**Data Science**

**Final Project Documentation**

**Contributors**

**Kevin Chou**

**Po-Yu Huang**

**Table of Contents**

[1. Overall Process 2](#_Toc2130256368)

[2. Data and Cleaning 3](#_Toc1342402614)

[3. Feature Extraction 7](#_Toc1023188991)

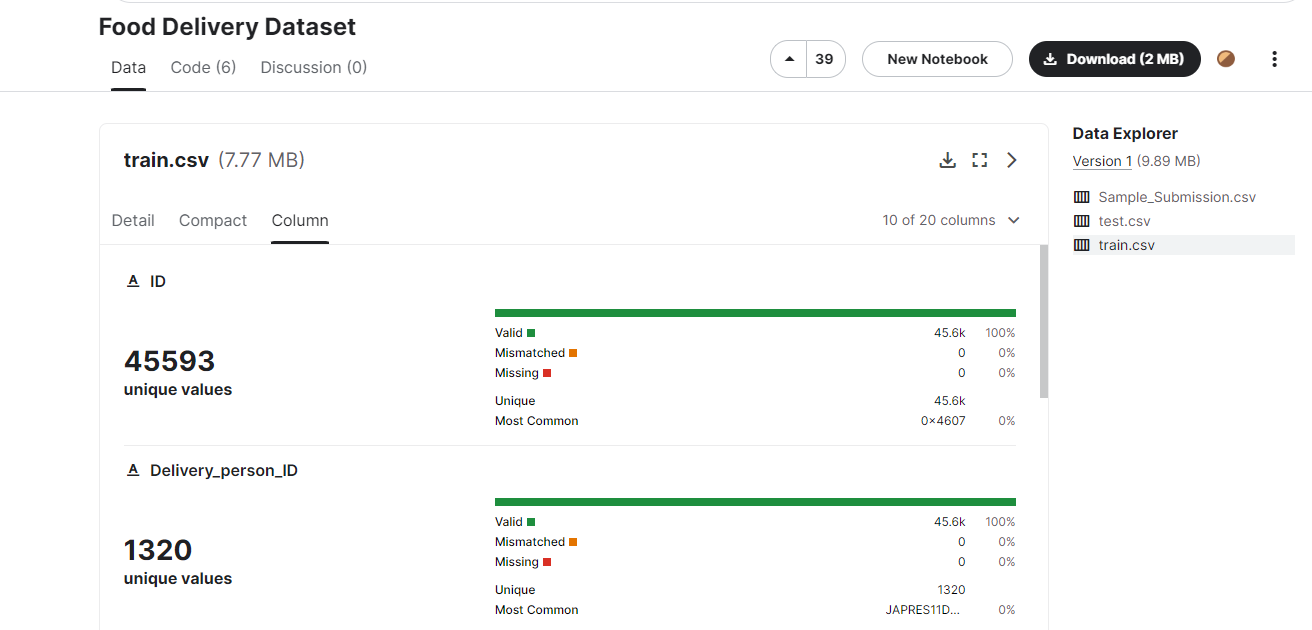
[4. Modeling and Tuning 9](#_Toc603412754)

[5. Analyze 11](#_Toc489058057)

[6. Conclusion 14](#_Toc105893582)

1. Overall Process

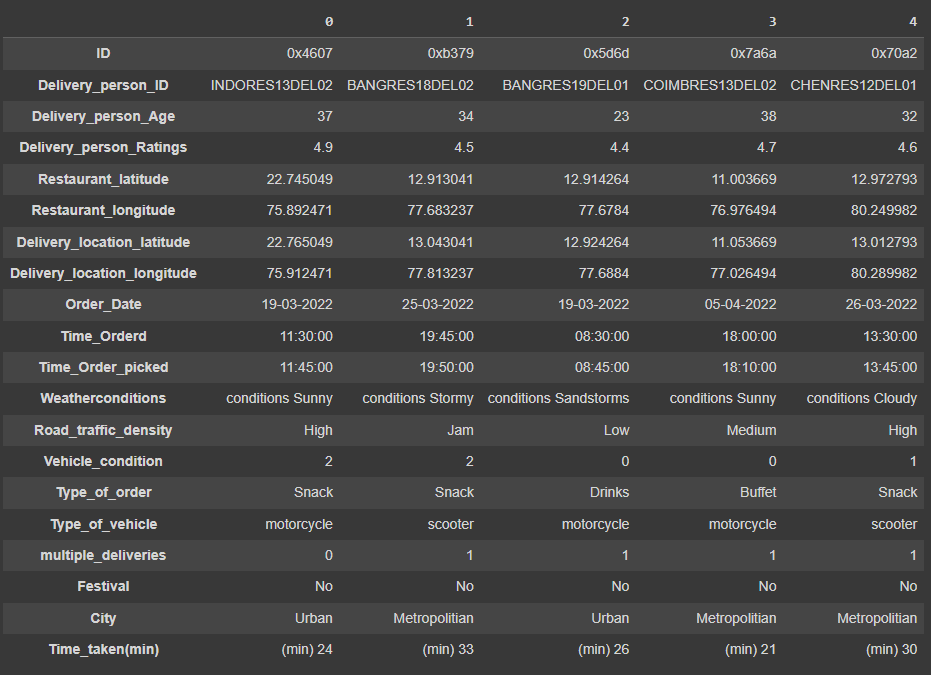
In this project, we are using the Food Delivery Dataset from the Kaggle website. This dataset keeps records of food delivery order details from restaurants and stores to customers. This data is collected from the ordering website or mobile app of a restaurant or grocer and food ordering companies. Delivery items include entrees, sides, drinks, desserts, or grocery items, and whether delivered in boxes or bags. This dataset also includes the type of vehicles, type of weather when delivering the order, size of city and the specific location for both restaurant and destination.



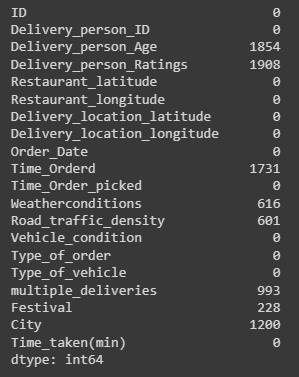
This dataset contains 45593 unique rows as our training samples and 11399 unique rows for our testing data. However, the results of the testing data were not open to us, so our solution is to split the training data into 66% of the training data and 33% of the testing data. Below is a figure of our data set on the Kaggle website. Our goal for this project is to predict the delivery time for this dataset and try to hit or surpass the highest accuracy we could find on Kaggle which is 82%.

1. Data and Cleaning

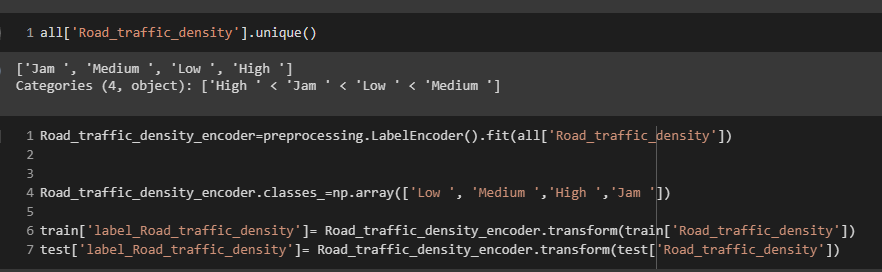
Frist of all, we go through all the columns we have in this database and these are the columns included “ID”, “Delivery\_person\_ID “, “Delivery\_person\_Age “, “Delivery\_person\_Ratings”, “Restaurant\_latitude”, “Restaurant\_longitude”, “Delivery\_location\_latitude”, “Delivery\_location\_longitude”, “Order\_Date”, “Time\_Orderd”, “Time\_Order\_picked”, “Weatherconditions”, “Road\_traffic\_density”, “Vehicle\_condition”, “Type\_of\_order”, “Type\_of\_vehicle 0”, “multiple\_deliveries”, “Festival”, “City”, “Time\_taken(min).”



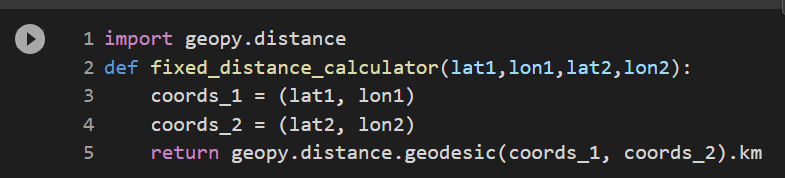
Among all these columns, there are some columns that we could use directly but some of them need to be cleaned and encoded. Thus, for the data cleaning, we first replace all the “NaN” values with np.nan and drop it. At this stage, we still have 41368 entries in this data.



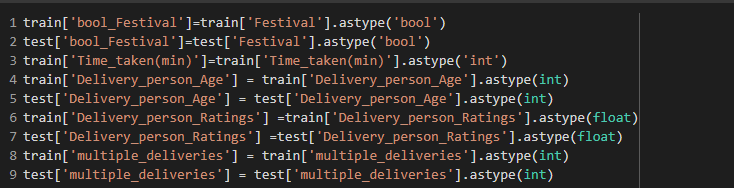
Then, we label encoded the columns “Weatherconditions”, “Road\_traffic\_density”, “Vehicle\_condition”, “Type\_of\_order”, “Type\_of\_vehicle”, “City”. Take “Road\_traffic\_density” as an example, we go through the proper order we need to assign it. “Low” density will be labeled to 0 and “Jam” will be labeled to 5. This could let the visualization be much clearer when the color is well organized using SHAP (SHapley Additive exPlanations).



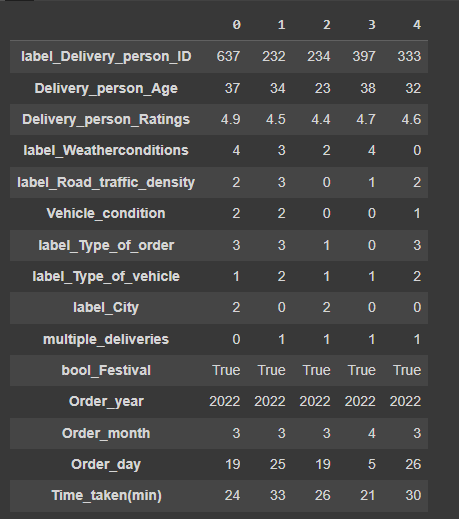
Then, we calculated the distance between two locations to generate the distances between the restaurant and the destination, and we tried two methods. The first one is directly using Pythagorean Theorem to calculate the distance between two places. However, we realized that the distance between two longitudes will get shorter as the latitude is not equatorial. The second method, which is also we are using is, included the library below called “geopy.distance.” This library could return the accurate distances between two places. Though the difference between two places is quite close, we still want our distance to be as accurate as it could be.



For the last step of data cleaning, we check all the data type to be transformed to np array. Like, transforming it to Boolean type, Integer type and Float type.

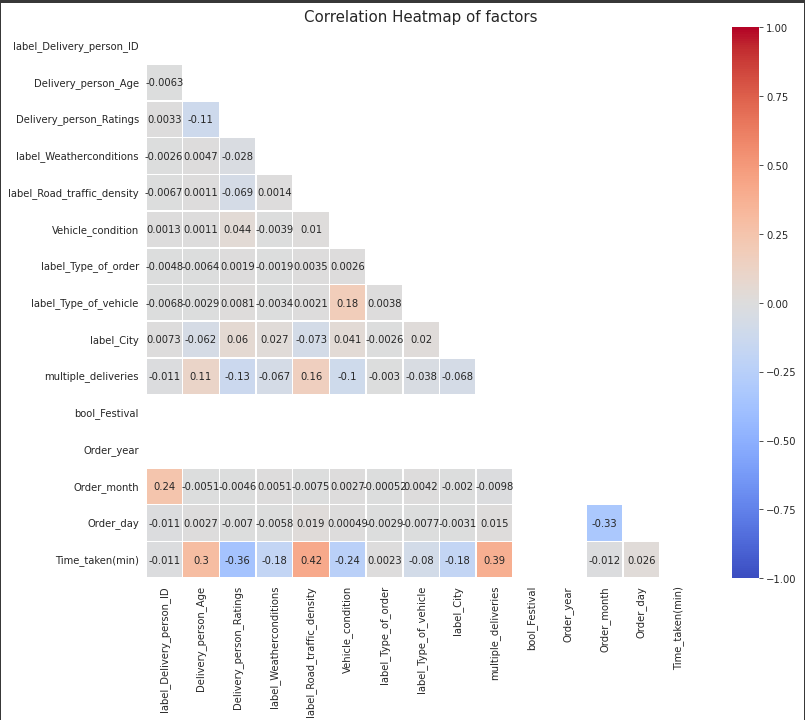


After all the steps, by the data set looks like the figure below.

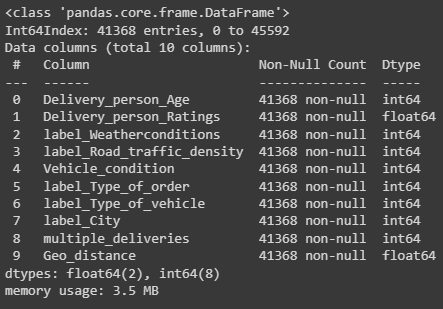


1. Feature Extraction

To find out which column does have more correlation with the delivery time. We add on our hyper-parameter which is geographic distance and generate correlation heatmaps to better visualize the correlation. From the heatmap, it is evident that delivery person age, delivery person ratings, label traffic density and multiple deliveries are the parameters that had compare high correlation to the delivery time.



We tried several combinations of our dataset, and we finally pick 'Delivery\_person\_Age', 'Delivery\_person\_Ratings', 'label\_Weatherconditions', 'label\_Road\_traffic\_density', 'Vehicle\_condition', 'label\_Type\_of\_order', 'label\_Type\_of\_vehicle', 'label\_City', 'multiple\_deliveries' and'Geo\_distance' as our final training columns. Then, we saved it into the x and split into “X\_train” and “X\_test.” The result dataset had also been split into “Y\_train” and “Y\_test.” All these datasets had been stored in the folder “Data.”



1. Modeling and Tuning

We loaded the training data and testing data out from the folder “Data”. Then, we tried mainly two models which are the linear regression model and the decision tree model. Below are the models that we are testing in this project.

Graphical user interface, application

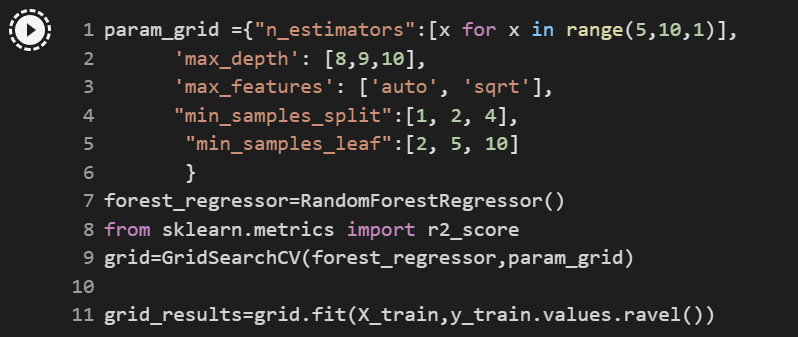
Description automatically generated

We used a loop to go through all the models to get the approximate accuracy from all the models we picked, and we had a conclusion that regression tree had averaging much better accuracy than the linear regression. Then, we will only be focusing on tuning the decision tree models. First, we step through the “max\_depth” for the tree to find out which depth will best fit our dataset. From the figure below, we could see that the best “max\_depth” for this data is around 9 to 10. Then, we ran through all the tree regression models that we picked, Random Forest Regressor, XGB Regressor and Decision Tree Classifier. Among all the models Random Forest Regressor had the best was the model that performed the best one.

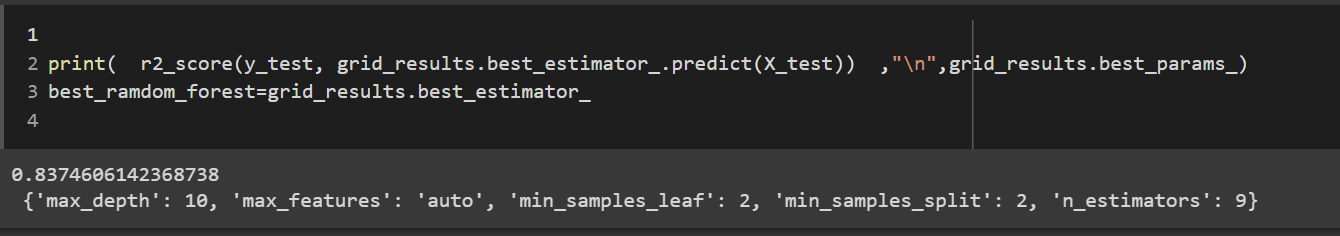
Chart, line chart

Description automatically generated

Secondly, we tried using “GridSearchCV” to have deeper research on all the parameters that we could tune. The stepover is also stored in the figure below.



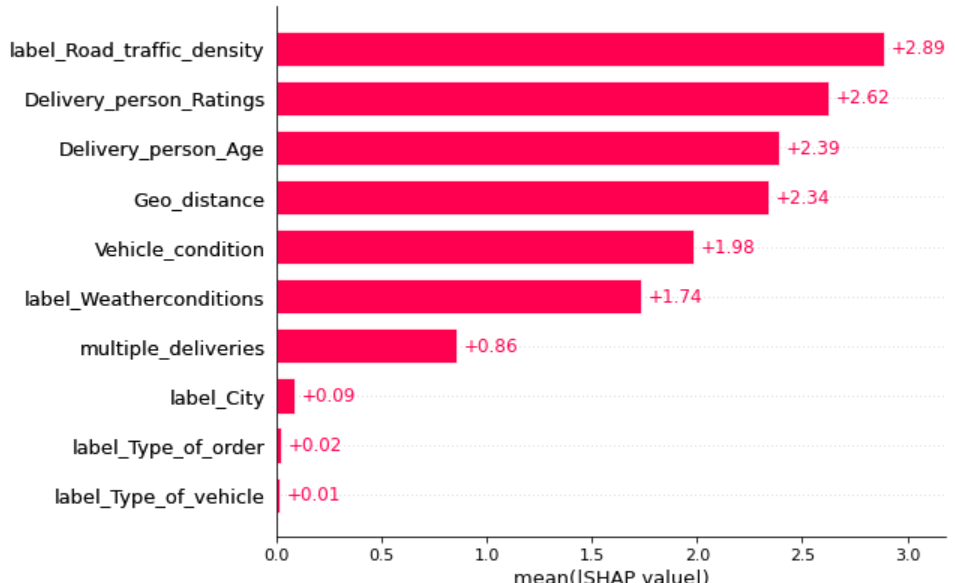
After all the process we had done, we had 83.7% accuracy for predicting our split test data. The final parameters for Random Forest Regression model are: ‘max\_depth’ is 10, 'max\_features' is auto, 'min\_samples\_leaf' is 2, 'min\_samples\_split' is 2 and 'n\_estimators' is 9. The best accuracy we saw on Kaggle is approximately 82% and we managed to push a little bit higher than what it had on it. However, this is only tested by our own testing data, so we do not know how our submitted test results will perform with our model.



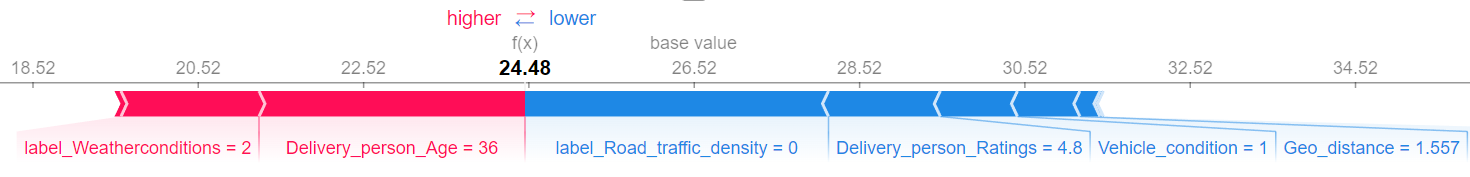
1. Analyze

After testing our best model, Random Forest Regression, we use SHAP (SHapley Additive exPlanations) to analyze our model. SHAP had clean figures for visualizing our data on both focusing on a single test case and on all the test cases together.

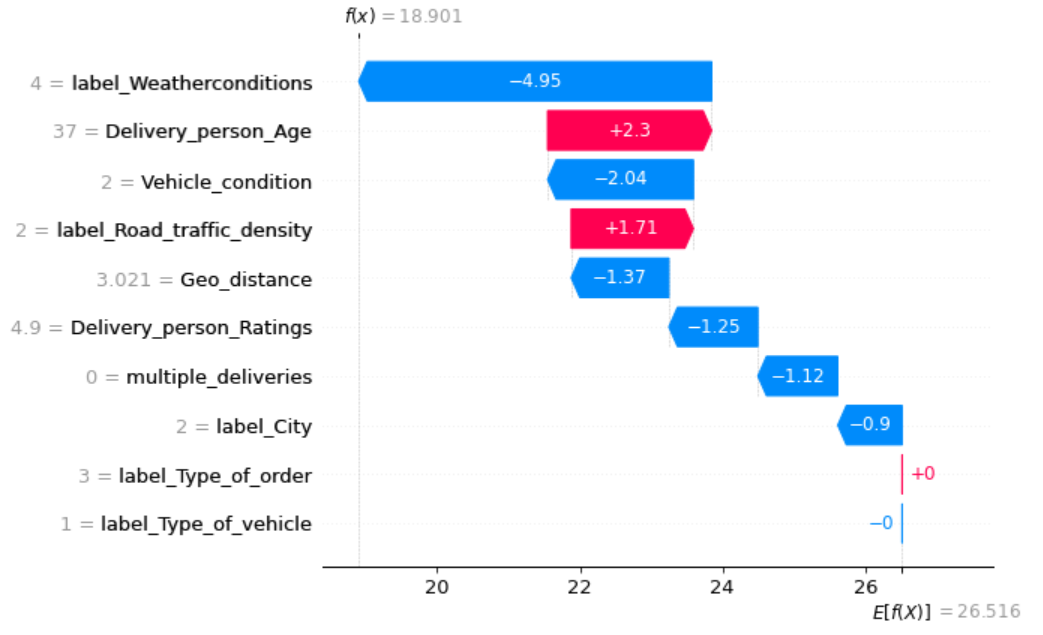
First of all , we want to go through is the summary plot. This plot gives us an idea about how important each column will affect the result. For example, the most vital features for this dataset are the “label\_Road\_traffic\_density,” “Delivery\_person\_Ratings” and “Delivery\_person\_Age” which almost matches to our heatmap that we generated before.



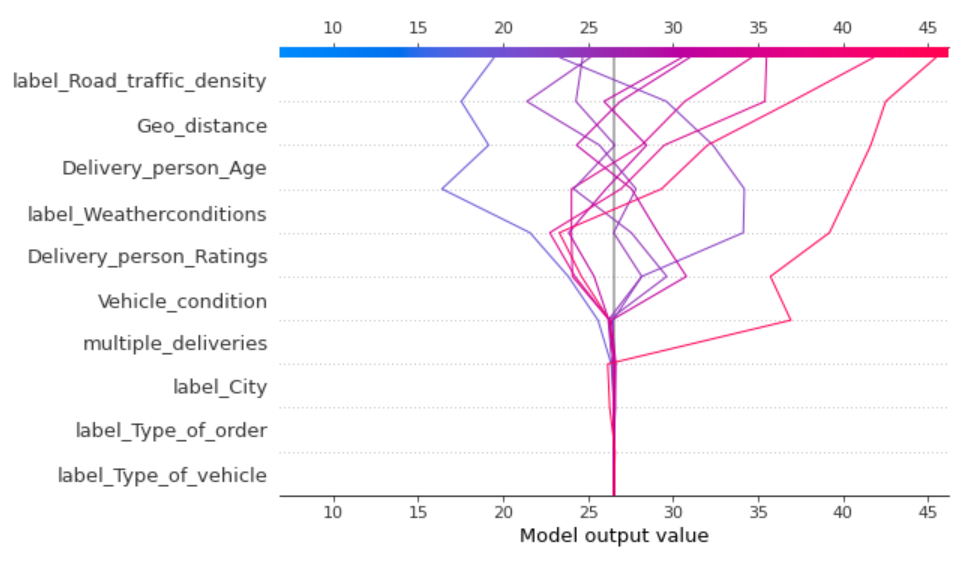
In SHAP, there are two different categories of plot we can use, one is for going through a single test case and another one is for comparing all the parameters. In this report, we are going to show two kinds of plots to visualize the single test case, which are Force plot and Waterfall plot. These two plots are similar yet have different ways to display all the influence of the parameters. Below is the Force plot, this plot clearly shows what parameters added up the estimation for delivery time for prediction and what parameters decreased the time. The number 24.48 is the final predicted results. The red bars are the factors that costed an increase to the predicted time, and the blue bars are the factors that decreased the time.



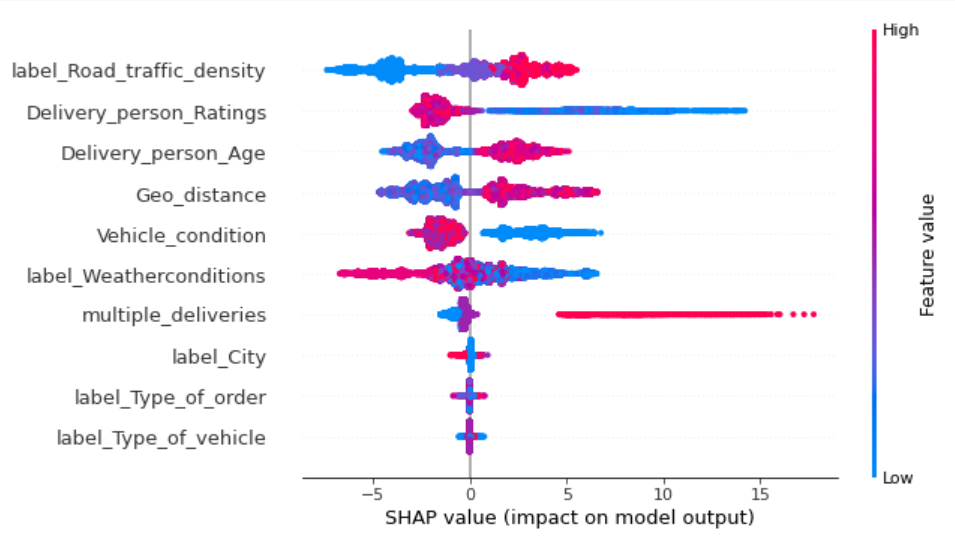
The plot below is the Waterfall plot. This plot is kind of mixture of summary plot and Force plot for a single test case. This plot clearly shows the predicted time after adding up every parameter. Additionally, this plot also sorted by the importance of the parameter, so we can see that the weather condition is the most important factor in this test case.



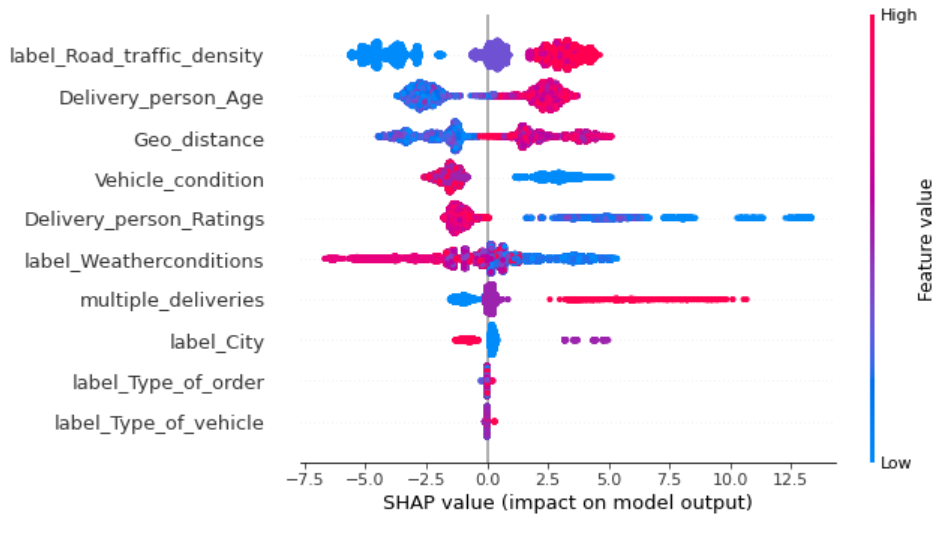
The Decision plot displays all the decisions that were made by the Random Forest Tree Regression. We can see that minimum and maximum prediction of our model are 19 and 46. After checking the longitude and latitude from this dataset we found out the delivery vehicles are all in motorcycles, scooters, electric scooters, or bicycles.



The last plot we are discussing is called the Beewarms plot, which shows all the distribution for the parameters we had in a single plot. This plot provides a better look of all the correlations of the parameters with our result. This plot also shows the difference between the distributions. For example, we will know that once the columns “Delivery\_person\_Ratings,” exceeds the average, the rating will not affect the delivery of time. However, for the “Delivery\_person\_Age”, the difference of ages will affect the final delivery time. Lastly, I would like to talk about is the “vehicle\_conditions,” this distribution also gives us a clear idea that only the vehicle conditions below will really affect the delivery time; otherwise, there are no significant differences for the delivery time estimation.



This is for Random Forest Regression.



XGBoost Regression. (For comparison)

1. Conclusion

After conducting several experiments and training multiple models, we managed to hit an accuracy of 83.7% from our testing data. Though this was not a significant improvement in the accuracy of this dataset, it really gave us a clear idea of how the parameters will affect the final prediction and how the decisions made will reflect on the final estimated time. In this project, our biggest take away was using the game theoretic approach to explain the output of a regression which is SHAP (SHapley Additive exPlanations.) It gave us an overview of how the prediction was made from the given values and how does the distribution looks compare to other parameters.