Predicting Shipping Time for Supply Chain Dataset using Regression Supervised Learning

Hellen Gaitan Camargo
Southern Alberta Institute of Technology
Data Science – DATA-050-003

Table of Contents

1. Business Understanding

1.1. Business objective

The primary business objective is to develop and compare various machine learning models to accurately predict future shipping costs. This involves splitting the dataset into training and testing sets, applying different algorithms, and evaluating their performance to determine the best predictive model.

1.2. Problem Statement

Here in this project, it will be developing a predictive model to accurately forecast shipping costs based on historical shipping data. This involves analyzing various factors such as shipment origin, destination, weight, mode, and additional charges to determine their impact on the overall shipping cost. The goal is to create a robust machine-learning model that can predict future shipping costs with high accuracy, enabling the company to optimize its logistics operations and cost management strategies.

1.3. Why is it worth solving

Operational Optimization: With a predictive model, the company can identify inefficiencies in its current logistics processes. This enables the company to streamline operations, reduce delays, and improve overall service quality.

Strategic Planning: Predictive insights into shipping costs can inform long-term strategic planning. The company can better forecast future expenses, allocate resources more effectively, and plan for expansion into new markets with a clearer understanding of potential costs.

2. Data Collection and Understanding

The data was retrieved From Kaggle. Shipping Optimization Challenge which is originally shared to help a logistics firm choose the best way to ship goods.

1.4. Data Description

The dataset contains 5113 rows and 13 **columns**. The dataset contains the following features, which are crucial for developing and evaluating machine learning models to predict future shipping costs:

Here is the detailed description of each column in the dataset:

1. **Shipment ID**
    - **Description**: Unique identifier for each shipment.
    - **Type**: Categorical (string).
    - **Role**: Identifier for individual shipments, not used as a feature.
2. **Send timestamp**
    - **Description**: Timestamp indicating when the shipment was sent.
    - **Type**: Categorical (datetime).
    - **Role**: Feature to analyze the impact of time on shipping performance.
3. **Pick up point**
    - **Description**: Code for the pick-up location.

- **Type**: Categorical (string).
- **Role**: Feature to analyze the impact of pick-up locations on shipping costs.

4. **Drop off point**
   - **Description**: Code for the drop-off location.
   - **Type**: Categorical (string).
   - **Role**: Feature to analyze the impact of drop-off locations on shipping costs.
5. **Source country**
   - **Description**: An encoded value representing the source country of the shipment.
   - **Type**: Categorical (string).
   - **Role**: Feature to analyze the impact of origin on shipping costs.
6. **Destination country**
   - **Description**: An encoded value representing the destination country of the shipment.
   - **Type**: Categorical (string).
   - **Role**: Feature to analyze the impact of destination on shipping costs.
7. **Freight cost**
   - **Description**: Cost of freight for the shipment.
   - **Type**: Numeric (float).
   - **Role**: Target variable for predicting shipping costs.
8. **Gross weight**
   - **Description**: Gross weight of the shipment in kilograms.
   - **Type**: Numeric (float).
   - **Role**: Feature to analyze the impact of weight on shipping costs.
9. **Shipment charges**
   - **Description**: Additional charges for the shipment.
   - **Type**: Numeric (float).
   - **Role**: Feature to analyze the impact of additional charges on overall shipping costs.
10. **Shipment mode**
    - **Description**: Mode of shipment (e.g., Air, Sea).
    - **Type**: Categorical (string).
    - **Role**: Feature to analyze the impact of shipment mode on shipping costs.
11. **Shipping company**
    - **Description**: Company responsible for shipping.
    - **Type**: Categorical (string).
    - **Role**: Feature to analyze the impact of the shipping company on shipping performance.
12. **Selected**
    - **Description**: Indicator for whether the shipment was selected for a specific criterion.
    - **Type**: Categorical (string).
      Role: Feature to analyze the selection criteria's impact on shipping.
13. **Shipping time**
    - **Description**: Time that takes to ship the product.
    - **Type**: Numeric(float)

- **Role**: This is the target variable.

3. Data Preparation

To ensure the dataset was suitable for building an accurate predictive model, several steps were taken, including exploratory data analysis, data wrangling, and feature engineering. These processes helped in understanding the dataset, cleaning it, and transforming it into a format optimal for effective model building. The data analysis was mainly done in Python while the other steps were handled in mlOS.

3.1. Exploratory Data Analysis

Preliminary data exploration was conducted to understand the dataset's content and structure. The following steps were undertaken:

- **Checking for Null Values:**
  The dataset was complete and did not contain any missing or null values. This means that every entry within the dataset had valid data for all columns.

```
df.isnull().sum()

shipment_id            0
send_timestamp         0
pick_up_point          0
drop_off_point         0
source_country         0
destination_country    0
freight_cost           0
gross_weight           0
shipment_charges       0
shipment_mode          0
shipping_company       0
selected               0
shipping_time          0
dtype: int64
```

Figure 1: Checking for Null Values

- **Data Type Verification**:
  The dataset consists of five columns, with four of these being numeric. Additionally, there is one column representing timestamps, which records the date and time when a product was sent. Although this timestamp column is formatted as an object type, it will not be converted to a date type, as it is not relevant to the model's inputs.
  The remaining data in the dataset is of object type and represents categorical features. The relevant categorical features will be used in the model after encoding, as they offer qualitative insights that are valuable for predictive analysis.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5114 entries, 0 to 5113
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   shipment_id          5114 non-null   object
 1   send_timestamp       5114 non-null   object
 2   pick_up_point        5114 non-null   object
 3   drop_off_point       5114 non-null   object
 4   source_country       5114 non-null   object
 5   destination_country  5114 non-null   object
 6   freight_cost         5114 non-null   float64
 7   gross_weight         5114 non-null   float64
 8   shipment_charges     5114 non-null   float64
 9   shipment_mode        5114 non-null   object
 10  shipping_company     5114 non-null   object
 11  selected             5114 non-null   object
 12  shipping_time        5114 non-null   float64
dtypes: float64(4), object(9)
memory usage: 519.5+ KB
```

Figure 2: Data Type Verification

- **Statistical Information**:
  Analyzing the statistics of the target variable reveals the following key points: The mean value is 12.64 days, indicating the average duration represented by the target variable. The 50th percentile, shows that 50% of the values fall within a range of approximately 5.5 days. The minimum value in the dataset is 5 days, while the maximum extends to 57 days. This spread suggests a considerable variation in the target variable, with most values clustered around the lower end of the range but with some observations extending significantly longer.

```
df.describe()
```

	freight_cost	gross_weight	shipment_charges	shipping_time
count	5114.000000	5114.000000	5114.000000	5114.000000
mean	91.200923	954.074099	0.871732	12.641822
std	5.154340	1266.102859	0.127972	10.273164
min	82.885600	5.000000	0.562500	5.000000
25%	87.750000	225.000000	0.750000	5.199910
50%	90.750000	505.000000	0.900000	5.405150
75%	92.837500	1100.000000	0.900000	19.644270
max	115.620000	10000.000000	1.125000	57.249650

Figure 3: Statistical Information

- **Correlation Analysis**:
  The correlation matrix below shows a strong positive correlation between shipping time and shipment mode, suggesting that different shipment modes have a significant impact on the duration of shipping. In contrast, there is a strong negative correlation between shipping time and shipping company. This implies that variations in the shipping company are inversely related to shipping time, indicating that certain companies are associated with shorter shipping durations. The remaining features exhibit only weak correlations with shipping time. Destination Country has a correlation of -0.11 and Freight Cost shows a correlation of -0.09, also reflecting a minimal negative impact. Gross Weight and Shipment Charges have correlations of 0.08 and 0.09, respectively, suggesting very weak positive relationships.

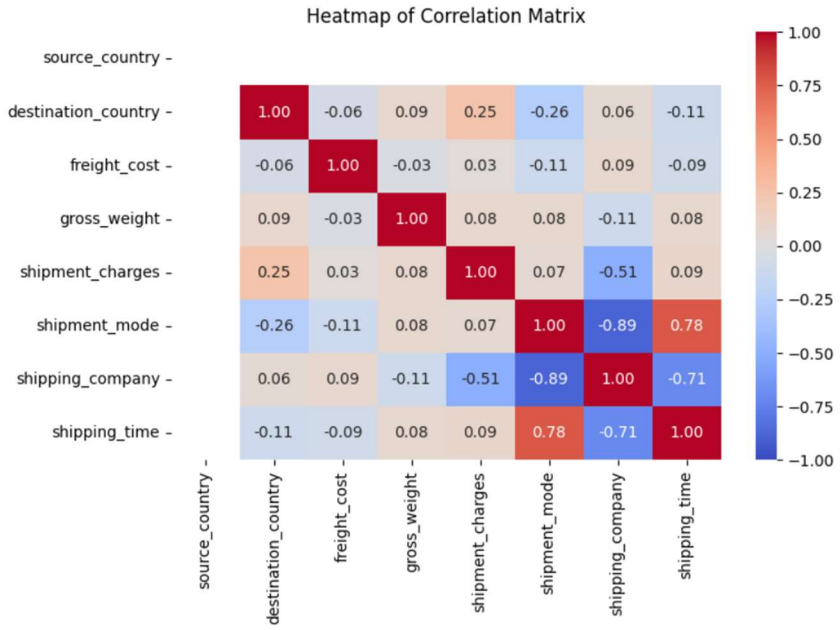

Figure 4: Correlation Analysis

- **Univariant Analysis – Shipping Time Distribution**:
  The histogram for the shipping time variable reveals a highly right-skewed distribution. As the shipping time increases, the frequency of occurrences decreases significantly, resulting in a long tail extending towards the higher end of the range. This right skewness indicates that while most shipments are completed relatively quickly, there are a few instances of much longer shipping times.

```
# Create a histogram for the 'shipping_time' column
plt.figure(figsize=(8, 4))
plt.hist(df['shipping_time'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Shipping Time')
plt.xlabel('Shipping Time')
plt.ylabel('Frequency')
plt.show()
```

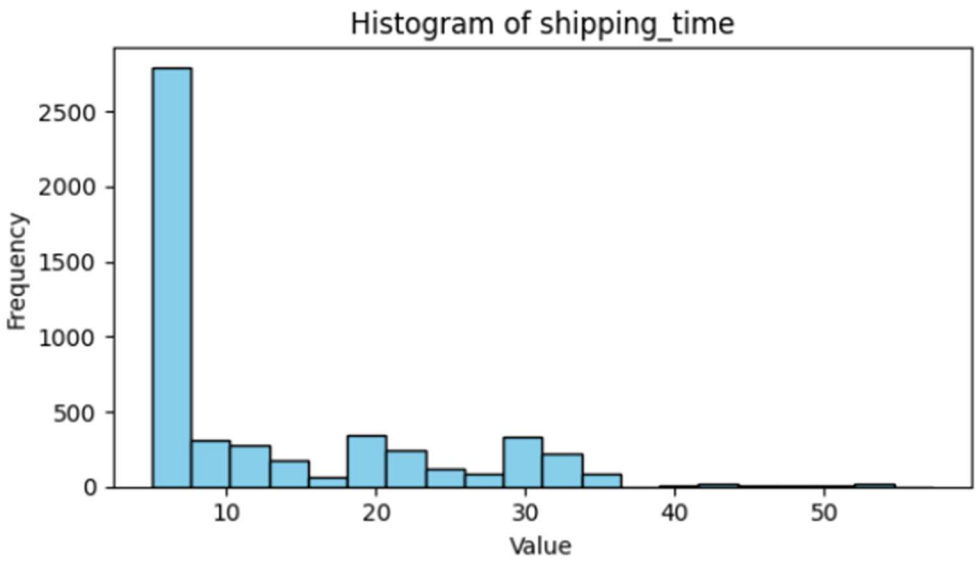

Figure 5: Univariant Analysis – Shipping Time Distribution

- **Outlier Detection:**
  Box plots were created to illustrate the presence of outliers in the numerical features of the dataset. The resulting figure shows outliers in freight cost, gross weight, and shipping times. Outliers, which are values significantly different from the majority of the data, can negatively impact machine learning models by distorting statistical analyses and potentially leading to decreased model performance. Identifying and addressing these outliers is essential for improving the accuracy and reliability of the models.

```
# Set up the matplotlib figure with 4 subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 7))

# Plot each box plot including tittle and y labels
sns.boxplot(ax=axes[0, 0], y=df['freight_cost'])
axes[0, 0].set_title('Box Plot of Freight Cost')
axes[0, 0].set_ylabel('Freight Cost')

sns.boxplot(ax=axes[0, 1], y=df['gross_weight'])
axes[0, 1].set_title('Box Plot of Gross Weight')
axes[0, 1].set_ylabel('Gross Weight')

sns.boxplot(ax=axes[1, 0], y=df['shipment_charges'])
axes[1, 0].set_title('Box Plot of Shipment Charges')
axes[1, 0].set_ylabel('Shipment Charges')

sns.boxplot(ax=axes[1, 1], y=df['shipping_time'])
axes[1, 1].set_title('Box Plot of Shipping Time')
axes[1, 1].set_ylabel('Shipping Time')

plt.show()
```

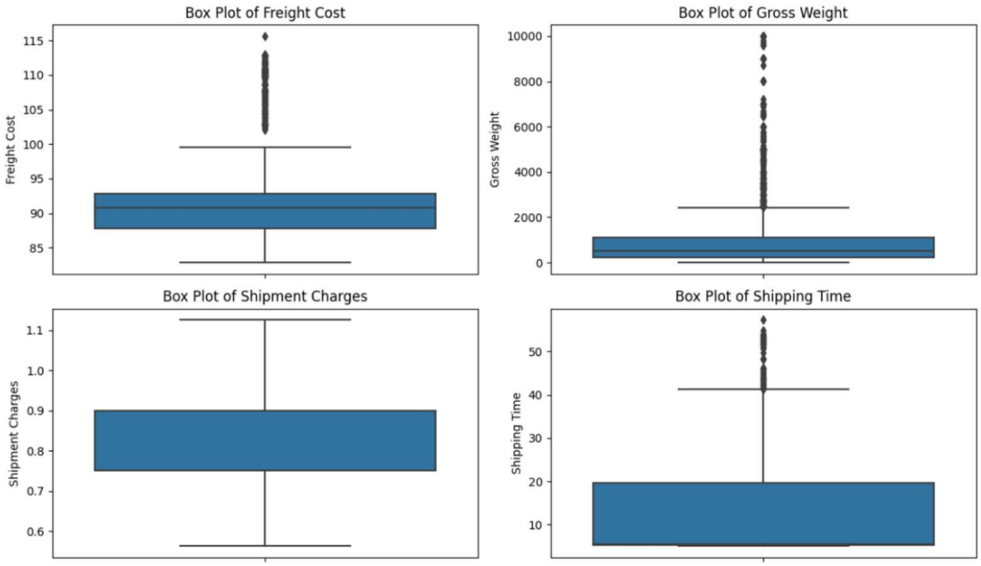

Figure 6: Outlier Detection

3.2. Data Wrangling

The dataset was prepared for modeling through the following wrangling steps:

- **Removing unnecessary columns:**
  Several columns were removed from the dataset as they were considered unnecessary for the machine learning model. The send_timestamp column was excluded because it contains unique timestamp values that do not contribute to the predictive modeling process. Similarly, shipment_id was removed as it merely serves as an identifier for the shipments and does not provide valuable information for the model.
  The pick_up_point and drop_off_point columns, which contained abbreviations for the source and destination countries, were also deleted. These columns were considered redundant since their information was already captured in the respective source and destination country features, making them duplicative and unnecessary. Lastly, the selected and source country columns, which consisted solely of unique values ("y" for selected and "GB" for source country), were removed as they offered no additional information.
  .

```
df.drop(columns=['send_timestamp','shipment_id','pick_up_point', 'drop_off_point','selected'], inplace=True)
df.head()
```

	source_country	destination_country	freight_cost	gross_weight	shipment_charges	shipment_mode	shipping_company	shipping_time
64	GB	IN	90.9403	128.00	0.9	Ocean	SC1	17.46875
70	GB	IN	90.9403	1006.00	0.9	Ocean	SC1	7.54421
76	GB	IN	90.9403	1206.00	0.9	Ocean	SC1	7.54606
80	GB	IN	90.9403	335.00	0.9	Ocean	SC1	27.57731
143	GB	IN	90.9403	115.21	0.9	Ocean	SC1	27.59583

Figure 7: Removing unnecessary columns

- **Removing Outliers:**
  The outliers identified previously were removed from the dataset. Specifically, values falling above the upper bound were filtered out, resulting in a cleaner dataset. This process helped to reduce the impact of extreme values on the model, which led to an improvement in model performance.

```python
# Function to remove outliers
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    filtered_column = column[(column >= lower_bound) & (column <= upper_bound)]
    return filtered_column

# Remove outliers for each column and store the filtered columns
df['freight_cost'] = remove_outliers(df['freight_cost'])
df['gross_weight'] = remove_outliers(df['gross_weight'])
df['shipment_charges'] = remove_outliers(df['shipment_charges'])
df['shipping_time'] = remove_outliers(df['shipping_time'])
```

Figure 8: Removing Outliers

The Figure below displays a new boxplot for outlier detection following the removal of outliers. The updated boxplot clearly indicates that the majority of the extreme values have been successfully removed, resulting in a more centralized and tighter distribution of the data. This refinement helps to improve the accuracy and reliability of subsequent analyses and model performance.

```python
# Set up the matplotlib figure
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 7))

# Plot each box plot with titles
sns.boxplot(ax=axes[0, 0], y=df['freight_cost'])
axes[0, 0].set_title('Box Plot of Freight Cost')
axes[0, 0].set_ylabel('Freight Cost')

sns.boxplot(ax=axes[0, 1], y=df['gross_weight'])
axes[0, 1].set_title('Box Plot of Gross Weight')
axes[0, 1].set_ylabel('Gross Weight')

sns.boxplot(ax=axes[1, 0], y=df['shipment_charges'])
axes[1, 0].set_title('Box Plot of Shipment Charges')
axes[1, 0].set_ylabel('Shipment Charges')

sns.boxplot(ax=axes[1, 1], y=df['shipping_time'])
axes[1, 1].set_title('Box Plot of Shipping Time')
axes[1, 1].set_ylabel('Shipping Time')

# Adjust layout
plt.tight_layout()
plt.show()
```

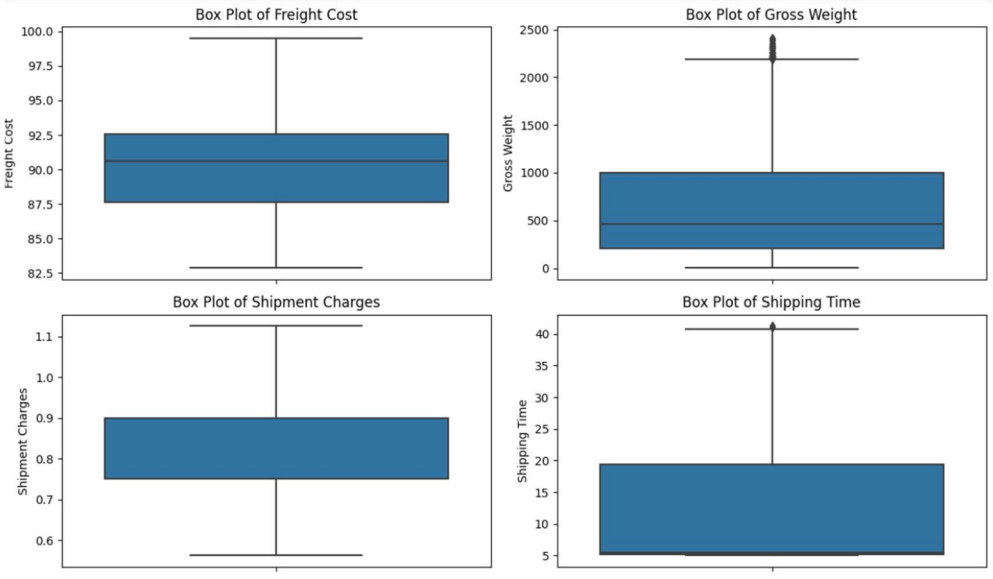

Figure 9: Boxplot After Removing Outliers

3.3. Feature Engineering

From this point onward, all steps were executed within mlOS, a low-code platform provided by Braintoy, a startup based in Calgary, Alberta. The platform offers a user-friendly environment for executing complex machine-learning tasks with minimal coding requirements. The dataset was prepared for modeling through the following feature preprocessing steps:

- **Text to Numerical Conversion:**
  The categorical values were converted to numerical formats within mlOS to facilitate data training. Specifically, the columns source country, destination country, shipment mode, and shipping company were encoded into numerical values, as illustrated in the figure below. This conversion is crucial for enabling the machine learning model to process and learn from categorical data effectively.

Figure 10: Converting Categorical to Numerical

Destination Country		Shipment Mode		Shipping Company	
Original	**Updated**	**Original**	**Updated**	**Original**	**Updated**
IN	0	Air	0	SC1	0
BD	1	Ocean	1	SC2	1
				SC3	**2**

Table 1: Categorical Features: Original vs Updated value

3.4. Data Splitting

The dataset was randomly split into training and validation sets, with 80% of the data allocated for training the model and the remaining 20% reserved for testing. The features that served as inputs were Source Country, Destination Country, Freight Cost, Gross Weight, Shipment Charges, Shipment Mode, and Shipping Company. The output feature was Shipping Time.

4. Model Building

The model-building process was conducted using ml-OS, a machine-learning platform designed to facilitate the development and evaluation of predictive models. For this project, several iterations were performed to evaluate the effectiveness of removing outliers and optimizing the model. The goal of these iterations was to refine the model and enhance its predictive accuracy by addressing data points that could negatively impact the model, including outliers and bad data.

- **Scenario 1:** This scenario served as the baseline and used the original dataset without any modifications.
- **Scenario 2:** This scenario used the dataset after removing the outliers detected during the exploratory data analysis (EDA).
- **Scenario 3:** This scenario used a filtered dataset that only retained good data points with an error rate of less than or equal to 5%. During the evaluation of the training and testing datasets from Scenario 2, it was observed that certain predicted shipping times were significantly different from the actual shipping times, some with really high errors. These instances were identified as bad data and subsequently removed to create a more reliable dataset.

It is important to note that although mlOS features an Autopilot function that ranks all applicable regression models based on their error, only the **Random Forest Regressor** was considered for final analysis because when other models were evaluated, they did not perform well. The Residual vs. Predicted Target and Predicted vs. Target graphs for these models exhibited vertical clustering, indicating that these models predict a narrow range of values and fail to capture the variability in the true target values. This evaluation highlights the limitations of the other models in accurately modeling the shipping time for the dataset. For an overview of the Residual vs. Predicted Target and Predicted vs. Target graphs for the other models, please refer to Figure 11.

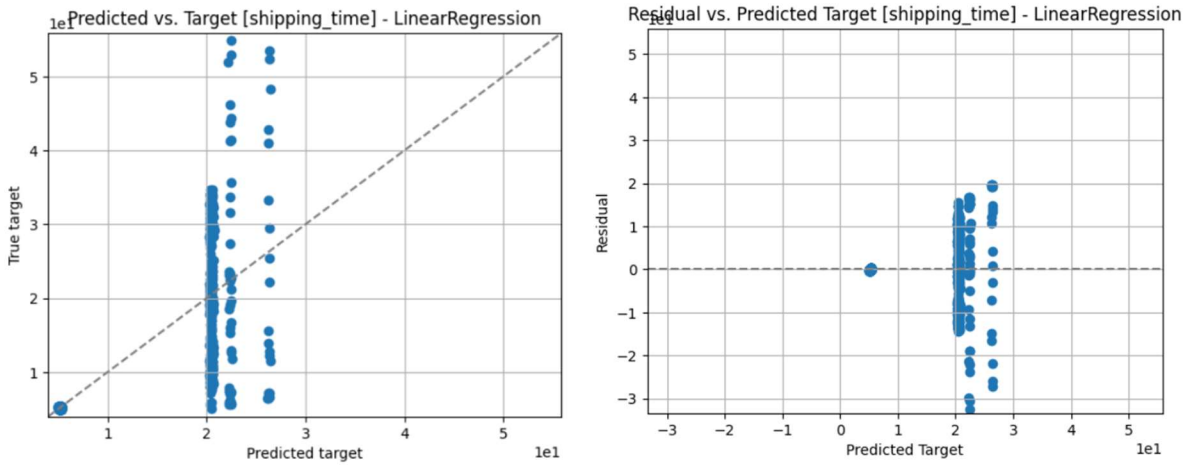

Figure 11: Other Regressor Model Graphs

Each scenario was meticulously checked to determine the impact of outlier removal on the model's performance. The results from these iterations provided insights into how outliers and bad data affect the predictive accuracy of the model, guiding the final model selection and optimization process. For detailed reference regarding the scenarios and results, please refer to Figures 12 to 14 and Table 1 below.

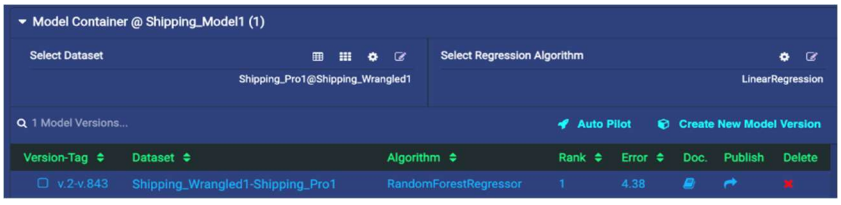

Figure 12: Scenario 1 Result

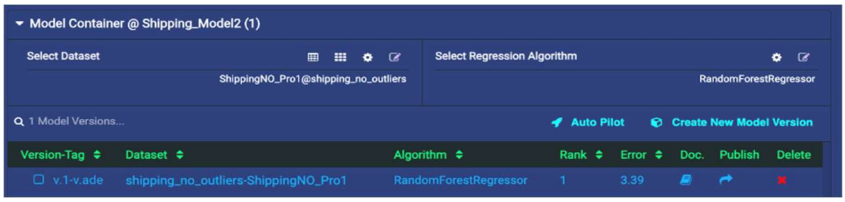

Figure 13: Scenario 2 Result

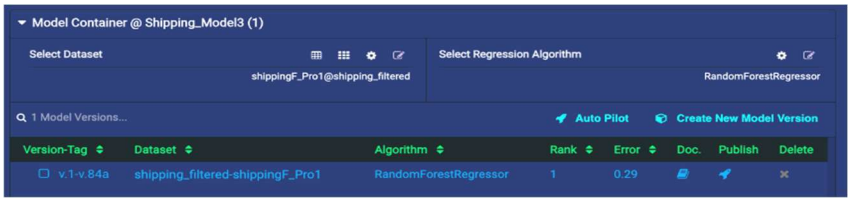

Figure 14: Scenario 3 Result

	Dataset Used	Result (Random Forest Regressor)
Scenario 1	Original	4.38
Scenario 2	Without Outliers	3.39
Scenario 3	Without Outlier and Bad Data	0.29

Table 2: Summary of Scenarios

5. Model Evaluation

Model evaluation was performed to assess the effectiveness of the **Random Forest Regressor** in predicting shipping days for the logistic department of the company. The evaluation is focused on Regression Model Performance, Residual vs Predicted Target and Predicted vs. Target.

5.1 Regression Model Performance

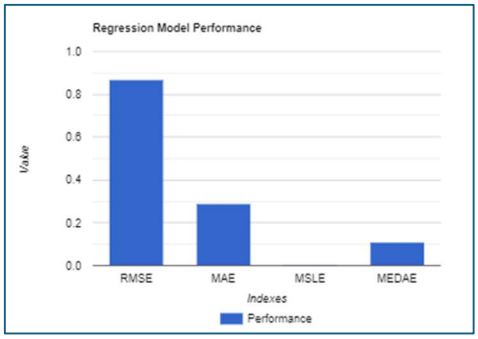

Figure 15: Regression Model Performance

- **Root Mean Squared Error-RMSE (0.87):** An RMSE of 0.87 indicates that, on average, the model's predictions are off by approximately 0.87 days. RMSE penalizes larger errors more than smaller ones, so it's useful for understanding the impact of larger deviations. A lower RMSE indicates better model performance. An RMSE of 0.87 suggests that the model is performing well, with predictions being close to the actual shipping time.

- **Mean Absolute Error- MAE (0.29):** The MAE is lower than the RMSE, which is expected since MAE does not penalize larger errors as much as RMSE does. An MAE of 0.29 days is quite good, indicating that the model's predictions are generally accurate and close to the true values.
- **Mean Squared Logarithmic Error - MSLE (0):** An MSLE of 0 indicates perfect predictions, particularly useful when dealing with targets that can grow exponentially. In this case, an MSLE of 0 suggests that there are no large errors, especially in the context of smaller values.
- **Median Absolute Error - MEDAE (0.11):** A MedAE of 0.11 days means that half of the errors are less than 0.11 days. This indicates that most of the predictions are very accurate, with most errors being minimal. This suggests that the model is consistent and reliable, with little influence from outliers.

The model exhibits strong predictive accuracy, with both MAE and MedAE indicating small average errors. The low RMSE also suggests that large errors are not common, reinforcing the model's reliability. The fact that MSLE is 0 and RMSE is not much higher than MAE suggests that the model's errors are not skewed towards larger values, meaning it performs well across different ranges of the target variable.

5.2 Residual vs Predicted Target

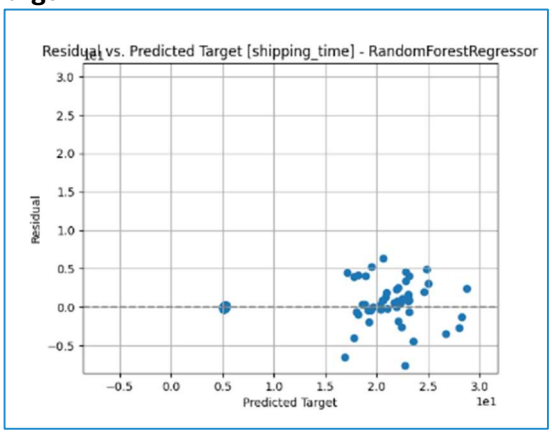

Figure 16: Residual vs Predicted Target

The residuals are plotted against the predicted shipping times. The following are the insights for this plot shows for the Random Forest Regressor model:

1. The residuals appear to be centered around zero, which is generally a good sign, indicating that the model is not consistently over- or under-predicting.
2. The residuals seem to be spread evenly around the zero line for most predictions, with a slight concentration of points around certain values.
3. There are some residuals that deviate slightly from zero, but they are not extreme, indicating that while the model makes occasional errors, these errors are not large.
4. The spread of residuals does not increase or decrease significantly as the predicted value increases, suggesting that the model's error variance is relatively constant across the range of predictions. This is a desirable trait, as it indicates that the model is stable and consistent in its predictions.

5.3 Predicted vs. Target

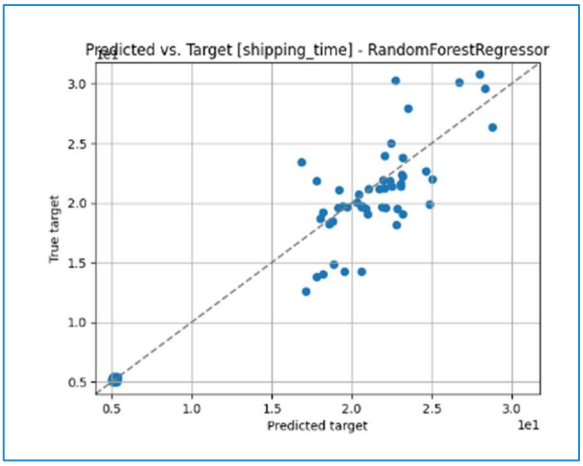

Figure 17: Predicted vs Target

This plot shows the relationship between the predicted shipping times (x-axis) and the actual (true) shipping times (y-axis). The following are the insights for this plot shows for the Random Forest Regressor model:

1. Many points are close to or aligned with the diagonal line, indicating that the model's predictions are generally close to the true shipping times. This is a good indication of the model's accuracy.
2. Some points deviate from the diagonal, indicating that there are cases where the model's predictions are less accurate. However, the deviations are not extreme, which suggests that while the model does make errors, they are not excessively large.
3. There appears to be a clustering of predictions, especially for shipping times between 1.5 and 2.5 days. This could suggest that the model is more confident or performs better within this range.
4. The fact that many predictions fall close to the diagonal line suggests that the model has learned the relationship between the input features and the shipping time quite well, with minimal bias.

These visualizations confirm that the model's predictions are typically close to the actual shipping times, and the errors are not systematically biased in any direction. The consistency in error variance and the alignment of predicted and actual values further support the conclusion that this is a well-performing regression model for the given task.

6. Model Governance

6.1. Model Development and Documentation

Data Scientists develop the model using the MLOS software. Document all steps, including data sources, preprocessing, feature engineering, and algorithms used (previously done in this paperwork). Documentation Includes model Performance like Metrics, validation results, and comparisons with baselines. After the deployment of the model, will be tracked changes and versions in MLOS.

6.2. Model Validation and Testing

Based on the logistics team's requirements for predicting shipping time, data scientists perform validation using MLOS tool and ensure the model meets predefined performance metrics (e.g. Error, RMSE). Conduct bias detection and fairness checks.

6.3. Approval Process

On the Model governance module in MLOS platform, The Data Science Manager reviews and evaluates the model based on documentation, validation results, and performance metrics. His decision could be to accept or reject the model.

- If accepted, proceed to deployment
- If rejected, provide feedback for further improvements

6.4. Deployment

After acceptance, Data Scientist deploy the model using MLOS deployment capabilities, ensuring deployment follows best practices for security and infrastructure.

6.5. Post-Deployment Monitoring and Maintenance

Quality Assurance/maintaining department for further taking care of the deployed model. This team will set up alerts for significant deviations in performance. Data scientists must retrain the model with new data as necessary to maintain accuracy.

When the model is operational, it's crucial to gather feedback from the sales, production, and logistics departments regarding its predictions and the actual delivery times.

6.6. Ethics and Compliance

It will be assigned an Ethics and Compliance Committee to review the model for ethical considerations such as fairness, transparency, and accountability. This will ensure the model complies with relevant regulations and organizational policies.

- **Legal Department:** Ensures the model complies with all relevant laws and regulations. Oversees adherence to internal policies and external regulatory requirements.
- **Risk manager:** Focus on ethical implications, ensuring the model does not propagate bias or unfair practices. Ensure data privacy and protection standards are maintained.

6.7. Stakeholder Engagement and Training

It will be provided training for stakeholders on how to interpret and use model predictions. This will enhance education of stakeholders on governance practices and the ethical use of the model.

6.8. Tools and Platforms

It will be leveraged MLOS capabilities for model development, validation, deployment, and monitoring.

7. Model Deployment

After developing and confirming the model's performance against the evaluation metrics, model deployment was conducted to enable real-time predictions. Model deployment involves making a

machine learning model accessible in a production environment to provide predictions based on new data. In this project, the Random Forest Regressor model, identified as the best model, was deployed to predict Shipping times.

The deployed model facilitates batch processing under the Data Scoring tab, allowing predictions for multiple scenarios and providing predicted shipping time on the model's training. To ensure dataset validation, it is important to score the dataset first. For more details about the Score Data process, refer to Figure 1.

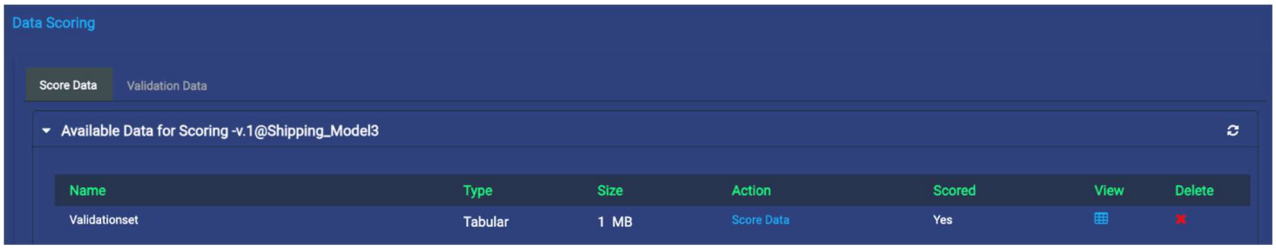

Figure 18: Score Data

To confirm that the data is scored, the validation data displays the predicted shipping time for each scenario. Each scenario presents both the actual and predicted shipping times along with all the other features affecting the prediction that served as inputs. By comparing the actual and predicted shipping times, it is evident that the difference is always less than one day, showcasing the model's effectiveness. For more details about the appearance and structure of the validated data, refer to Figure 19.

Row #	Predicted_shipping_time	shipping_time	source_country	destination_country	freight_cost	gross_weight	shipment_charges	shipment_mode	shipping_company
1	5.1192888	5.32546	0	1	93.1	55	0.75	0	2
2	5.2837246	5.19676	0	1	91.84	330	0.75	0	2
3	5.14907486	5.1713	0	1	92.4	765	1.05	0	1
4	5.114103	5.3419	0	1	93.87733272	1000	0.75	0	2
5	5.1461666083	5.21921	0	1	91.65	103.5	1.05	0	1
6	5.2680968	5.38345	0	1	90.97397711	300	1.05	0	1
7	5.2343613	5.08727	0	1	92.91	856	1.05	0	1
8	18.0187266	18.69745	0	1	90.5	205	0.9	1	0
9	5.1817354	5.05289	0	1	85.12	600	1.05	0	1
10	5.2665882	5.02778	0	1	85.97806032	1505	0.75	0	2
11	5.1937726	5.03171	0	1	87.2271581	1150	0.75	0	2
12	5.0976516	5.31829	0	1	87.24	120	0.75	0	2
13	5.0984886	5.37558	0	1	89.8	1500	0.75	0	2
14	5.1895348	5.30405	0	1	88.40030274	61	0.75	0	2
15	5.1149293	5.24387	0	1	86.99	25	0.75	0	2
16	5.208499	5.0191	0	1	87.93	291	0.75	0	2
17	5.221993	5.27361	0	1	91.56	200	1.05	0	1
18	5.2132378	5.40961	0	1	90.58	449.5	0.75	0	2
19	5.2519017	5.23345	0	1	87.82	1506	0.75	0	2
20	5.2683841	5.339	0	1	92.67	505	0.75	0	2

Figure 19: Validated Data

The deployed model also facilitates real-time predictions, allowing users to modify input variables such as Source Country, Destination Country, Freight Cost, Gross Weight, Shipment Charges, Shipment Mode, and Shipping Company. Based on these inputs, the model predicts the shipping time. Additionally, the model identifies the most significant variables influencing the predicted output, aiding stakeholders in making informed decisions to potentially alter the prediction. For an illustration of how different inputs affect predictions, refer to Figures 20 and 21.

For example, in Example 1, with '0' input for 'Air' in Shipment Mode, the predicted shipment time is 7.41 days, with 'Shipping Company' being the most influential variable. In Example 2, using the same inputs except for Shipment Mode, which is changed to '1' for 'Ocean', the shipment time increases to 12.89 days, still with 'Shipping Company' as the most influential variable.

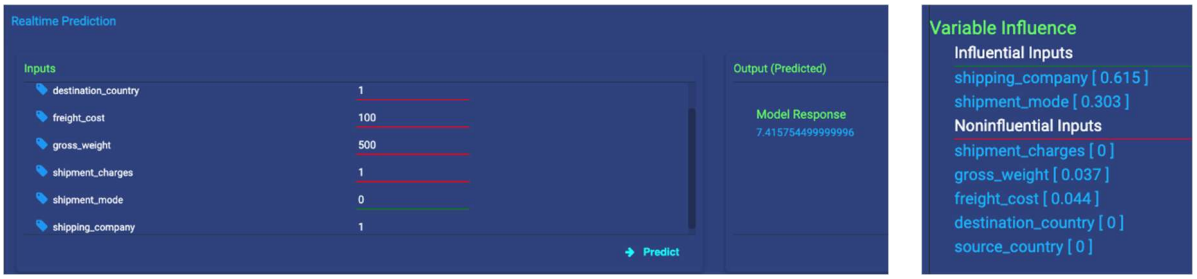

Figure 20: Real Time Prediction Example 1

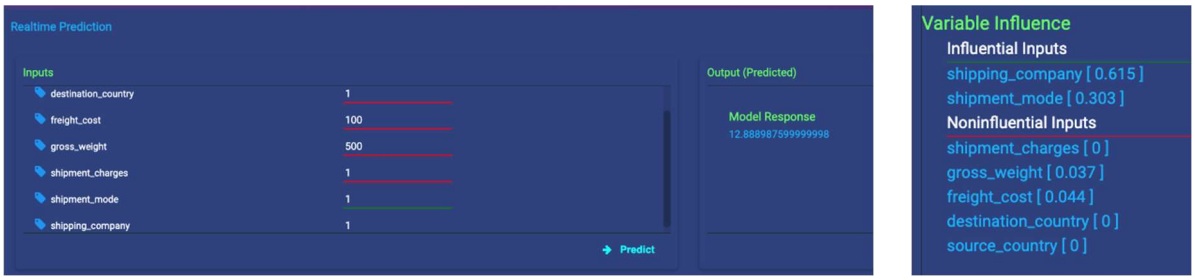

Figure 21: Real Time Prediction Example 2

Lastly, the Real-Time Predictions tab also displays the API status. This API allows new shipping data to be sent to the model for prediction and returns the predicted Shipping Time. The API status provides information on the number of scenarios run and the percentage of successful responses. Ideally, the success rate should always be 100%. For more details about the API status for this model, refer to Figure 22.

Figure 22: API Status

8. Conclusion

In conclusion, three different scenarios were evaluated using the Random Forest Regression model. The first scenario, which used the original dataset, resulted in an RMSE of 4.38. The second scenario, which excluded outliers, yielded an RMSE of 3.39. The third scenario, which removed both outliers and bad data, achieved the lowest RMSE of 0.29. Among these, the third scenario demonstrated the best performance.
The provided plots reinforce the metrics' indication that the Random Forest Regressor model is performing well in predicting shipping times. The residuals are centered around zero and evenly distributed, indicating that the model is making balanced predictions without systematic errors. The Predicted vs. True Target plot shows that most predictions are close to the actual values, with only minor deviations, suggesting that the model is both accurate and reliable.

These visualizations confirm that the model's predictions are typically close to the actual shipping times, and the errors are not systematically biased in any direction. The consistency in error variance and the alignment of predicted and actual values further support the conclusion that this is a well-performing regression model for the given task.

In the future, the model can be further enhanced by incorporating additional features such as weather conditions, holidays, and port congestion data to improve the accuracy of shipping time predictions. Additionally, integrating real-time data feeds will enable continuous learning and adaptation to be changing conditions, ensuring high prediction accuracy over time.

In real-life applications, this model can be deployed to assist logistics companies in optimizing their shipping routes and schedules. By accurately predicting shipping times, companies can make informed decisions to minimize delays and reduce operational costs. This model can also be integrated into customer-facing platforms to provide more accurate delivery estimates, thereby enhancing customer satisfaction. Furthermore, the model's ability to identify key variables influencing shipping times can help companies proactively address potential delays by adjusting shipping methods or choosing different carriers.

Overall, the project demonstrates the effectiveness of using machine learning techniques to predict shipping times. The model's deployment in a real-time environment has shown its capability to provide accurate and actionable predictions, with the difference between actual and predicted shipping times being less than one day in most scenarios. The use of the Random Forest Regressor model has proven to be particularly successful in this context. The project's success underscores the value of data-driven decision-making in logistics and highlights the potential for further advancements in predictive modeling.