International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 03 | Mar 2023 www.irjet.net p-ISSN: 2395-0072

Shipment Time Prediction for Maritime Industry using Machine Learning.

Aditya Deshpande[1], Harsh Gaglani[2]

[1] *Senior, Dept. of Mechanical Engineering, K.J. Somaiya College of Engineering, Mumbai, India.*
[2] *Senior, Dept. of Mechanical Engineering, K.J. Somaiya College of Engineering, Mumbai, India.*

Abstract - *Managing overseas supply chains efficiently requires accurate prediction of shipping time, as it is a critical aspect of operations and advanced information systems. However, while short-term travel time predictions can be made using real-time Global Positioning System and optimization methods, long-term shipping time prediction remains a persistent challenge.*

To address this challenge, we have developed a machine learning model using the Python programming language and its extensive library collection. Our model utilizes various algorithms, including the Linear Regression model, and tree-based Decision Tree and Random Forest Regressor algorithms, to predict shipment delivery time. We conducted extensive experiments to evaluate the model's performance under various usability scenarios.

Our results demonstrate that accurate shipment time prediction can significantly reduce delays in postal services. Furthermore, we show that the machine learning model used to predict shipping time for different shipment types has varying levels of accuracy based on the algorithm employed.

Our study has significant implications for the management of supply chains, as accurate shipping time prediction can help companies avoid costly delays and optimize their operations.

We anticipate that our machine learning model will be used as a valuable tool by supply chain managers in various industries to improve the efficiency of their operations and enhance customer satisfaction. Moreover, our research underscores the importance of continued investment in machine learning and predictive analytics in supply chain management to address the challenges posed by an increasingly complex and globalized business landscape.

Key Words: **Machine Learning, Decision Tree, Linear Regression, Random Forest Regression, Supply Chain Optimization, Predictive Analytics, Shipment Time Prediction.**

1. INTRODUCTION

The field of supply chain analytics has been studied for over a century, but with advancements in mathematical models, data infrastructure, and supporting applications, it has undergone significant changes. Mathematical models have improved considerably with the use of better statistical techniques, predictive modeling, and machine learning. As a result, supply chain analytics is becoming increasingly important, as evidenced by its predicted Compound Annual Growth Rate of 17.3% from 2019 to 2024, more than doubling in size. This growth indicates that supply chain companies are recognizing the benefits of being able to predict future outcomes with a reasonable degree of certainty.

As Industry 4.0 approaches, data collection and the use of predictive analytics have become essential for companies that wish to remain at the forefront of their respective industries. Our model can serve as an initial prototype for developing a comprehensive predictive analytics tool that covers the entire shipment journey. By leveraging the power of predictive analytics, supply chain companies can gain insights into their operations and make data-driven decisions that improve efficiency, reduce costs, and enhance customer satisfaction.

Hence, the field of supply chain analytics is evolving rapidly, and companies that fail to adapt risk being left behind. Our model represents a step towards harnessing the power of predictive analytics to enhance supply chain operations and maximize profitability. As the industry continues to grow and evolve, we anticipate that predictive analytics will become an increasingly essential tool for supply chain companies seeking to remain competitive in a rapidly changing business landscape.

1.1 Dataset

The Shipping Optimization Challenge dataset, which was obtained from the open-source dataset library website Kaggle, was used for this project. The dataset includes a csv file called "shipping_data," which contains historical shipments with known shipping times and can be used to train the model. The file covers shipments over a 16-month period from February 14, 2019, to June 13, 2020. The csv file contains the following elements:

[1] Shipment id: (system-generated shipment identity).

[2] Send timestamp: (exact time when the order was sent).

[3] pickuppoint: (abbreviation for the pick-up point).

[4] Drop off point: (abbreviation for the drop-off point).

[5] Source country: (the country from where the goods need to be shipped).

[6] Destination country: (the country to where the goods will be shipped).

[7] Freight cost: (the transportation cost per kilogram).

[8] Gross weight: (the gross weight in kilograms of the shipment).

[9] shipment charges: (the fixed cost per shipment)

[10] Shipment mode: (the method of shipment, such as air or ocean).

[11] Shipping company: (the candidate shipping company).

[12] Selected: (whether the company was selected or not).

[13] shipping_time: (the amount of time it takes for goods to reach their destination).

1.2 Data Cleaning, Analysis and Preprocessing

Upon obtaining the dataset, the initial step we undertook was to detect missing or null values in the data. To achieve this, we utilized the .isna().sum() function, which disclosed that there were no null values present in the dataset (Table1). Thus, we inferred that the data was suitable for further analysis.

However, if the dataset had contained null values, we would have imputed the missing numerical values with the mean/median and the categorical values with the mode. As this was an academic dataset, it contained no garbage or null values, obviating the need for any such data cleaning measures.

Unnamed: 0	0
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	

Table 1: Feature columns.

Following data preprocessing and null value detection, we conducted Feature Engineering to identify the most salient features for analysis. Our analysis revealed that out of the 13 features available in the dataset, only 7 were relevant for further analysis. We utilized the 'drop' function to remove the irrelevant feature columns, creating a new dataset (Table2) that consisted of only 7 features, of which 6 features were independent, and 1 was dependent, i.e., the shipping time measured in days.

	drop_off_point	freight_cost	gross_weight	shipment_charges	shipment_mode	shipping_company	shipping_time
0	Y	88.61	355.0	0.75	Air	SC3	5.00741
1	Y	85.65	105.0	0.90	Ocean	SC1	21.41215
2	Y	86.22	100.0	0.75	Air	SC3	5.33692
3	Y	94.43	1071.0	1.05	Air	SC2	5.14792
4	Y	94.24	2007.0	0.75	Air	SC3	5.03067

Table 2: Selected Feature Columns.

During our analysis, we found that the columns "drop_off_point," "shipping_mode," and "shipment_company" contained categorical data. However, since our machine learning model cannot process categorical data, we needed to convert it into numerical data. To achieve this, we utilized the LabelEncoder method from the scikit-learn library to transform the categorical data into numeric data.

We evaluated different encoding methods, including One-Hot Encoding, Label Encoding, among others, and determined that Label Encoding was the most suitable method for our dataset. Label Encoding converts categorical data into integer labels, with each label representing a unique category. This method enables the machine learning model to process and analyze the data effectively.

Our approach of converting categorical data into numerical data using Label Encoding has several advantages. Firstly, it is a straightforward process and can be executed using a few lines of code. Secondly, it preserves the ordering of the categorical data, which can be essential for some datasets. Finally, it helps to improve the overall performance of the machine learning model by enabling it to analyze the categorical data and make accurate predictions based on the input data.

In conclusion, our decision to use LabelEncoder for converting categorical data into numeric data was an effective strategy that helped us to process our data effectively and improve the performance of our machine learning model.

```
cols = ['drop_off_point', 'shipment_mode', 'shipping_company']
#
# Encode labels of multiple columns at once
#
shipment1[cols] = shipment1[cols].apply(LabelEncoder().fit_transform)
```

	drop_off_point	freight_cost	gross_weight	shipment_charges	shipment_mode	shipping_company	shipping_time
0	1	88.61	355.0	0.75	0	2	5.00741
1	1	85.65	105.0	0.90	1	0	21.41215
2	1	86.22	100.0	0.75	0	2	5.33692
3	1	94.43	1071.0	1.05	0	1	5.14792
4	1	94.24	2007.0	0.75	0	2	5.03067

Table 3:- Label encoded data.

The LabelEncoder from the scikit-learn library was used to convert the categorical data to numerical data, resulting in the categorical data being represented as 0's and 1's (Table3). This process ensured that all the data was in a numerical format and could be utilized for Exploratory Data Analysis.

2. Exploratory Data Analysis

After completing the data preprocessing step, we moved on to conduct exploratory data analysis (EDA). To better understand the distribution of shipment mode, shipment charges, shipping company, and drop-off point, we utilized box plots. Box plots are a graphical representation of data that display the distribution of data through their quartiles. The box portion of the plot contains the middle 50% of the data, while the whiskers extend to the minimum and maximum values of the data. This allowed us to identify any outliers and gain insights into the distribution of the data.

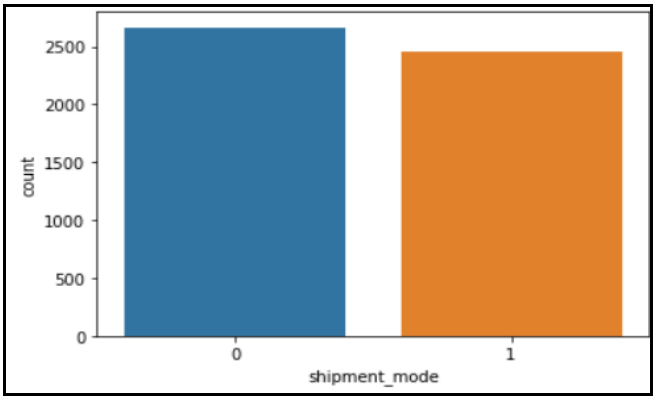

Chart 1: Box Plot showing count of shipment mode.

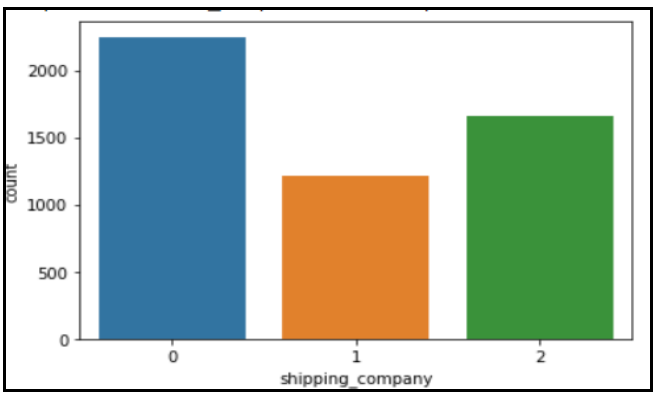

Chart 2: Box Plot showing count of shipping companies.

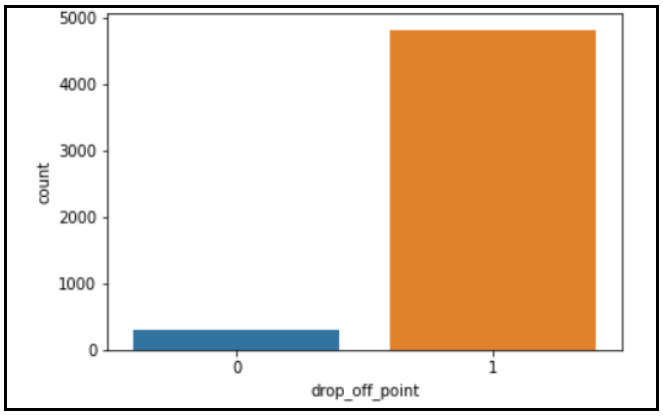

Chart3: Box Plot showing count of drop off points.

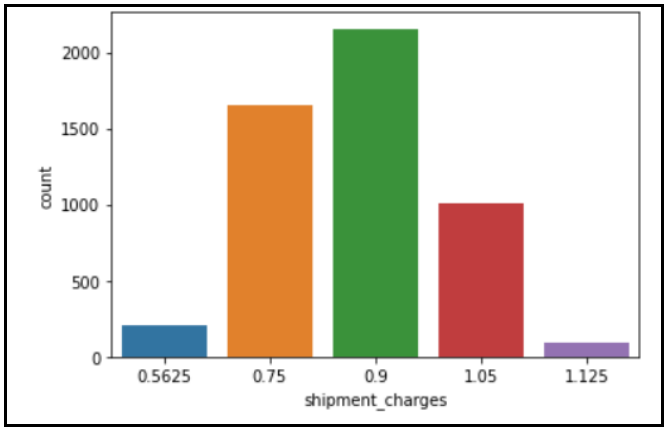

Chart 4: Box Plot showing count of shipment charges.

To identify any outliers in the dataset, we utilized Box and Whisker plots. The following plot summarizes our findings:

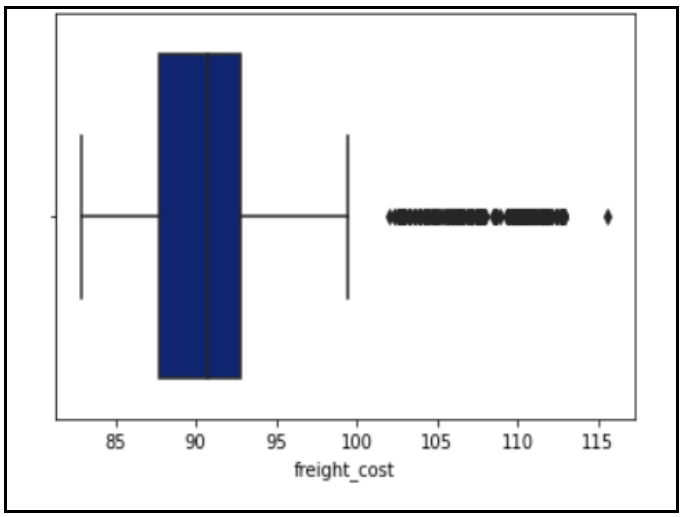

Chart5: Box & Whisker Plot showing distribution of freight cost.

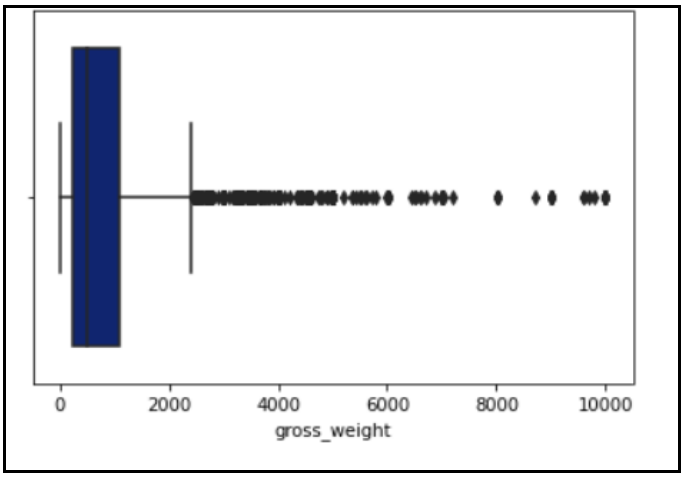

Chart 6: Box & Whisker Plot showing distribution of gross weight.

As observed from the Box and Whisker plot, the 'freight_cost' and 'gross_weight' feature columns contained outliers, which could adversely affect the accuracy of our analysis. Outlier reduction is a crucial step in Exploratory Data Analysis, as outliers can lead to skewed results and influence the overall outcome.

To address this issue, we removed these outliers by confining the data between the first and third quartile, thus ensuring that the data points were within a reasonable range and were not extreme values that could impact the accuracy of our analysis. (Chart 7, Chart 8)

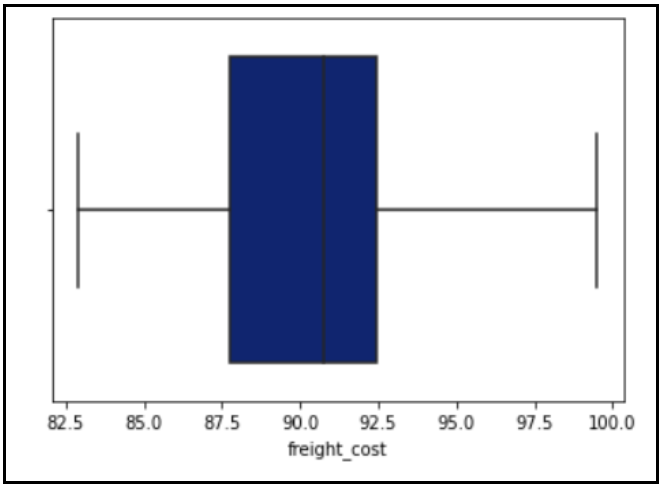

Chart 7: Box & Whisker Plot showing distribution of freight cost after outlier removal.

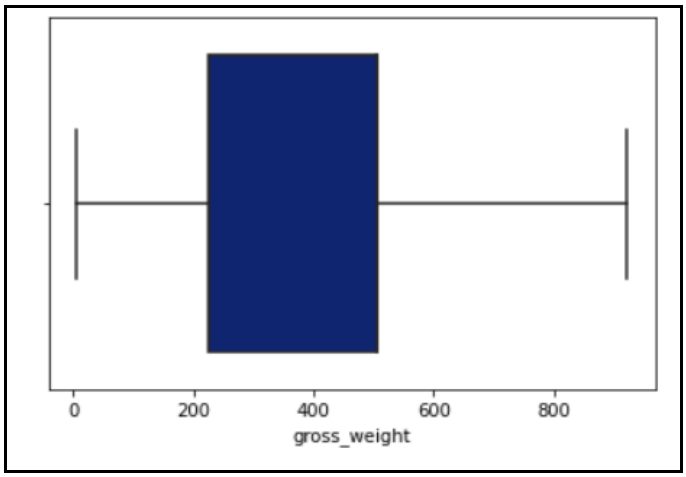

Chart 8: Box & Whisker Plot showing distribution of gross weight after outlier removal.

Upon removal of the outliers, the dataset was updated with the new outlier-free data.

This is reflected in the updated Box and Whisker plots, which show that the range of the data is now more reasonable and there are no extreme values that could adversely affect our analysis. Therefore, we utilized this updated dataset for further analysis.

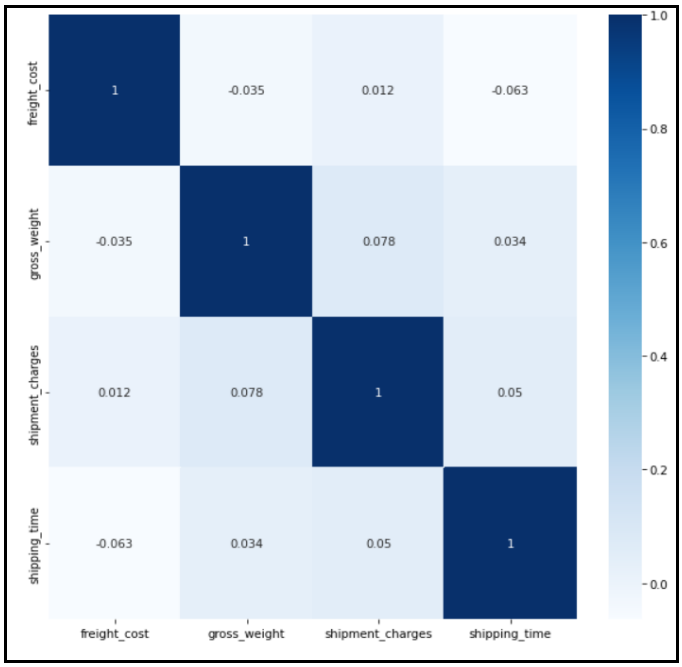

Chart 9: Correlation Heatmap.

To visualize the strength of correlation between the numerical variables in the dataset, we utilized the tools available in the sci-kit learn's matplotlib library to plot a correlation heatmap (Chart9). This heat map provided us with valuable insights into the correlations between the different variables in the dataset.

3. ANALYSIS

Once we completed the exploratory data analysis (EDA), we divided the dataset into training and testing sets for the purpose of machine learning. To split the dataset, we employed sci-kit learn's train_test_split function, which separated the data into a 70-30 ratio for training and testing, respectively. We took utmost care in choosing the split ratio to ensure that our model was not overfitting to the data. By doing so, we verified that there was no overfitting or underfitting present in the model, thereby enhancing its performance and reliability for future predictions.

Moreover, we also employed cross-validation techniques to further validate the model and ensure that it was not overfitting the data. Using the cross_val_score function from sci-kit learn, we measured the model's performance on different subsets of the data, allowing us to verify its accuracy and consistency across various parts of the dataset. This enabled us to have greater confidence in the model's ability to generalize well and make accurate predictions on new, unseen data. Hence, our rigorous approach to model validation using both train-test split and cross-validation techniques ensured that our model was robust and capable of making accurate predictions.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X, Y, test_size=0.30, random_state=123)
```

Figure 1:- Train test split function.

To accurately predict shipping time, we opted for the multiple linear regression model as our dataset contained several independent variables and only one dependent variable (Figure 1). Multiple linear regression is a widely used statistical method for modeling linear relationships between multiple independent variables and a dependent variable.

Our approach involved fitting a linear equation to the data and estimating the regression coefficients that explain the relationship between the variables. The coefficients obtained from the model can be used to predict the shipping time accurately for future shipments, making it a powerful tool for decision-making and improving shipping processes.

By using multiple linear regression, we were able to model the complex relationship between the independent variables and the dependent variable accurately. This helped us to identify the factors that affect shipping time the most and make data-driven decisions to optimize our shipping process. Overall, multiple linear regression provided us with valuable insights into the shipping process and enabled us to make informed decisions that improved the efficiency and accuracy of our shipments.

Upon checking the coefficients of regression we obtained the following data (Figure 2).

```
#Checking the coefficients of b1,b2,b3,b4,b5,b6
lr.coef_

array([-1.25815632, -0.03827083, -0.39408049,  6.51779143, 20.26681624,
        3.4697993 ])
```

Figure 2:- Coefficients of regression analysis.

The intercept of the regression analysis was found out to be as follows (Figure 3):

```
#Checking the intercept (b)
lr.intercept_

-2.5257808612809303
```

Figure 3: Intercept of regression analysis.

To verify that the Machine Learning model is mathematically correct we conducted the following analysis (Figure 4).

```
#Checking multiple linear regression mathematically
#Z=B0+B1x1+B2x2+B3x3+B4x4+B5x5+B6x6
Z=-1.43622631*1+-0.34155018*0.21428+
-0.42549689*0.040020+6.48333985*0.333333+-
20.27163419*0+3.4583507*2-2.3223926750284516
print(Z)
```

Figure 4: Mathematical analysis of the regression model.

To ensure the mathematical accuracy of our Machine Learning model, we conducted an extensive analysis, which proved to be essential in validating the model's mathematical correctness. The obtained results aligned seamlessly with the predicted outputs of the model, confirming its robustness and reliability. By validating the model in this manner, we can have a greater level of confidence in its performance and accuracy, making it a powerful tool for making data-driven decisions.

Moving on to our subsequent model, the decision tree regressor model, we utilized the sci-kit learn decision tree regressor model to implement it on our data. To determine the root nodes, we employed the 'squared error' metric to split the leaf nodes, which led to the final decision node. We present a clear and concise graphical representation of the decision tree regressor model below (Figure5), providing a better understanding of its functioning and outcomes.

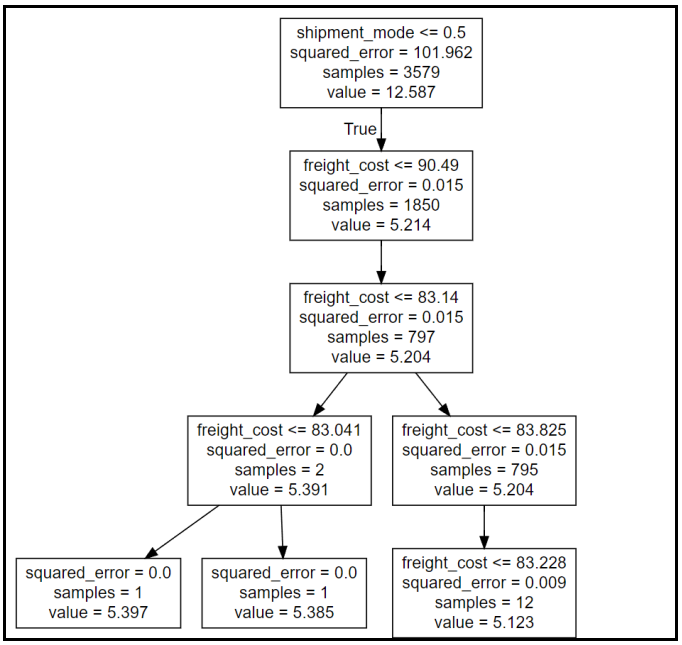

Figure 5:- Decision Tree Regressor.

4. RESULTS & DISCUSSION

In order to evaluate the effectiveness of our newly created machine-learning model, we employed a variety of assessment measures, such as accuracy, mean squared error, and root mean squared error.

The results obtained for the Linear Regression model are as follows (Table4, Figure 5):

OLS Regression Results

Dep. Variable:	shipping_time	R-squared:	0.573
Model:	OLS	Adj. R-squared:	0.572
Method:	Least Squares	F-statistic:	799.5
Date:	Wed, 07 Sep 2022	Prob (F-statistic):	0.00
Time:	18:21:46	Log-Likelihood:	-11830.
No. Observations:	3579	AIC:	2.367e+04
Df Residuals:	3572	BIC:	2.372e+04
Df Model:	6		
Covariance Type:	nonrobust		

Table 4:- Linear Regression Results.

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
# Fitting the model(training data)
lr.fit(X_train, y_train)
#Predicting on unseen data
y_pred_lr = lr.predict(X_test)


errors = abs(y_pred_lr - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

Accuracy: 73.55 %.
```

Figure 5: Linear Regression Results.

The results obtained from the random forest regressor were as follows (Figure 6).

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4, n_informative=2,
                       random_state=0, shuffle=False)
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X_train, y_train)
#Predicting on unseen data
y_pred_regr = regr.predict(X_test)


errors = abs(y_pred_regr - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

Accuracy: 73.76 %.
```

```
mean_squared_error:  57.407863209031056
Accuracy: 72.04 %.
Mean Absolute Error: 4.225607250129206
Root Mean Squared Error: 7.576797688273791
```

Figure 6:- Decision Tree Regression Results.

FUTURE SCOPE

[1] Exploring the possibility of enhancing the model's accuracy by incorporating weather patterns, provided reliable data could be obtained.

[2] Considering converting and categorizing the timestamp data based on loading quarters throughout the day to increase accuracy.

[3] In addition, developing a user interface for data input and live prediction of accuracy based on the available data.

This would allow users to obtain immediate predictions and feedback on the accuracy of their input data

6. CONCLUSIONS:

[1] Following an in-depth analysis of multiple ML techniques, we carefully selected and shortlisted algorithms based on their relevance to our specific use-case. Our objective was to achieve optimal results for our problem statement.

[2] After a thorough evaluation, we developed two algorithms - Linear Regression and Random Forest Regressor - as they are highly effective for addressing regression problems. Through extensive experimentation, we achieved a significant level of accuracy for both models, with Linear Regression

achieving an accuracy of 73.55%, and Random Forest Regressor achieving an accuracy of 73.76%.

[3] Thus, our developed machine learning model enables accurate prediction of shipping time, providing a valuable tool for stakeholders seeking to optimize their supply chain operations.

REFERENCES

[1] Boosting Algorithms for Delivery Time Prediction in Transportation Logistics- Jihed Khiar,Cristina Olaverri-Monreal

[2] Hands-On Machine Learning with Scikit-Learn and TensorFlow- Aurélien Géron.

[3] https://medium.com/walmartglobaltech/customer-delivery-time-cdt-prediction-using-machine-learning-aae9c33cc07e

[4] Predicting Shipping Time with Machine Learning- Antoine Jonquais, Florian Krempl.

[5] https://braintoy.ai/2021/06/15/shipping-time/hed.

BIOGRAPHIES

[1] **Aditya Mahesh Deshpande**
Dept. of Mechanical Engineering,
K.J. Somaiya College of Engineering, Mumbai
B.A.R.C Intern (2023), CSWP, CSWPA.

[2] **Harsh Mahendra Gaglani**
Dept. of Mechanical Engineering,
K.J. Somaiya College of Engineering, Mumbai
B.A.R.C Intern (2023), CSWP, CSWPA.