sets to include details about item sales dates and corresponding weeks.

4. Next, I merged this combined dataset with the item_weekly_sold dataset, using the week as a common identifier. This merge allowed me to calculate revenue by multiplying the quantity sold by the sell price.

5. From the resulting dataset, I extracted two columns: date and revenue. Which were then grouped the data by date and summed the revenue values.

6. The final dataset consisted of 1941 rows, with daily revenue information.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1941 entries, 0 to 1940
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     1941 non-null   datetime64[ns]
 1   revenue  1941 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 30.5 KB
```

Fig8. Forecasting final dataset

Data Preprocessing Steps

Predictive Model

1. Handling Missing Values: Checked for missing values across all features and Identified that 'sell_price,' 'revenue,' 'event_name,' and 'event_type' had missing values. Since 'event_name' and 'event_type' had a high percentage of missing values (>91.5%) and were not considered for analysis. The missing values in 'sell_price' and 'revenue' were filled with 0.

2. Data Splitting: Separated the dependent feature ('revenue') and independent features into 'y' and 'X,' respectively, in a 70:30 ratio. As the dataset was huge I decided to split in this ratio.

3. Feature Engineering: Split the 'date' column into four columns: 'day,' 'month,' 'day_of_week,' and 'year.'

4. Dropped Features: Removed the below columns from the dataset:

- 'cat_id': Information was captured in the 'item_id.'
- 'Sales': Focus was on predicting revenue, not sales.
- ‘date': A datetime column cannot be used directly in modelling.
- 'wm_yr_wk': Initially used for merging tables but no longer needed.
- 'sell_price': Per-item cost information was discarded as the primary interest was in revenue.
- 'event': Many dates had no events, making the column less useful for modelling.
- 'day': Removed due to a high number of values; 'month,' 'day_of_week,' and 'year' were used instead.

5. Utilized a Pipeline for Transformations:

- Item_id Transformation: Applied Target Encoding to the 'item_id' column (3049 categories) to capture relationships with the target variable.
- Categorical Column Transformation: Used One-Hot Encoding for 'store_id' and 'day_of_week' to convert object-type features into numeric format.
- Numerical Column Transformation: Scaled all numerical columns using Min-Max Scaling to ensure consistent data range for modelling.

Forecasting Model

1. Data Splitting: The 'date' column was set as the index, a common practice in time series forecasting. The dataset was divided into two parts: 'df_train_data' (rows 1 to 1541) and 'df_test_data' (rows 1542 to 1941), following the original training and testing sets.

2. Stationarity Assessment: The Dickey-Fuller test was applied to the 'revenue' dataset to assess its stationarity.mInitially, the dataset was found to be non-stationary, with a p-value exceeding the standard significance level of 0.05.

3. Data Transformation: To address non-stationarity, a first-order difference (revenue first shift) was computed, but it did not make the data stationary. A weekly first shift (shifting data by 7 periods) was done, resulting in a p-value of 0.00 and stationarity data but due to the presence of trends, a weekly second shift (shifting data by 14 periods) was applied, successfully rendering the dataset stationary with fewer trends.

Data Visualization:

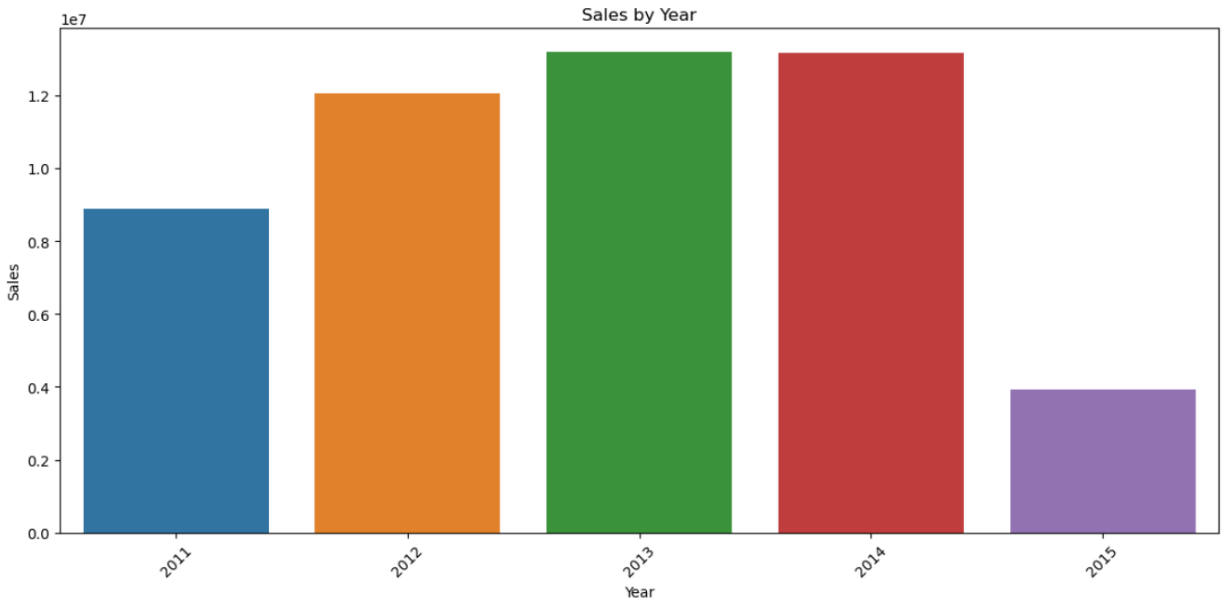

Fig9. Yearly sale of items

5. Modeling

Predictive Models

In the context of modelling revenue prediction, three machine learning algorithms were utilized:

Model 1: Decision Tree Regressor

Rationale: Decision tree regressors are chosen because they can capture non-linear relationships within the data. They are also interpretable, allowing for easy visualization and understanding of feature importance.

Model 2: XGBoost Regressor

Rationale: XGBoost is selected due to its reputation for high predictive accuracy and speed. It can effectively capture complex relationships between features and target variables. The use of regularization helps prevent overfitting, which is essential for robust predictions.

Model: 3. LightGBM (LGBM) Regressor

Rationale: LightGBM is chosen for its efficiency and speed, making it suitable for large datasets. It offers high predictive performance and is particularly effective when dealing with complex, high-dimensional data.

I have utilized a pipeline to execute the following sequence of actions which is used in all predictive models:

1. Item_id Transformation: The `item_id` column (having 3049 categories) has undergone a transformation using Target Encoding.

2. Categorical Column Transformation: The columns 'store_id,' and 'day_of_week' have been encoded using One-Hot Encoding.

3. Numerical Column Transformation: All the numerical columns have been processed using Min-Max Scaling.

independent feature (store_id, item_id, month, day_of_week, year)

target feature(revenue).

a. Approach 1

- The first model used in revenue prediction is the Decision Tree Regressor. Decision tree regressors work by recursively splitting the data into subsets based on the most significant attribute at each node, forming a tree-like structure. Here are the specific details of this model:
- The model aimed to capture non-linear relationships and provide interpretable results.
-

b. Approach 2

- The XGBoost Regressor was trained on the training data. It sequentially built multiple decision trees, with each tree correcting the errors of the previous ones.
- It incorporates regularization techniques and ensemble learning to provide accurate predictions while mitigating overfitting.

c. Approach 3

- GBM is a gradient-boosting framework that excels in regression tasks. The LGBM Regressor was trained to capture complex relationships between the independent and target features. It is known for its high predictive performance and efficiency in handling large datasets. Number of Trees (n_estimators): The number of boosting rounds or trees to build during training.

Forecasting Model

The SARIMA (Seasonal Autoregressive Integrated Moving Average) model was chosen for modelling because it effectively addresses the seasonality present in the dataset. SARIMA is an extension of the ARIMA (Autoregressive Integrated Moving Average) model, specifically designed to handle time series data with seasonal patterns. It includes

1. Seasonal Autoregressive (SAR) terms: These capture the relationship between the current value and previous values within each seasonal cycle.

2. Seasonal Integrated (I) term: This represents the differencing required to make the data stationary across seasonal cycles.

3. Seasonal Moving Average (SMA) terms: These account for the correlation between the current value and past forecast errors within each seasonal cycle.

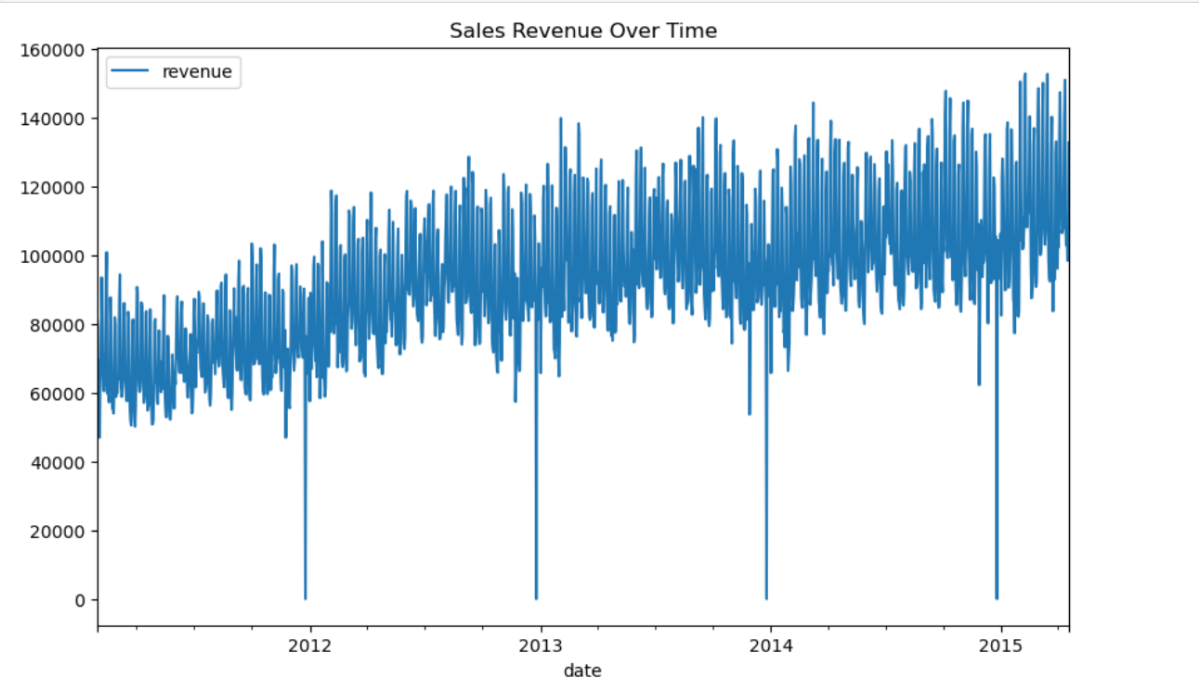

Fig10. Non stationary dataset showing sales revenue over time

Overall, the hyperparameters, such as p, d, q, P, D, Q, and the seasonal period, are essential for configuring the model to fit the characteristics of the dataset best and achieve accurate sales revenue forecasts.

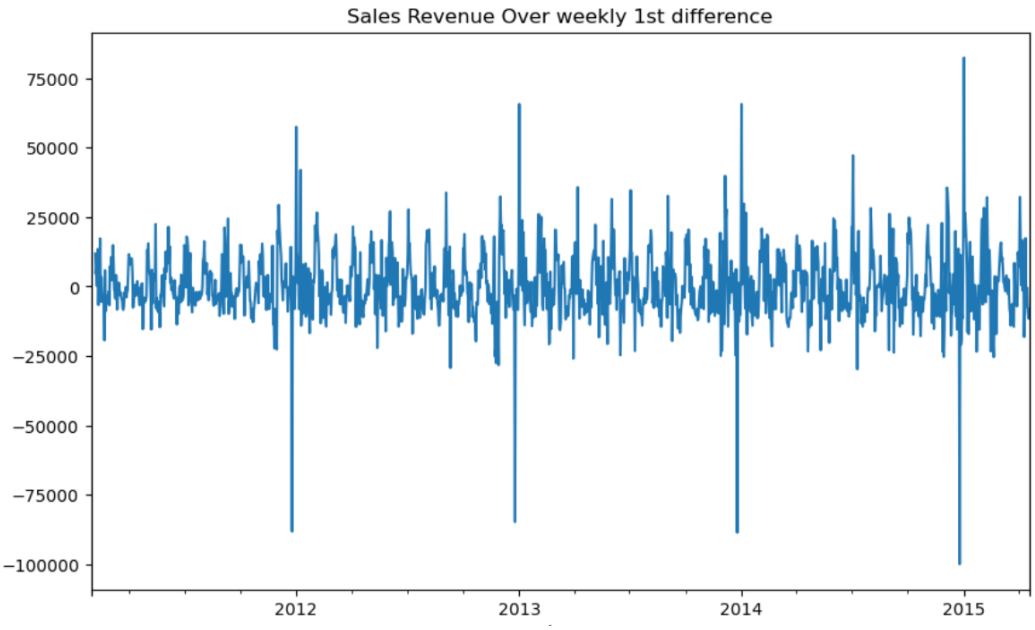

Fig11. Stationary dataset showing sales revenue over time

The stationary data df_train_data[‘ weekly first difference’] has been used for modelling.

Based on the ACF ,PACF plots and hyperparameterization I have choosed the below hyperparameters:

order=(5, 3, 2), seasonal_order=(1, 1, 1, 7)

Weekly first difference

In [38]:
```
acf7 = plot_acf(df_train_data["weekly first difference"].dropna())
pacf7 = plot_pacf(df_train_data["weekly first difference"].dropna())
```

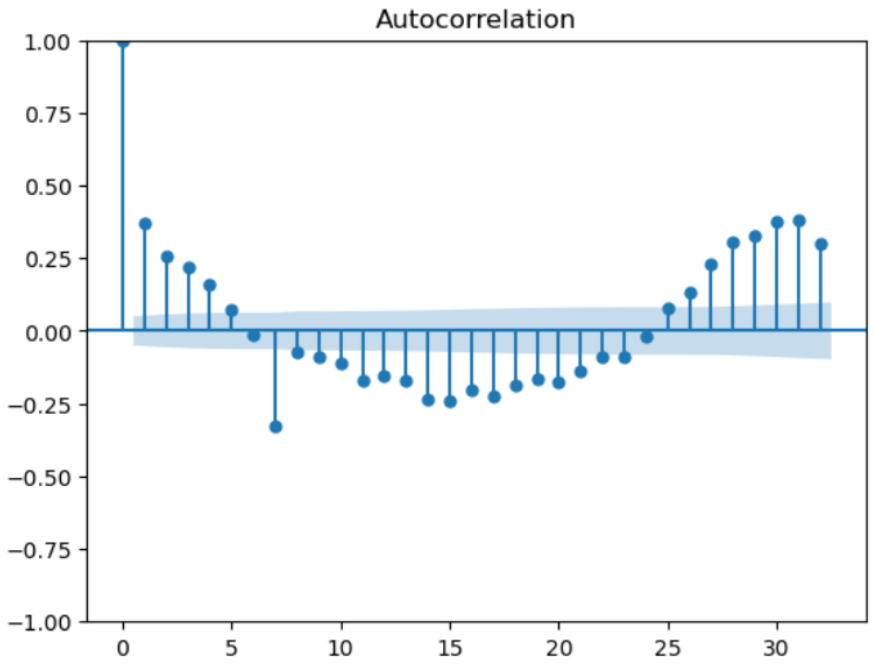

Fig12. ACF plot on 'weekly first difference'

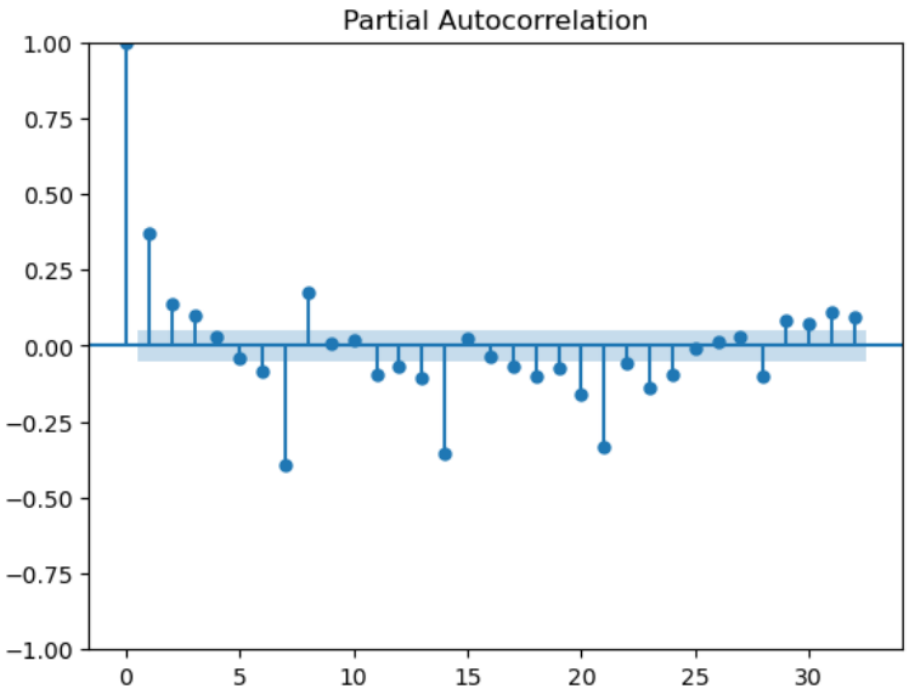

Fig13. PACF plot on 'weekly first difference'

6. Evaluation

a. Evaluation Metrics

- RMSE (Root Mean Squared Error) is a preferred metric in retail regression models due to its interpretability and sensitivity to large errors. It penalizes large errors more heavily, which is crucial in retail, where inaccuracies can be costly. It simplifies model comparison and aligns with mathematical principles, making it a practical choice.

- Relation to Project Goals: RMSE aligns with the project goal of accurately forecasting sales revenue. It measures the average magnitude of errors in predicting revenue values, helping to assess the overall model performance. Minimizing RMSE is directly related to improving the accuracy of revenue forecasts, which is a critical objective in retail.

b. Results and Analysis

Prediction Models

S No.	Regression Model	Train set RMSE	Validation Set RMSE
1.	Decision Tree Regressor	6.99	7.03
2.	XGBoost Regressor:	7.58	7.60
3.	LightGBM (LGBM) Regressor	7.55	7.56

Table 1

Decision Tree Regressor:

It achieved RMSE scores of 6.99 on the training set and 7.03 on the validation set. These scores indicate that the model's predictions are very close to the actual values for both datasets.

Good Generalization: The close proximity of the training and validation RMSE scores suggests that the model generalizes well to unseen data.

XGBoost Regressor:

The RMSE scores of 7.58 on the training set and 7.60 on the validation set. These scores indicate a reasonable performance with a good fit to the data.

Generalization: The similarity in RMSE scores between the training and validation sets suggests that the XGBoost model generalizes well.

LightGBM Regressor:

It achieved RMSE scores of 7.55 on the training set and 7.56 on the validation set. These scores are close to each other, indicating that the model performs consistently on both datasets.

Hyperparameter: The LGBM Regressor was trained with 100 estimators.

Effective Model: The LGBM model demonstrates good predictive capabilities, with both training and validation RMSE scores being relatively low.

The Best Model:

Overall, all three models exhibit similar performance, with RMSE scores in the same range. Thus, other factors which include model interpretability, training time, and resource requirements are considered in choosing the best model for deployment. LightGBM outperforms the other two models.

Forecasting Model

SARIMA model:

After careful selection of Hyperparameters on the basis of acf and pacf plots. The RMSE score of 18,985 was obtained.

Analysis of the RMSE score:

A high RMSE score can partly be attributed to the complexity of the sales revenue data. SARIMA models are effective at capturing seasonal patterns, but if the data exhibits intricate or irregular patterns that do not align well with the SARIMA framework, the model's performance may suffer. There is scope for choosing better hyperparameters to improve te model performance.

```
model_ARIMA_fit.aic
```

```
33071.89495608513
```

```
model_SARIMA_fit.aic
```

```
32800.635801409895
```

Akaike information criterion (aic score) is used to compare models. Lower the score better is the model. From the image, it is clear that SARIMA performs better than ARIMA model. Thus I have chosen SARIMA model for forecasting as it captures seasonality better than ARIMA.

c. Business Impact and Benefits

The final models, LightGBM and SARIMA, have a significant impact and offer several benefits to the business use cases identified.

LightGBM Model

- LightGBM excels at capturing complex relationships in sales data, leading to more accurate short-term sales forecasts. This accuracy is crucial for effective inventory management and staffing.
- The accuracy in predictions would help retailers maintain the right product quantities, reducing overstock and understock issues. This optimization minimizes carrying costs and product wastage.
- The Retailers can allocate resources efficiently based on sales predictions, leading to cost savings and improved service quality.
- Precise forecasts enable dynamic pricing adjustments, maximizing revenue and profit margins.

SARIMA Model

- Seasonal Pattern Handling: SARIMA effectively addresses seasonal sales patterns caused by holidays, events, or weather changes, crucial for retail where seasonality is common.
- Robust Long-Term Forecasting: SARIMA excels in long-term forecasting, helping retailers plan for the entire year, including peak and off-peak periods.

In conclusion, the final LightGBM and SARIMA models offer substantial benefits in terms of sales forecasting, inventory management, resource allocation, pricing strategies, and customer satisfaction. While the exact value generated may vary by business, these models have the

potential to result in significant cost savings and revenue increases, making them valuable assets for the retail business.

d. Data Privacy and Ethical Concerns

Data Privacy Implications in Retail Businesses

- The project involves using historical sales transaction data. While this data may not directly contain personally identifiable information, it can still reveal sensitive information about customer purchasing behaviours. Aggregating and anonymizing this data is essential to protect customer privacy.
- Business Confidentiality: The sales data may also include information about product pricing, sales strategies, and other business-sensitive information. Unauthorized access to this data could harm the business's competitive advantage.

Ethical Concerns

- Data Usage Transparency: Ethical concerns arise when data is used without the explicit consent or knowledge of the individuals whose data is being analyzed. Transparency in data usage and informing customers about how their data is used is crucial.

- Bias and Fairness: Machine learning models can inadvertently perpetuate biases present in historical data. This can lead to unfair or discriminatory outcomes, such as biased pricing or promotions. It is essential to monitor and mitigate bias in model predictions.

Steps to Ensure Data Privacy and Ethical Considerations

- Data Anonymization: Customer-specific information, such as names or addresses, should be removed or anonymized from the dataset to protect individual privacy.
- Data Security: Implement robust data security measures to prevent unauthorized access to sensitive data. Encryption and access controls should be in place.
- Consent and Transparency: Ensure that customers are informed about how their data is used and have the option to opt out. Transparent privacy policies should be in place and easily accessible.
- Bias Mitigation: Regularly assess model predictions for bias and fairness. Adjust the model if it perpetuates bias, and strive for fairness in pricing and promotions.
- Compliance: Adhere to relevant data protection regulations, such as GDPR or CCPA, depending on the geographical scope of the business. Ensure that data usage and storage practices are compliant.
- Regular Audits: Conduct regular privacy and ethics audits of the project to identify and address any emerging concerns or issues.

In retail-related projects, privacy implications are addressed by conducting data privacy assessments, anonymizing customer data, obtaining consent, ensuring data security, and offering opt-out options for customers. These measures protect individual privacy and comply with data protection regulations. [2] In retail contexts, data privacy necessitates collaboration

among three main stakeholders: consumers, retailers, and regulatory bodies, each working in harmony to protect the interests of the others.[3]“Privacy by Design" in ML retail involves integrating privacy protections into the design and development of machine learning systems from the beginning, ensuring data privacy is a central consideration throughout the entire process.

7. Deployment

Steps	Description
Step 1: Model Serialization	Before deploying, the trained model needs to be serialized into a format that can be easily loaded and used in a deployment environment. The formats such as joblib or pickle can be used to save the trained model.
Step 2: Version Control with Git	The serialized model, along with any necessary code and dependencies, should be version-controlled using Git. This ensures that you have a record of changes and can collaborate with others effectively if required.

Step 3: API Implementation (FastAPI)	Use FastAPI, a modern, fast, web framework for building APIs with Python, to create an API that can accept input data, make predictions using the trained model, and return results.
Step 4: API Documentation	Document your API endpoints and provide clear instructions for users on how to make requests and interpret responses.
Step 5: Containerization with Docker	To package the application and its dependencies into a single container ensures consistency across different environments. It can be done by creating a `Dockerfile` that specifies the base image, sets up the environment, installs necessary libraries, and copies the code and model files into the container.
Step 6: Build and Test the Docker Image	With the use of Docker commands build an image from the Dockerfile[4] and run the container locally to ensure that everything is working as expected before deployment.
Step 7: Container Registry	Push the built Docker image to a container registry like Docker Hub or a cloud-based registry like Amazon ECR or Google Container Registry. This makes the image accessible for deployment.

Step 8: Deployment Platform (Heroku)	Heroku is a Platform-as-a-Service (PaaS) that simplifies application deployment. It supports container deployments, making it suitable for deploying Docker containers. Create a Heroku app and link it to the container registry. (On Heruko Student account needs to be applied for and the approval of which takes about 3 to 4 days.)

Recommendations:

- Use infrastructure-as-code (IaC) tools like Terraform or AWS CloudFormation for reproducible deployments.
- Implement CI/CD pipelines to automate the deployment process. Tools like GitHub Actions or GitLab CI/CD can trigger deployments when changes are pushed to the repository.

8. Conclusion

Working with large datasets can be memory-intensive, so optimization is crucial to maintain performance. The LGBBoost regressor is known for its efficiency and accuracy with large datasets, while SARIMA is effective at capturing seasonality. Both models are valuable choices when dealing with substantial data. Retail businesses by making data-driven decisions can effectively increase their profits by maintaining high-demand products and being prepared for the high in-demand products. The process of model deployment is sequential and involves a series of steps.

Future Work and Recommendations:

1. Hyperparameter Tuning: Further hyperparameter tuning for the models can potentially reduce the RMSE and improve forecasting accuracy.

2. Real-Time Updates: Implement real-time updates for the models to adapt to changing market conditions and improve forecasting accuracy.

3. Advanced Deployment: Consider deploying the models on cloud platforms like AWS, Azure, or Google Cloud for enhanced scalability and performance.

9. References

[1] Getting your data in and out of vaex. (n.d.). Retrieved October 10, 2023, from https://vaex.io/docs/getting_data_in_vaex.html

[2] Martin, K. D., Kim, J. J., Palmatier, R. W., Steinhoff, L., Stewart, D. W., Walker, B. A., Wang, Y., & Weaven, S. K. (2020). Data privacy in retail. Journal of Retailing, 96(4), 474–489. https://doi.org/10.1016/j.jretai.2020.08.003

[3] Numbers tell the story behind privacy by design for retailers and other enterprises [infographic]—Boomi. (2020, July 15). Boomi Resources. https://live-boomi-resources.pantheonsite.io/numbers-tell-the-story-behind-privacy-by-design-for-retailers-and-other-enterprises-infographic/

[4] Banerjee, A. (2022, September 3). [tutorial]: Serve a containerised ml model using fastapi and docker. *Medium*. https://medium.com/@ashmi_banerjee/tutorial-serve-a-containerised-ml-model-using-fastapi-and-docker-eaa0d9b05743

GitHub link: https://github.com/bhutanisaumya/Adv_MLA_AT2

Appendix

- Screenshots of Docker

  Commands used:

  - cd C:/Users/saumy/OneDrive/Desktop/Semester-3/AdvMLA/AdvMLA_AT2
  - docker run -d -p 80:80 docker/getting-started
  - docker build -t retail_ml-fastapi:latest .
  - docker run -dit --rm --name adv_mla_at2_retail_ml_fastapi -p 8080:80 retail_ml-fastapi:latest
  - cd app
  - python -m uvicorn main:app --host 0.0.0.0 --port 80
  - docker stop adv_mla_at2_retail_ml_fastapi

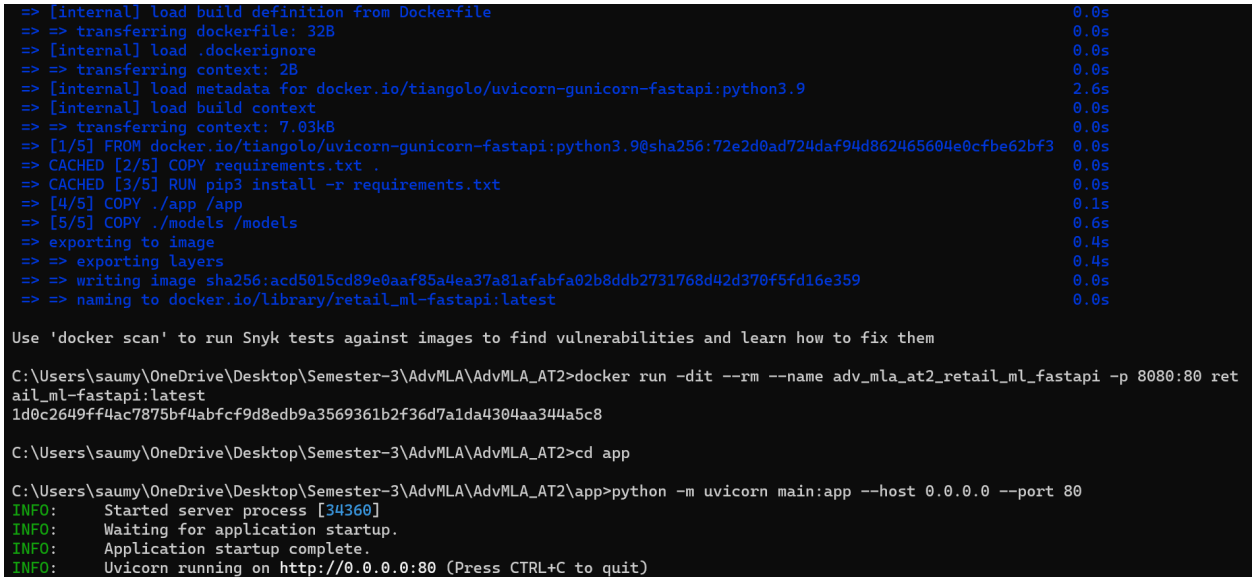

Fig.(a)

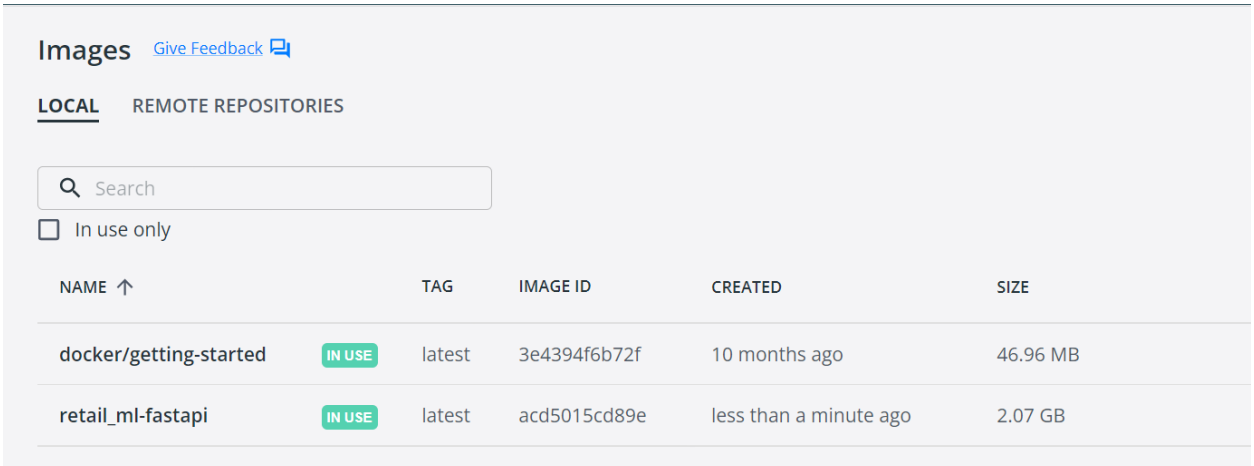
Images
Give Feedback
LOCAL
REMOTE REPOSITORIES
Search
In use only
NAME
TAG
IMAGE ID
CREATED
SIZE
docker/getting-started
IN USE
latest
3e4394f6b72f
10 months ago
46.96 MB
retail_ml-fastapi
IN USE
latest
acd5015cd89e
less than a minute ago
2.07 GB

Fig.(b)

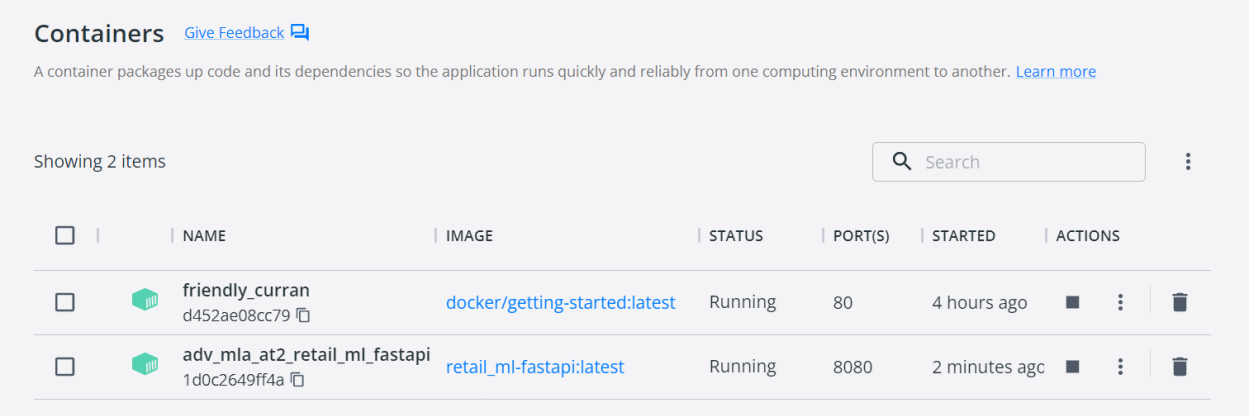
Containers
Give Feedback
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more
Showing 2 items
Search
NAME
IMAGE
STATUS
PORT(S)
STARTED
ACTIONS
friendly_curran
d452ae08cc79
docker/getting-started:latest
Running
80
4 hours ago
adv_mla_at2_retail_ml_fastapi
1d0c2649ff4a
retail_ml-fastapi:latest
Running
8080
2 minutes ago

Fig.(c)

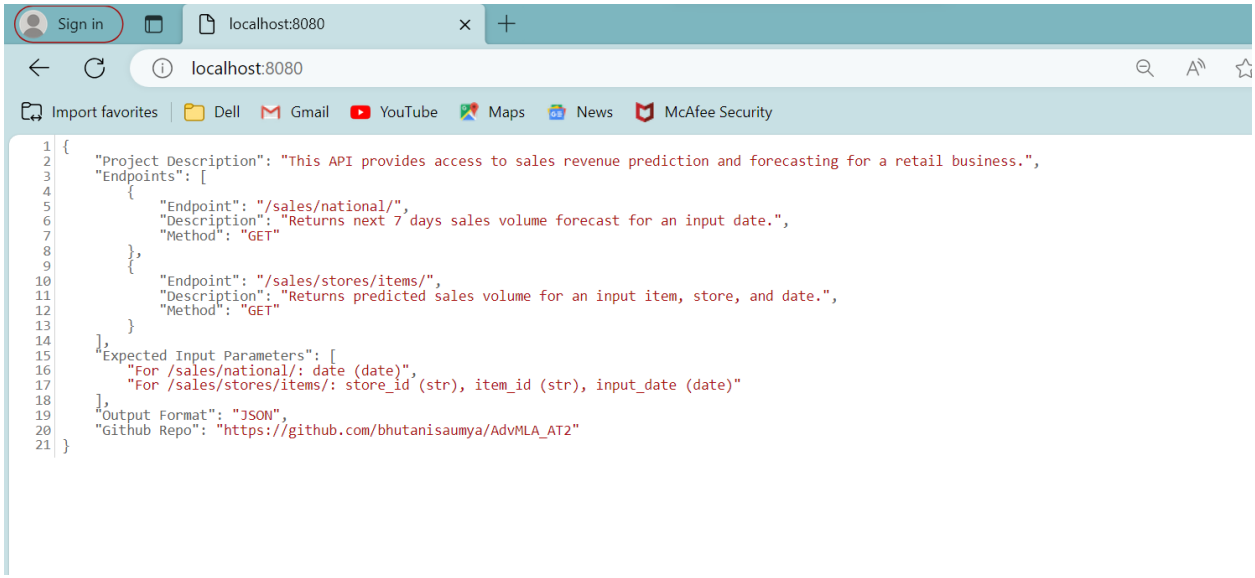

Fig.(d)

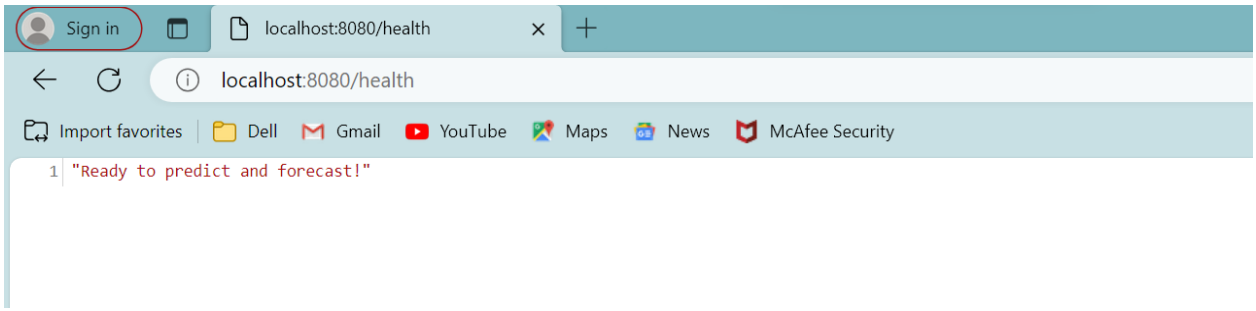

Fig.(e)

- Issues Faced:

  I could not use the events data in my modelling process as the dataset was huge and I did face memory issues while working on it.

  Note: No major Hyperparametertunning was done for these predictive ML models as the dataset was huge. I tried to perform Hyperparametertunning but the model would take a lot of time and end up showing memory error. In order to reduce the size I imported

the vaex and converted the dataset to HDF5 file format but that also crashed with Memory error.

```
In [56]: from vaex.ml.sklearn import IncrementalPredictor
         from sklearn.ensemble import RandomForestRegressor

         # Create the RandomForestRegressor
         rf_reg = RandomForestRegressor(n_estimators=50, random_state=42)

         # Convert the NumPy arrays to lists
         X_train_scaled_list = X_train_scaled.tolist()
         y_train_list = y_train.tolist()

         # Initialize IncrementalPredictor with features, target, and model
         vaex_model = IncrementalPredictor(features=X_train_scaled_list, target=y_train_list, model=rf_reg, batch_size=500000)

         # Fit the model
         vaex_model.fit(progress='widget')

         ---------------------------------------------------------------------------
         MemoryError                               Traceback (most recent call last)
```

Fig. Memory error

I tried to push it in GitHub it did not work due to the large size of SARIMA_model.joblib file. Even after compressing it while dumping it. It exceeded the allowed limit. (earlier it was 136Mb after compressing 76Mb). I have updated in my zip folder but git repo does it contain it. After deleting the reference to the previous large file of joblib when I tried to push some of the files did not merge (as I uploaded manually before) later I created a new repo and pushed all my updates (https://github.com/bhutanisaumya/Adv_MLA_AT2) However, it does not contain the Sarima_model.joblib file.Thus, please consider my zipped folder for evaluation.

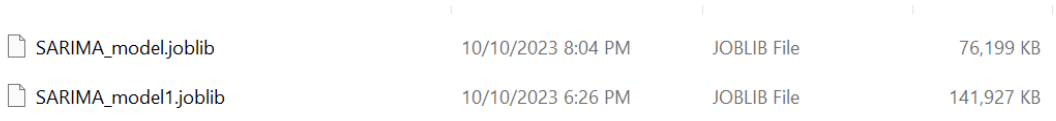

Fig. file size

```
Writing objects: 100% (6/6), 71.03 MiB | 2.32 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote: warning: See https://gh.io/lfs for more information.
remote: warning: File models/forecasting/SARIMA_model.joblib is 74.41 MB; this is larger than GitHub's recommended maximum file size
of 50.00 MB
remote: warning: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To https://github.com/bhutanisaumya/Adv_MLA_AT2.git
   2681bc4..efb532c  master -> master
```

Fig. Upload error

I encountered difficulties when running the "sales/national" and "sales/stores/items" endpoints on Docker. These issues arose from discrepancies in Python and scikit-learn (sklearn) versions used for model training and serialization with joblib, as well as the versions available within the Docker environment. Despite my attempts to resolve the problem by reinstalling matching versions consistent with my Jupyter notebook environment, Docker persisted in referencing an incompatible local version.

Consequently, although the endpoints functioned correctly locally, I was unable to run them on Docker due to the presence of mismatched joblib files and conflicting Python and scikit-learn versions.

The process of model deployment did not work for me as it was showing memory issues.

API Structure Instructions:

1. Predictive model: /sales/national/ Endpoint
    - This endpoint takes an input date in the format 'YYYY-MM-DD' as a parameter.
    - It converts the input date to a datetime object for processing.
    - It generates a forecast for the next 7 consecutive days starting from the input date using a SARIMA model.
    - For each of the 7 days, it creates a dictionary with the date and the forecasted sales value and appends it to a list.
    - Finally, it returns the list of forecasted sales for 7 consecutive days in JSON format.

2. Forecasting model: /sales/stores/items Endpoint:
    - This endpoint takes three parameters: store_id (string), item_id (string), and date (in 'YYYY-MM-DD' format).
    - It converts the input date to a datetime object.
    - It calculates additional features such as month, day of the week, and year from the date.
    - These features, along with store_id and item_id, are used as input to a LightGBM model (lgbm_pipe) to predict sales.
    - The predicted sales value is returned as part of a JSON response, including the store_id, item_id, input date, and prediction.
    - Additionally, the /health and / endpoints provide information about the status of the API and a brief description of the project, respectively. The information is returned as JSON responses.

    This FastAPI application is designed to provide sales prediction and forecasting services for a retail business, making it accessible through the specified endpoints and input parameters.