Supply Chain Machine Learning Project

Esmeralda Hernandez De Loa, Steven Buks, Divya D'Souza
CS 463/663 – Foundations of Machine Learning (Spring 2025)
May 15, 2025

Abstract

Online shopping places that produce consumers' physical items must tackle the issue of making sure that their supply chain can ship orders out so that they may arrive at the customers on time. Here, we use different machine learning algorithms to determine which model is better able to predict whether or not an order will arrive at a customer on time. Through investigation into our data, we can get a good grasp of what is occurring in this business's supply chain to be able to interpret the top features of our best-performing model (CART) that are affecting whether or not an order will be late.

Contents

1 Introduction

More and more people are turning towards buying items online rather than in a physical location. This means that e-commerce businesses can expect to see more traffic on their websites and transactions to be made on their platform. Now that you have a growing number of customers, these e-commerce businesses have to work towards keeping their customers so that they continue shopping with you again and so that they do not move on to buy from your competitors. For an e-commerce or online marketplace styled business, the supply chain process of getting the final product to customers at the given data and time is very important to master to ensure that customers stay happy, loyal, and that the number of transactions and new customers increases.

Our interest in the relative importance of shipments arriving on time for an online business has motivated us to look into the matter more closely to see what factors contribute to whether or not a shipment will arrive late to a customer or not as well as how we can create a model that can best help a business predict whether the shipment will arrive late or not. To do this, we have obtained a dataset on a business's supply chain data and have engineered a binary categorical target variable, "Late Arrival", and implemented classification models on the data to be able to determine which models we have implemented would best be able to predict whether a shipment will arrive late or not.

1.1 Research Question(s)

For this project, we aim to answer the following questions: Can we develop a model that can accurately predict whether or not a shipment will arrive late or not? Additionally, what are the features that are most important in determining if a shipment will be late?

1.2 Target Audience

By performing this research, companies, specifically those that have an online marketplace to conduct business of their physical item, will benefit the most from our investigations and insights. By answering our research questions, a company will be able to implement tactics to improve the supply chain systems so that the rate of shipments arriving on time or early increases, so that they may gain customer loyalty, increase customer retention rates, and ultimately increase the company's profits. Additionally, a good predictive model can help a company provide customers with an accurate order arrival date so that customers can make informed shopping decisions that work for them and so that they may continue to view the company as being reliable.

2 Related Work

Prior work has demonstrated that it is, in fact, possible and useful to create a predictive model determining whether or not a shipment will arrive at a customer on time. Mateusz Wyrembek and others wrote about their experience in using machine learning in determining delivery delay in the article "Causal machine learning for supply chain risk prediction and intervention planning". Wyrembek approaches predicting whether there will be a delay in shipping or not by using causal machine learning algorithms such as Average Treatment Effect (ATF) and Conditional Average Treatment Effect (CATE). Causal machine learning models aim to estimate causal effects as opposed to focusing on prediction, which is what those models we have learned in class do. After noticing that this report only uses causal machine learning algorithms for determining its target variable, we decided to use the predictive machine learning algorithms we have learned in class to uncover another way to handle the issue of determining if a shipment will arrive on time or not.

3 Methods

3.1 Dataset(s) & Preprocessing

We used a dataset from Kaggle called *"DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS"*. While some of us were able to download and upload the dataset straight into google drive, some of us had to use GitHub Gist to access our data. This meant that we had to manually upload our dataset to GitHub Gist. We did so using git since our data was ~96MB and the web interface only supported up to 25MB of data.

Upon inspection, we learned that the dataset had 180,519 rows and 53 columns. We expected a good amount of the columns to be redundant and consulted the data dictionary(fig 3.1.1) to trim out features that would not give us meaningful insights to the question we were trying to explore.

FIELDS	DESCRIPTION
Type	: Type of transaction made
Days for shipping (real)	: Actual shipping days of the purchased product
Days for shipment (scheduled)	: Days of scheduled delivery of the purchased product
Benefit per order	: Earnings per order placed
Sales per customer	: Total sales per customer made per customer
Delivery Status	: Delivery status of orders: Advance shipping , Late delivery , Shipping canceled , Shipping on time
Late_delivery_risk	: Categorical variable that indicates if sending is late (1), it is not late (0).
Category Id	: Product category code
Category Name	: Description of the product category
Customer City	: City where the customer made the purchase
Customer Country	: Country where the customer made the purchase
Customer Email	: Customer's email
Customer Fname	: Customer name
Customer Id	: Customer ID
Customer Lname	: Customer lastname
Customer Password	: Masked customer key
Customer Segment	: Types of Customers: Consumer , Corporate , Home Office
Customer State	: State to which the store where the purchase is registered belongs
Customer Street	: Street to which the store where the purchase is registered belongs
Customer Zipcode	: Customer Zipcode
Department Id	: Department code of store

Department Name	: Department name of store
Latitude	: Latitude corresponding to location of store
Longitude	: Longitude corresponding to location of store
Market	: Market to where the order is delivered : Africa , Europe , LATAM , Pacific Asia , USCA
Order City	: Destination city of the order
Order Country	: Destination country of the order
Order Customer Id	: Customer order code
order date (DateOrders)	: Date on which the order is made
Order Id	: Order code
Order Item Cardprod Id	: Product code generated through the RFID reader
Order Item Discount	: Order item discount value
Order Item Discount Rate	: Order item discount percentage
Order Item Id	: Order item code
Order Item Product Price	: Price of products without discount
Order Item Profit Ratio	: Order Item Profit Ratio
Order Item Quantity	: Number of products per order
Sales	: Value in sales
Order Item Total	: Total amount per order
Order Profit Per Order	: Order Profit Per Order
Order Region	: Region of the world where the order is delivered : Southeast Asia ,South Asia ,Oceania ,Eastern Asia, West Asia , West of USA , US Center , West Africa, Central Africa ,North Africa ,Western Europe ,Northern , Caribbean , South America ,East Africa ,Southern Europe , East of USA ,Canada ,Southern Africa , Central Asia , Europe , Central America, Eastern Europe , South of USA
Order State	: State of the region where the order is delivered
Order Status	: Order Status : COMPLETE , PENDING , CLOSED , PENDING_PAYMENT ,CANCELED , PROCESSING ,SUSPECTED_FRAUD ,ON_HOLD ,PAYMENT_REVIEW
Product Card Id	: Product code
Product Category Id	: Product category code
Product Description	: Product Description
Product Image	: Link of visit and purchase of the product
Product Name	: Product Name
Product Price	: Product Price
Product Status	: Status of the product stock :If it is 1 not available , 0 the product is available
Shipping date (DateOrders)	: Exact date and time of shipment
Shipping Mode	: The following shipping modes are presented : Standard Class , First Class , Second Class , Same Day

Fig 3.1.1 data dictionary

Two columns had null values, ‘Order Zipcode’ and ‘Product Description’, and were dropped since they did not provide any meaningful information. The features we decided to remove are mostly personally identifying features like customer name, etc. We dropped continuous variables like zip codes that would not give us any valid statistics of late deliveries. We also dropped columns that would not give us meaningful information if we one-hot-encoded it. With this in mind, these are the columns that we collectively decided to drop:'Category Id', 'Category Name', 'Customer Email', 'Customer Fname', 'Customer Lname', 'Customer Id', 'Customer Segment', 'Customer Password', 'Order Customer Id', 'Order Id', 'Order Item Cardprod Id', 'Order Item Id', 'Product Card Id', 'Product Category Id', 'Product Description', 'Product Image', 'Product Name','Benefit per order','Delivery Status','Customer Street','Customer Zipcode','Market','Order Region','Product Status','Order Zipcode','Department Id'.

3.2 EDA & Feature Engineering

Before EDA

The columns 'Days for shipping (real)', 'Days for shipment (scheduled)' were used to create a ‘Delay’ column that told us how many days the package would be delayed by. A negative value indicates the package arrived before time, a zero indicates that the package arrived when it said it would, and a positive value indicates it was delayed. We use this variable as a continuous variable to analyze trends. We then used this ‘Delay’ column to create our target variable, ‘Late Arrival’, which would be a binary of whether the package was delayed or not.

EDA

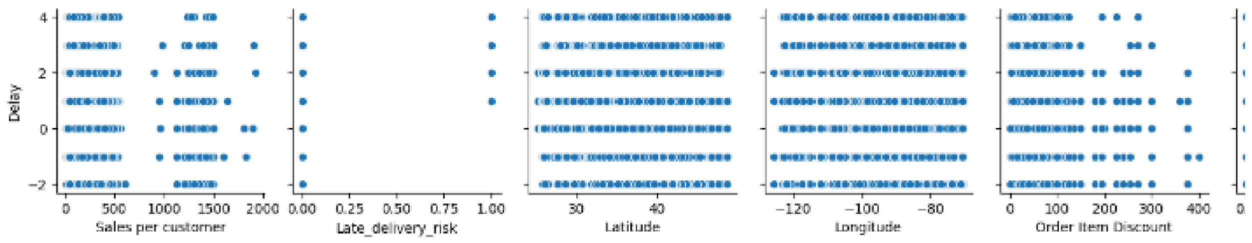
These were the steps of EDA that then gave us information on how the features need to be further engineered.

Descriptive Statistics

After cleaning the dataset and creating new features, descriptive statistics (like df.head(), df.info(), df.describe(), etc) were computed to summarize the numerical and non-numerical attributes of the dataset.

Pairplot Analysis

A pairplot was generated to visualize the relationships between the `'Delay'` and other numeric variables. This plot helps in understanding potential correlations and patterns in the data. The pairplot indicated that there were no obviously linear relationships, thus prompting us to consider KNN as a supervised learning model.



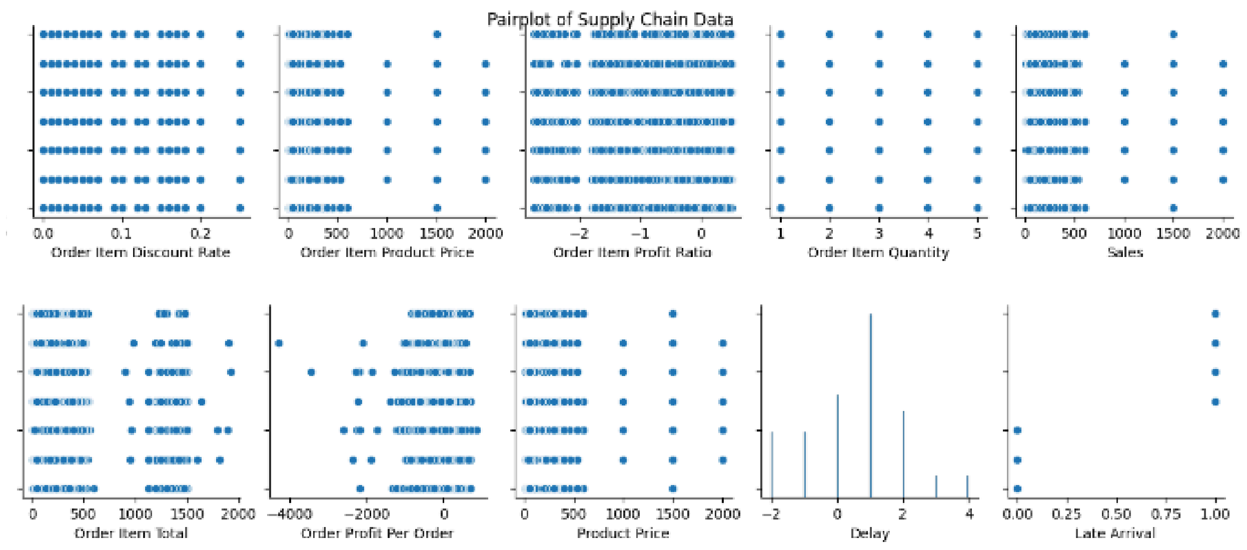
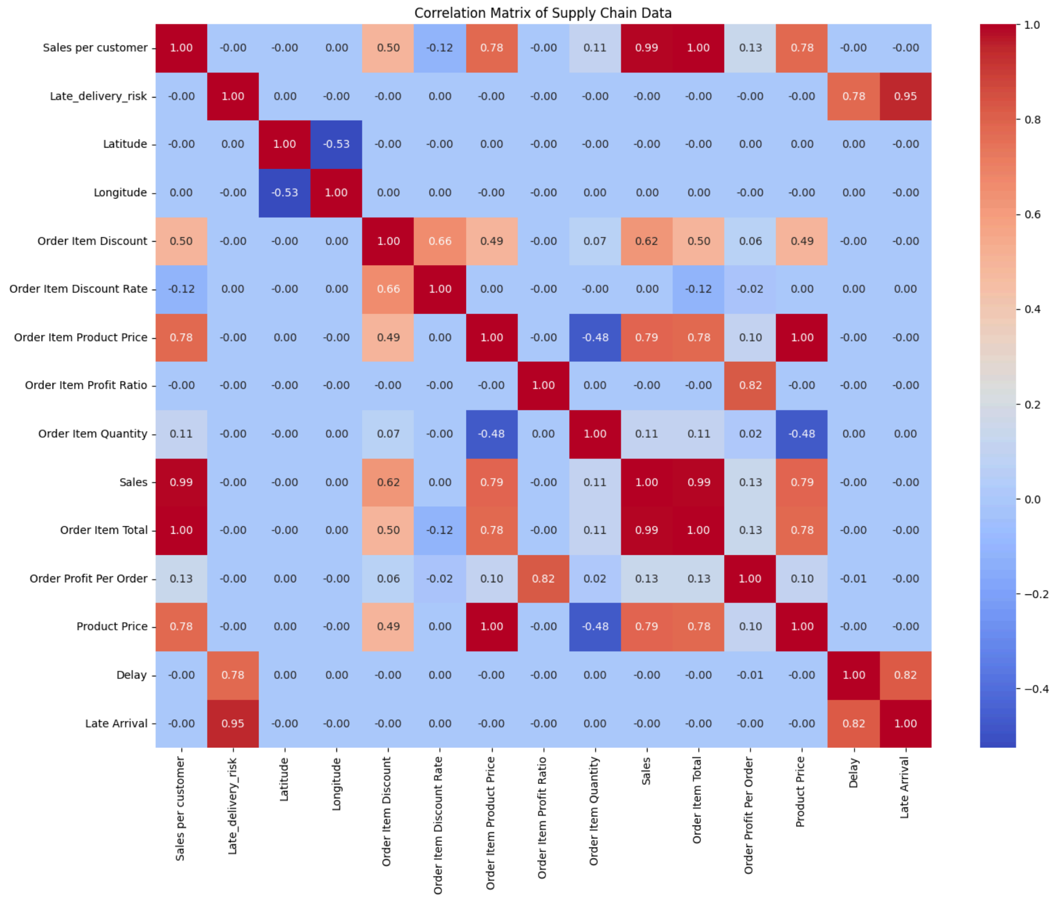


Fig 3.2.1 pairplots of all other numerical features against `Delay`

Correlation Matrix

A correlation matrix was constructed to quantify and visualize the linear relationships between all pairs of numerical columns in the dataset. This heatmap told us that there was some multicollinearity and also, no apparent correlations between other features and 'Delay'.



	Sales per customer	Late_delivery_risk	Latitude	Longitude	Order Item Discount	Order Item Discount Rate	Order Item Product Price	Order Item Profit Ratio	Order Item Quantity	Sales	Order Item Total	Order Profit Per Order	Product Price	Delay	Late Arrival
Sales per customer	1.00	-0.00	-0.00	0.00	0.50	-0.12	0.78	-0.00	0.11	0.99	1.00	0.13	0.78	-0.00	-0.00
Late_delivery_risk	-0.00	1.00	0.00	-0.00	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	0.78	0.95
Latitude	-0.00	0.00	1.00	-0.53	-0.00	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	0.00	0.00	0.00	-0.00
Longitude	0.00	-0.00	-0.53	1.00	0.00	0.00	-0.00	-0.00	0.00	0.00	0.00	-0.00	-0.00	0.00	-0.00
Order Item Discount	0.50	-0.00	-0.00	0.00	1.00	0.66	0.49	-0.00	0.07	0.62	0.50	0.06	0.49	-0.00	-0.00
Order Item Discount Rate	-0.12	0.00	-0.00	0.00	0.66	1.00	0.00	-0.00	-0.00	0.00	-0.12	-0.02	0.00	0.00	0.00
Order Item Product Price	0.78	-0.00	0.00	-0.00	0.49	0.00	1.00	-0.00	-0.48	0.79	0.78	0.10	1.00	-0.00	-0.00
Order Item Profit Ratio	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	1.00	0.00	-0.00	-0.00	0.82	-0.00	-0.00	-0.00
Order Item Quantity	0.11	-0.00	-0.00	0.00	0.07	-0.00	-0.48	0.00	1.00	0.11	0.11	0.02	-0.48	0.00	0.00
Sales	0.99	-0.00	-0.00	0.00	0.62	0.00	0.79	-0.00	0.11	1.00	0.99	0.13	0.79	-0.00	-0.00
Order Item Total	1.00	-0.00	-0.00	0.00	0.50	-0.12	0.78	-0.00	0.11	0.99	1.00	0.13	0.78	-0.00	-0.00
Order Profit Per Order	0.13	-0.00	0.00	-0.00	0.06	-0.02	0.10	0.82	0.02	0.13	0.13	1.00	0.10	-0.01	-0.00
Product Price	0.78	-0.00	0.00	-0.00	0.49	0.00	1.00	-0.00	-0.48	0.79	0.78	0.10	1.00	-0.00	-0.00
Delay	-0.00	0.78	0.00	0.00	-0.00	0.00	-0.00	-0.00	0.00	-0.00	-0.00	-0.01	-0.00	1.00	0.82
Late Arrival	-0.00	0.95	-0.00	-0.00	-0.00	0.00	-0.00	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	0.82	1.00

Visualizing Delivery Location Trends

Since we had delivery location latitude and longitude, and all deliveries were localized to the USA and Puerto Rico, we thought it could be interesting to see if there was any apparent, visual correlation between late deliveries and the location they were delivered to. Upon inspection, the clusters of late deliveries seem pretty similar to the clusters of deliveries that were on time. Hence, can conclude that there were no visual correlations between delivery location and late deliveries. We would need supervised models to get more information on this.

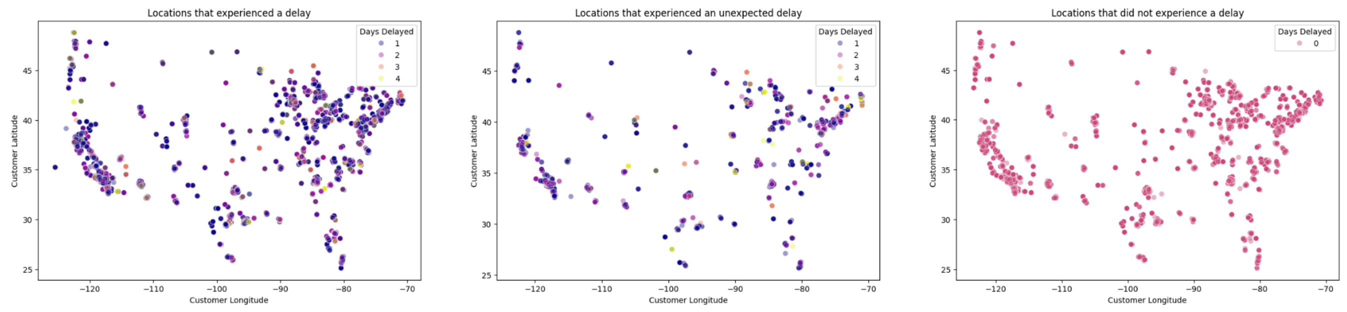


Fig 3.3.3 clusters of late deliveries(purple) and on time deliveries(pink)

Categorical Data Handling

Categorical variables such as `'Department Name'`, `'Shipping Mode'`, and `'Customer Country'` were encoded using dummy variables to prepare the dataset for modeling. This step ensures that categorical data can be used effectively in machine learning algorithms.

Temporal Feature Engineering

Date and time features `'order date (DateOrders)'` and `'shipping date (DateOrders)'` were parsed and transformed into separate columns (`'order_year'`, `'order_month'`, `'order_day'`, `'order_hour'`, `'order_minute'`, `'shipping_year'`, `'shipping_month'`, `'shipping_day'`, `'shipping_hour'`, `'shipping_minute'`). These features enable analysis and modeling based on temporal patterns in the data.

Final Dataset Preparation

After feature engineering and encoding, columns that were primarily kept for analysis like `'Type'`, `'Delay'`, and `'Late_delivery_risk'` were removed from the dataset to prepare it for modeling. This ensures that the dataset is clean, structured, and ready for machine learning algorithms.

Separate Dataset for Location Information

One-hot-encoding location information like `'Order State'`, `'Order Country'`, and `'Order City'` resulted in creating over 4,000 additional columns. Creating a dataframe that only analysed these variables with `'Late Arrival'` as the target, could give us insight into whether it could be meaningful to explore questions like *"Are there shipping locations that can be optimized to not result in late deliveries?"*

After checking feature importance with Logistic Regression(chosen since all columns are binary), we concluded that the product origin locations did not provide all that much information. We validated this deduction when we ran our models without the locations encoded and that significantly improved our classification reports for all models.

3.3 Models

3.3.1 Model 1: Random Forest

Random forest is a machine learning algorithm that creates and combines multiple decision trees. Given that our target variable is a binary, categorical variable, we chose to factor in this model to find which model is best able to predict our target variable correctly. Other advantages to this model that made us interested in running it with our data is that it reduces overfitting, is able to handle large data sets, and has the ability to perform feature importance.

To implement this model, we ran a grid search so that we could find the optimal hyperparameter combination that would give us the best performing random forest model. Some parameters given to our grid search were that it uses accuracy score to evaluate models and that it perform a 5-fold cross validation, meaning that it will split the data into 5 parts and it will train the model given a combination of hyperparameters with 4 parts of the data, using the last part left to train. For each combination of hyperparameters, this process will occur 5 times so that the model gets tested by all 5 parts of the divided data.

The hyperparameters we selected for random forest for grid search to use were the number of decision trees developed then combines ('n_estimators': [25, 35, 50]), maximum depth side of the tree (‘max_depth’: [5, 10]), minimum number of samples needed to perform a split of a node (‘min_smaple_split’: [2, 5]), minimum number of samples needed for a leaf node (‘min_sample_leaf”: [1, 2, 4]), maximum feature to be considered for ebay split to be determined by square root of the total number of features (‘max_features’: [‘sqrt’]), and whether or not bootstrapping is used in the model (‘bootstrap’: [True, False]).

As a result of the grid search, the random forest grid search outputted the following parameters to result in the best performing model four our data: no bootstrapping (bootstrap = False), maximum depth of 10 (max_depth = 10), minimum sample split of 5 (min_smaple_split = 5), and that the number of decision trees developed to be 25 (n_estimators = 25). The only other hyperparameter present that I had not mentioned is random_state, which controls the random number generator used by the model to ensure that the same results are produced each time the code is run.

3.3.2 Model 2: Gradient Boosting

Our second mode we used was Gradient Boosting. This is an ensemble model that builds typically decision tree models sequentially where the learner, or individual model built in each step of the process, makes corrections to reduce errors based on the ensemble models made so far. This model is able to help create a model for our binary categorical variable that uses log loss as the loss function for our classification gradient boosting model. We decided to run this model with our data because of the model’s ability to adapt, due to its sequential creation of models and practice of working to reduce error each round of models, as well as its ability to provide feature importance as well.

While the grid search parameters were the same as those discussed before, using accuracy score to determine best model and performing a 5-fold cross validation, the hyperparameters used for Gradient Boosting models in the grid search were different. Like with Random Forest, we included values for number of decision trees developed (‘n_estimators’: [25, 30, 35]), but unlink Random Forest, we also included the hyperparameter conditions for learning rate which controls the how fast the model corrects it’s errors (‘learning_rate’: [0.05, 0.1]). While we wanted to add more hyperparameters such as maximum depth, minimum sample split, minimum sample leaf, and some larger number for decision trees developed, at the end were were not able to do so as the grid search would either run for a really long time without finishing or it would crash due to no having enough RAM.

As a result of our grid search, the hyperparameters of our best Gradient Boosting classifier model developed only included the hyperparameter of 35 for our number of decision trees developed with, of course, the inclusion of the random state.

3.3.3 Model 3: K-Nearest Neighbors

Our third model we used was K-Nearest Neighbors. The way that this algorithm works is that given a data point it selects the "K" closest data points and looks at the majority class of those points and assigns that class to the given data point. For example, if a point P has five neighbors and four of those points have class 1 then point P is predicted to be class 1.

The parameters we used to run this model was the number of neighbors that influence the class of each data point. Initially, we set that "K" number to be five by default. This decision is almost completely arbitrary and in order to select the best K value we used cross-validation on a range of reasonable K values (i.e. from four to twenty) and would compare each run of KNN using their respective accuracy scores. As a result of the cross-validation, the K value with the highest accuracy score was K = 18 with an accuracy of 0.713.

The reason why we chose KNN was because when we made pairplot visualizations of our features we didn't notice any linear correlations. As a result, we chose to use KNN due to its ability to work well with datasets that have nonlinear correlations.

3.3.4 Model 4: Logistic Regression

We used Logistic Regression for our classification task, which estimates the probability of a binary outcome by finding a decision boundary between the two classes. The model predicts probabilities and assigns a class based on a threshold (usually 0.5). We trained the model on scaled features to ensure optimal performance and removed any rows with missing values.

To evaluate the model, we used the ROC curve and AUC (Area Under the Curve). The ROC curve shows the model's ability to distinguish between the classes, while the AUC quantifies this performance. A higher AUC indicates better model discrimination.

Logistic Regression was chosen for its effectiveness with linear relationships between features and the outcome. Despite non-linear correlations observed in our pairplots, it serves as a strong baseline for binary classification.

3.3.5 Model 5: CART (Decision Tree)

We used a Decision Tree Regressor (CART) to predict continuous outcomes by recursively splitting data based on feature values. The tree structure is built to minimize variance within each split, and key hyperparameters such as max_depth, min_samples_split, and min_samples_leaf control the tree's growth and complexity.

Decision trees were chosen for their ability to handle both linear and non-linear relationships, and for their interpretability through feature importance and decision paths
We tuned the model using GridSearchCV with 5-fold Stratified Cross-Validation and optimized the F1 score to find the best combination of hyperparameters.

We evaluated the model using Root Mean Squared Error (RMSE), R^2, and feature importance to assess its accuracy and identify influential features.

3.4 Training and Evaluation Methodology

Due to the sheer amount of entries in our dataset, we had to shrink down the amount of data that we used when training our models as, if we didn't, we would have run out of RAM when running cross-validation for our models. The way we originally shrank the data was by using a stratified sample of the data (i.e. a sample of a dataset that retains the distributions of the dataset as a whole). We initially used a sample of 20% of the data and then split 80% of that sample into a training set and 20% of that sample into a testing set. We ultimately abandoned this idea because we wanted our testing set to come from a different sample than the sample used in training. What we wanted to do was make a stratified sample where 40% of the dataset is used for testing and 60% is used for training. However there was an issue with the sizes of certain sets being unequal and the code would crash so, we used a different 20% sample for X_test, y_test, X_train, y_train.

Our cross-validation method was Grid Search Cross-Validation. We used Grid Search CV because it would run tons of different combinations of our models with different hyperparameters. After running Grid Search we would then report what the best hyperparameters were and, at the end, would run the best performing model on the training set with the best model with the best combination of hyperparameters

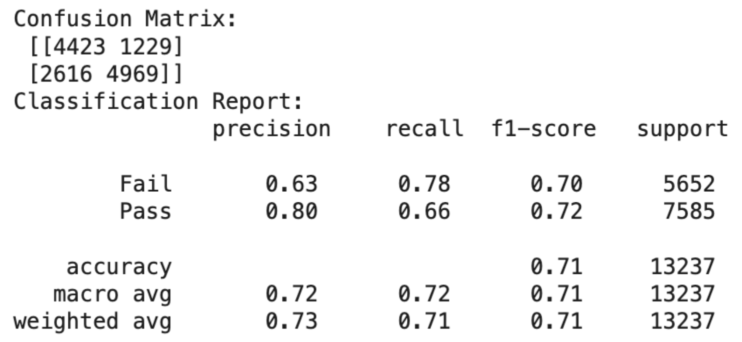
The evaluation metrics we used were R^2, RMSE, Accuracy Scores, Confusion Matrices, and Classification Reports. We used initially R^2 and RMSE when running CART however this was a mistake as, once we determined that we were using classification models, we switched to using classification metrics. The reason why we used Accuracy Scores was because it was just one simple number that we could use to compare models easily. We used Confusion Matrices and Classification Reports so we could see the rates of true positive, false positive, true negative, and false negative predictions.

4 Results

After running all our models we used Accuracy Scores, Classification Matrices, and Classification Reports. In the next section, I will present these results and in the section after I will showcase a visualization of all the models accuracy scores compared against each other.

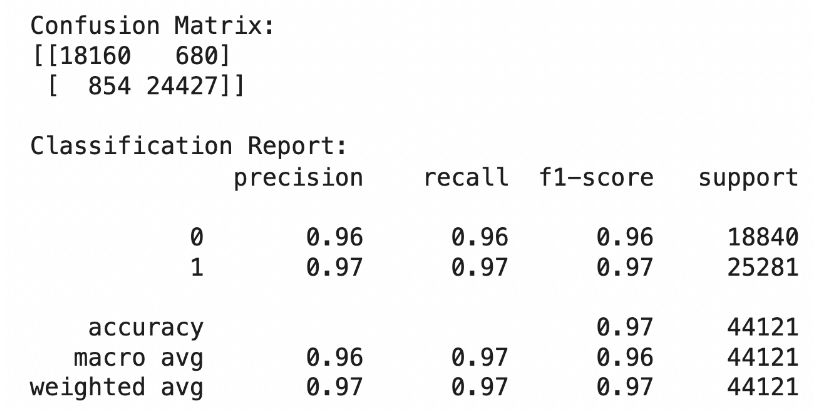
4.1 Model Evaluation

Below is the KNN Confusion Matrix. These results are not that great as only 63% of predicted 0s were correct and 66% of predictions for class 1 are correct. This model was very bad but it wasn't our worst one.



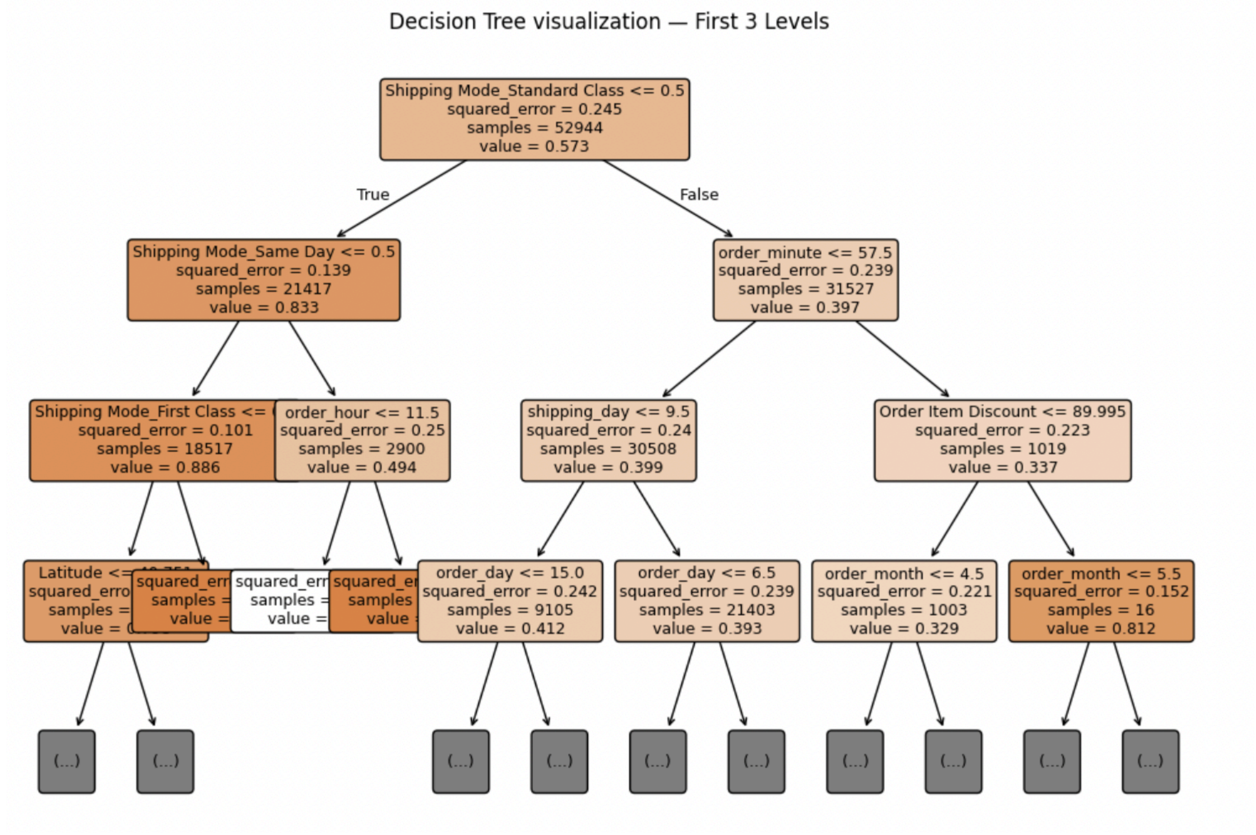
```
Confusion Matrix:
 [[4423 1229]
 [2616 4969]]
Classification Report:
               precision    recall  f1-score   support

        Fail       0.63      0.78      0.70      5652
        Pass       0.80      0.66      0.72      7585

    accuracy                           0.71     13237
   macro avg       0.72      0.72      0.71     13237
weighted avg       0.73      0.71      0.71     13237
```

Below is the Confusion Matrix and Classification Report for our CART model.



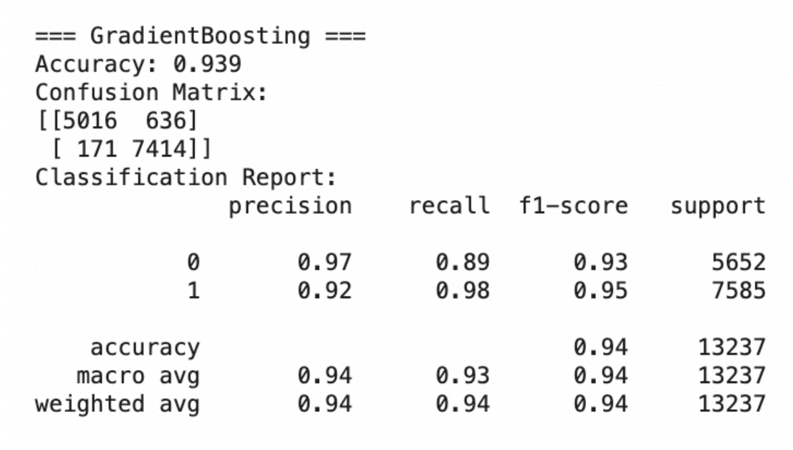
```
Confusion Matrix:
[[18160   680]
 [  854 24427]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     18840
           1       0.97      0.97      0.97     25281

    accuracy                           0.97     44121
   macro avg       0.96      0.97      0.96     44121
weighted avg       0.97      0.97      0.97     44121
```

As you can see the results are very promising as 96% of predicted 0s were correct and were correctly predicted. Additionally 97% of predicted 1s were correct and correctly predicted. Below is the Decision Tree visualization of CART. However we used other metrics to compare models so this isn't used for much.



Decision Tree visualization — First 3 Levels
Shipping Mode_Standard Class <= 0.5 squared_error = 0.245 samples = 52944 value = 0.573
True
False
Shipping Mode_Same Day <= 0.5 squared_error = 0.139 samples = 21417 value = 0.833
order_minute <= 57.5 squared_error = 0.239 samples = 31527 value = 0.397
Shipping Mode_First Class <= squared_error = 0.101 samples = 18517 value = 0.886
order_hour <= 11.5 squared_error = 0.25 samples = 2900 value = 0.494
shipping_day <= 9.5 squared_error = 0.24 samples = 30508 value = 0.399
Order Item Discount <= 89.995 squared_error = 0.223 samples = 1019 value = 0.337
Latitude <= squared_erro samples = value =
squared_err samples = value =
squared_err samples = value =
squared_er samples value
order_day <= 15.0 squared_error = 0.242 samples = 9105 value = 0.412
order_day <= 6.5 squared_error = 0.239 samples = 21403 value = 0.393
order_month <= 4.5 squared_error = 0.221 samples = 1003 value = 0.329
order_month <= 5.5 squared_error = 0.152 samples = 16 value = 0.812
(...)

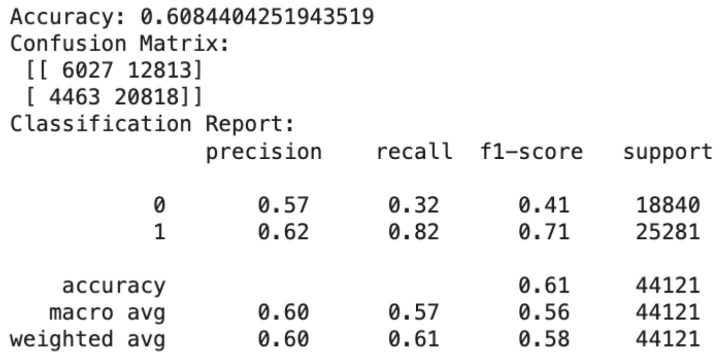
For Gradient Boosting the results are quite promising as the recall and precision values for all the classes. However it still is not as good as the CART results.



```
=== GradientBoosting ===
Accuracy: 0.939
Confusion Matrix:
[[5016  636]
 [ 171 7414]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.89      0.93      5652
           1       0.92      0.98      0.95      7585

    accuracy                           0.94     13237
   macro avg       0.94      0.93      0.94     13237
weighted avg       0.94      0.94      0.94     13237
```

Here are the Results for the Random Forest Classification. As you can see it is still good but not as high as CART's results. Additionally the fact that only 77% of the packages predicted 0 were correct is a bit worrisome.

Below is the confusion matrix for Logistic Regression. These results are very bad. For Example, Only 32% of actual class 0s were detected and 57% of predicted class 0 are actually correct. Overall, this was our worst performing model.

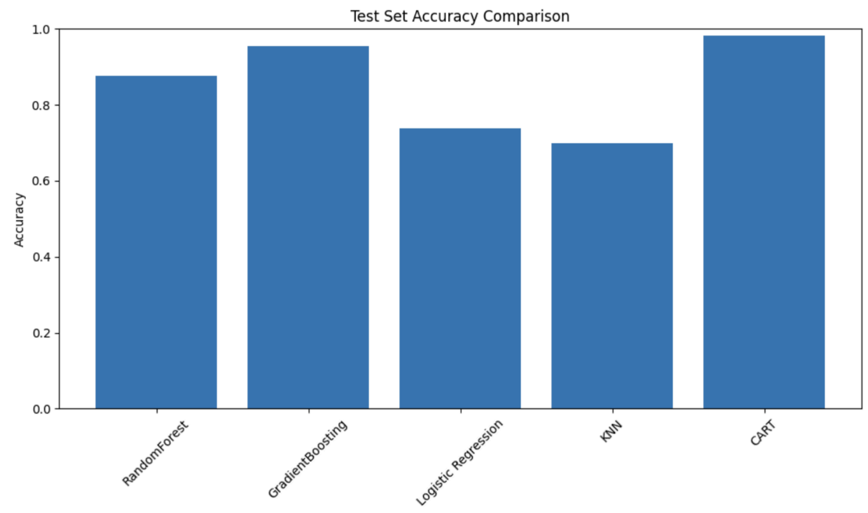


```
Accuracy: 0.6084404251943519
Confusion Matrix:
 [[ 6027 12813]
 [ 4463 20818]]
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.32      0.41     18840
           1       0.62      0.82      0.71     25281

    accuracy                           0.61     44121
   macro avg       0.60      0.57      0.56     44121
weighted avg       0.60      0.61      0.58     44121
```

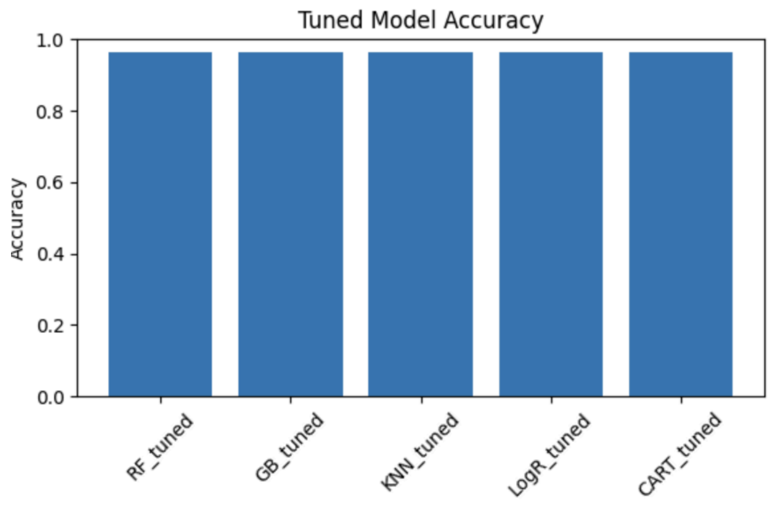
4.2 Comparative Analysis

Below are the results of all our models' accuracy scores before we tuned them. As you can see CART was the best performing one. Gradient Boosting was close to that and Logistic Regression, and KNN all had very similar accuracy scores. It looks like the reason why CART is the best one is because of the nonlinearity regarding the relationships between our features. This nonlinearity may also be the reason why Logistic Regression was not as good as some of the other models. KNN may have had a low score due to irrelevant features influencing the model. Gradient Boosting is able to combine many trees to improve itself over iterations which may be why it also performed high. While Random Forest is able to reduce overfitting it can struggle with irrelevant features too.

```
weighted avg       0.98      0.98      0.98     22061
```



After tuning all the models our results were that the accuracy scores ended up all being the same as shown here:



This could be because our Grid Search parameters were too narrow in their ranges. This could be because we are testing on the same test set before and after tuning. Also it is possible that the models have hit a ceiling as to how well the models can perform.

5 Discussion

5.1 Connections to Prior Work

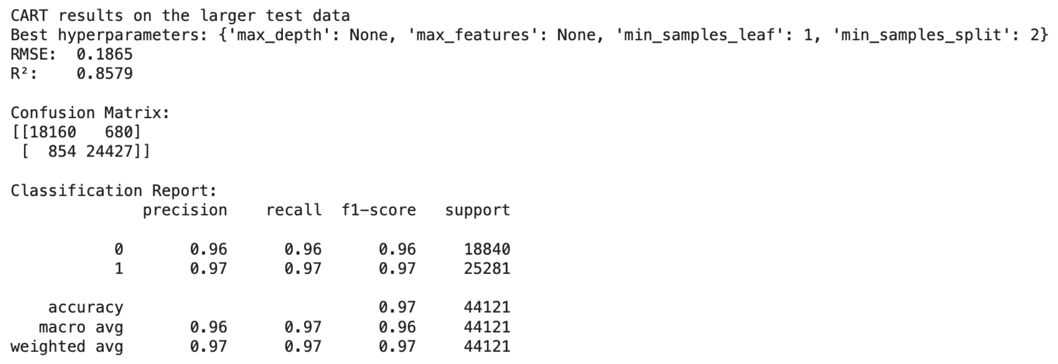
When comparing our work to existing literature, such as the article mentioned before by Wyrembek, we noticed that something more unique about our research was that we also looked into the feature importance hierarchy of the features we ran for our x-variables. By doing this, we have gained some more valuable insights that Wyrembek and his team had not. We were able to uncover the features that are contributing most to whether or not a shipment will be late. In this way, the company can use their understanding of the features in relation to their supply chain and make informed decisions about how they can improve the performance of those features that seem to be most important if it is within their reach. For example, order day seemed to be the most important feature in the best-performing model, which happened to be our best grid search model for CART. With this information, the company can now look into what order days seem to result in the most late arrivals of orders and implement measures, such as having more hands on board or having more trucks shipping out orders that day, to improve their supply chain.

5.2 Limitations

The main limitation we faced had to do with running our code. We experienced the most difficulty with running a grid search with all the desired hyperparameters we wanted to test. What would occur with our running of grid search was that it would take hours to run to the point that we decided to end the running of the code due to its ineffectiveness, or it would crash either immediately or after some time of running said code with a message that our RAM was running out. As a result, there were some hyperparameters we could not run or that we had to decrease the numeric value for, despite our desire to experiment with them. Additionally, we came into the project wanting to run XGBoost as we were interested to see how this model, which is supposed to be able to run well with larger datasets, would affect our research results when comparing all the models. Unfortunately, when we tried to run XGBoost, the code would continue to crash due to RAM running out despite our continual editing of hyperparameters.

6 Conclusion & Future Work

In conclusion, CART was the best performing model as seen in the visuals provided. Unfortunately we were not able to run CART with the whole dataset but, through extensive Grid Search, we were able to confidently say that our CART model is able to classify whether or not a package will arrive late.



```
CART results on the larger test data
Best hyperparameters: {'max_depth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
RMSE:  0.1865
R²:    0.8579

Confusion Matrix:
[[18160   680]
 [  854 24427]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     18840
           1       0.97      0.97      0.97     25281

    accuracy                           0.97     44121
   macro avg       0.96      0.97      0.96     44121
weighted avg       0.97      0.97      0.97     44121
```

Fig 6.1 CART metrics

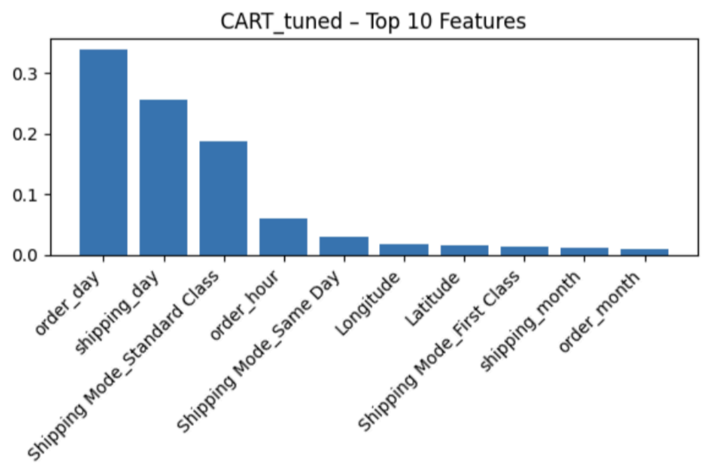


Fig 6.2 CART feature importances

Order_day and shipping_day seem to be the most important features and that is also interesting as it tells us that the day the order was placed and the day the order was shipped really plays an important role in determining whether an order would arrive late. Apart from that, shipping modes also influence the arrival of a package.

Future considerations:

- Interpreting feature importance:
  Right now, we only know what factors influence the prediction. We do not know if these variables influence the target positively or negatively, and for what values.
- Fix boosting:
  We also tried implementing boosting models and for some reason, kept running into errors or meaningless results. Fixing this could tell us stuff about our data that we didn't know.
- Understanding hyperparameters:
  Our current hyperparameters seem to be super fine tuned and we should look into ways that make more general data classifications. This could help us get more success with boosting.

References

Wyrembek, Mateusz, et al. "Causal Machine Learning for Supply Chain Risk Prediction and Intervention Planning." International Journal of Production Research, Jan. 2025, pp. 1–20. EBSCOhost, https://doi.org/10.1080/00207543.2025.2458121.