Predicting Time of Arrival for Food Delivery Service

Daniel Carpenter, Sonaxy Mohanty, Zachary Knepp

Daniel Carpenter: danielcarpenter@ou.edu

Sonaxy Mohanty: sonaxy.mohanty@ou.edu

Zachary Knepp: Zachary.M.Knepp-1@ou.edu

University Of Oklahoma

Data Science and Analytics Institute

DSA 5103-995 | Intelligent Data Analytics

Fall 2022 | Charles Nicholson

Table of Contents

Executive Summary

In today's society, people are constantly on the go. People already utilize delivery services in all types of industries, so food delivery services offer one more convenience for a busy schedule. Food delivery services is an on-demand food delivery platform that allows customers to order food and beverages from restaurants and get the food delivered in a stipulated time. The process of food delivery is then executed by delivery agents hired by the company, whose main goal is delivering food on time, keeping all the hindrances in mind that may affect the delivery time. To aid these delivery persons, the company needs to improve its system that calculates the time taken to deliver with the help of intelligent software that can predict the time of arrival given known and unknown conditions that a delivery person can face, rather than relying on some fixed formula. Food delivery services aim to provide accurate time estimates to the customer for food delivery thus ensuring the retention of its customers. The data set used for this project is taken from the Kaggle website which consists of nineteen columns describing different characteristics of the delivery driver and the conditions that they face while driving to the destination. The target or predicted variable is the minutes taken to deliver the food. The data set consists of sixty-five thousand text files which are cleaned and combined to provide a framework for effective predictive modeling. Within this data, some missing values needed to be imputed. The skewed variables are normalized to provide a better model for the training and test data. Feature selection is done to simplify the model without impacting much of the performance. Several models are developed to train on the dataset to provide the best predicting model with minimum prediction error. Moreover, since the test data set doesn't have a target column to compare the predicted values with, it remains a future scope of work.

Problem Background

Problem Description

This analysis aims to predict estimated delivery times for a food delivery service. A firm can give the option for delivering the food to a customer's house; using current technology, the company may give the consumer an estimated time of arrival to help manage their expectations, which could lead to enhanced retention of customers for future orders. Companies like DoorDash or GrubHub give customers an estimated time of delivery for food and beverage orders. Most consumers may expect a few conditions to affect the time to deliver, but there may be many circumstances that impact the delivery time. For example, if the algorithm knows that there is a crash impeding traffic between the major routes of the customer and the delivery service, then that may impact the transport time. Additionally, severe weather may delay the ability of a driver to deliver the food to the destination. Overall, providing accurate estimates to the customer will help manage expectations, which may lead to retained customers.

Data Description

The data set for this problem consists of nineteen columns describing the characteristics of the delivery driver and the conditions that they face while driving to the destination. The target or predicted variable is the minutes taken to deliver the food. Location data such as the latitude and longitude of both the source restaurant and delivery location are included. The data offers details such as the time the that the customer placed the order and the time that the delivery service picked it up. Additionally, the data describes the type of order placed. Other characteristics about the city or known festivities occurring during the time of delivery are

included. Finally, the remaining data reveals observed weather, traffic, and vehicle conditions. Altogether, this information helps create a model to predict the time to deliver the food or drinks.

Data Dictionary

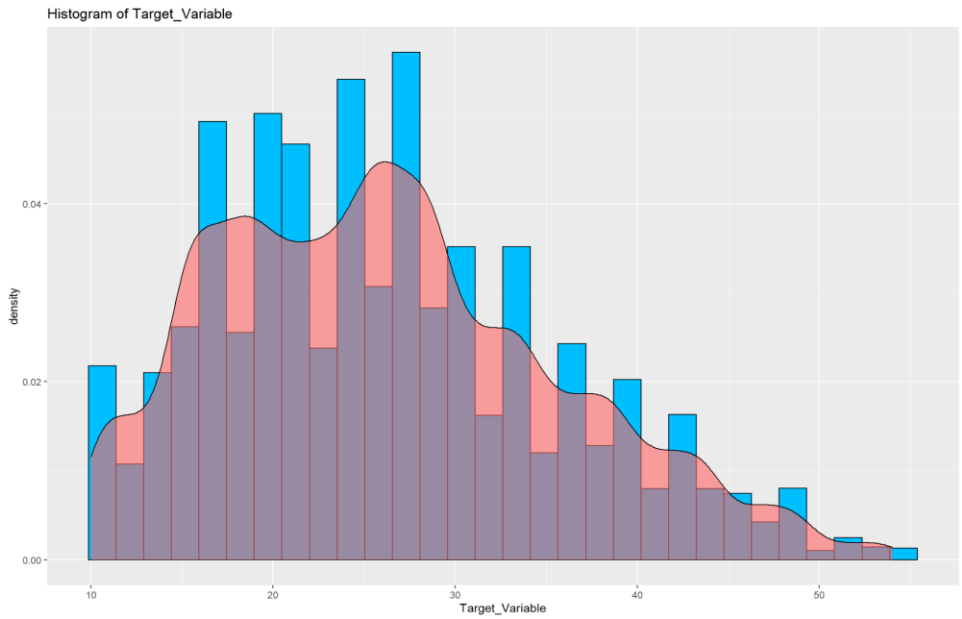
Variable name	Description
ID	Unique identification number
Delivery person ID	Unique identification number for delivery person
Delivery person age	Age of delivery person
Delivery person	Ration of delivery person
Restaurant latitude	Latitude of the restaurant.
Restaurant longitude	Longitude of the restaurant.
Delivery location latitude	Latitude of the Delivery location.
Delivery location longitude	Longitude of the Delivery location.
Order Date	The date when the order was placed.
Time Ordered	Time when the order was placed.
Time Order picked	Time when the order was picked from the restaurant.
Weather conditions	The weather conditions (Windy, Sunny, Cloudy, Stormy, Fog, Sandstorms, etc)
Road traffic density	Road traffic density (Jam, High, Medium and Low)
Vehicle condition	The condition of the vehicle. (Smooth, good or average)
Type of order	The type of order (Snack, Meal, Buffet, Drinks, etc)
Type of vehicle	The type of vehicle one is using (motorbike, bicycle etc.)
Multiple deliveries	The number of orders to be delivered in one attempt
Festival	Represents whether day is festive or not
City	Represents the city
Time Taken	The time taken by the delivery person to deliver the order. [TARGET]

Chart references https://www.kaggle.com/datasets/radadiyamohit/time-taken-by-delivery-person

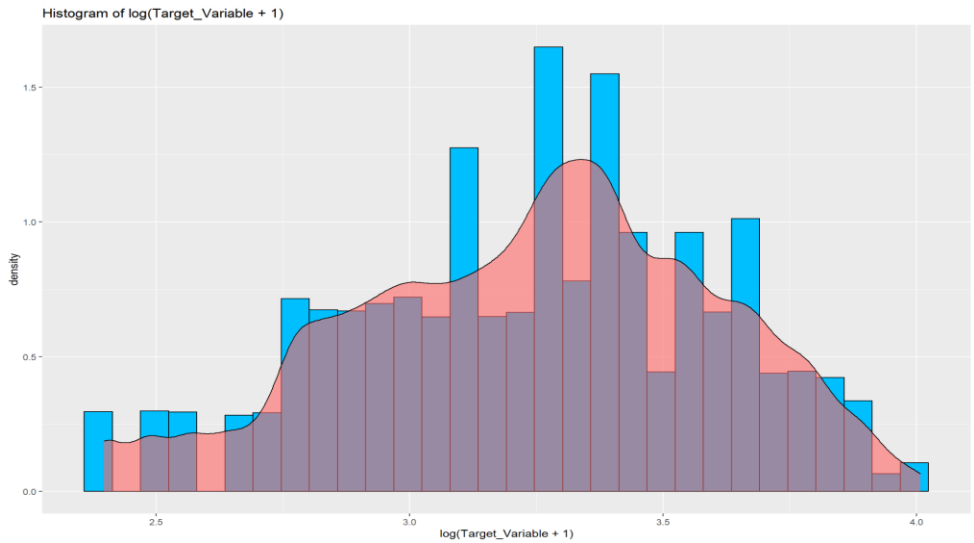
Exploratory Data Analysis

Analysis 1: Exploring the Target Variable

A histogram was drawn for the target variable to show the distribution. According to the histogram, the data appears to be right skewed.

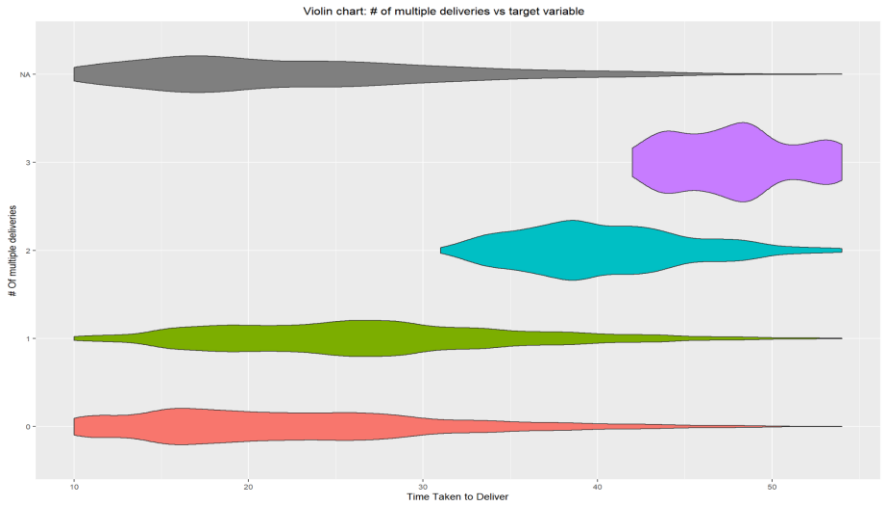


In an attempt to fix the right skewness, and to make the data look more normal, a logarithmic transformation is performed. The results are as follows:

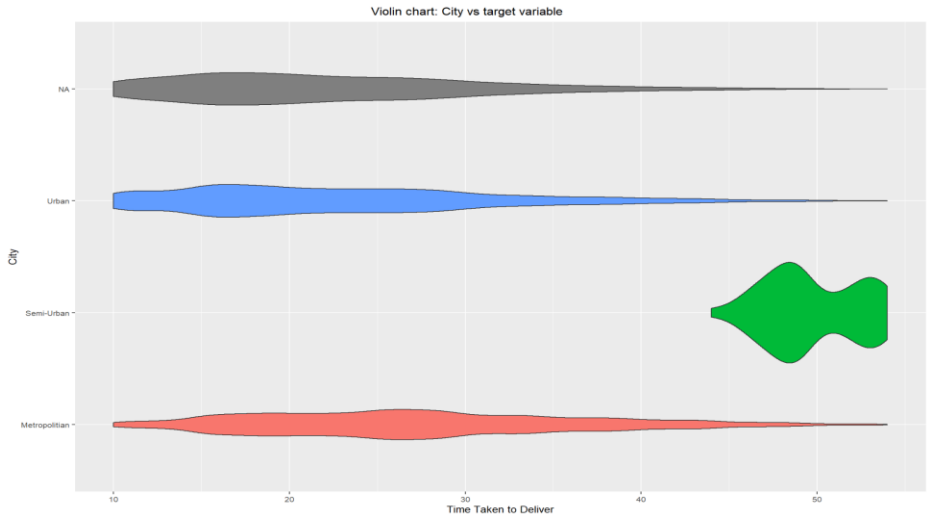


Analysis 2: Visualizations of interactions between target variable and factor variables

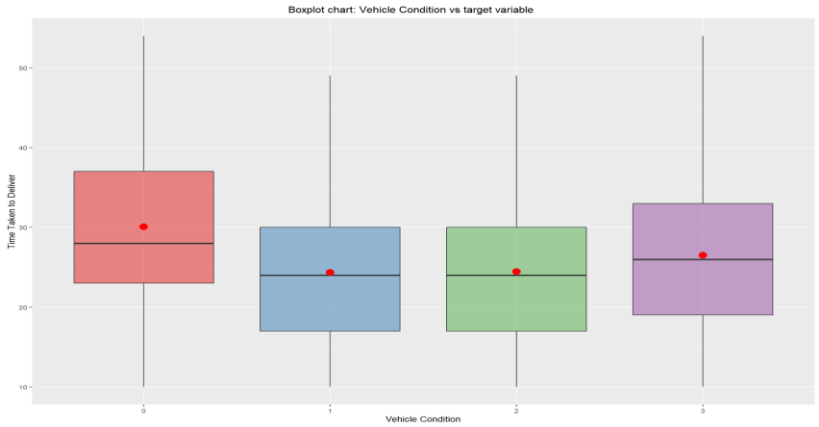
The violin chart of "*# of multiple deliveries vs target variable*" shows that the more deliveries the delivery drivers make, the higher the target values they achieve, i.e., the minutes taken for the delivery of food will increase if there are a greater number of deliveries assigned to the delivery drivers.



The Violin chart of "*City vs target variable*" indicates that Semi-Urban areas have the highest amount of the target value, i.e., the time taken to deliver food or drinks in semi-urban areas is the longest. It is a slim distribution, meaning it does not vary as much as the other factors.



The Boxplot of "*Vehicle Condition vs Target Variable*" shows that vehicle conditions 0 and 3 have a higher target value.



Analysis 3: Visualizations of interactions between Target variable and numeric variables

The first scatterplot drawn shows the relationship between multiple deliveries and the increasing amount of the target variable. Delivery drivers who make multiple deliveries tend to have more time taken to deliver food or drinks. The scatterplot also provides an overview of how the ratings of delivery persons are affected concerning the delivery time. Most of the deliveries take between 15 and 30 minutes. Additionally, most of the delivery persons have a high rating of more than 4.

Another scatterplot was drawn with the city being used as the determination for color. This visual shows semi-urban having the highest amount of time taken for delivery, followed by urban and metropolitan areas.

Methodology

Data Cleansing

Overall, the data combines and cleans sixty-five thousand text files containing food delivery data to provide a framework for effective predictive modeling.

Missing Data

To handle missing data, the model imputed both factor and numeric data separately. To impute factor data, the algorithm called K-Nearest Neighbors uses a distance measure to infer missing values based on the five closest “neighbors” in the data. To impute the numeric data, the model leverages predictive mean matching, which creates a hybrid approach between regression-based imputation, while limiting the range of variation to the data in the training set. This

method maintains the variation in the training data while limiting the production of outlying data. Choosing these methods for factor and numeric data allows for completeness in the data so that the model can maximize all possible data for enhanced prediction.

Skews and Outliers

The model normalizes three skewed variables to better model the training and test data. Implementing the Box-Cox function for the Time_Ordered, Time_Order_Picked, and the Target_Variable helps normalize these distributions. Note that the Target_Variable experienced outliers, but the normalization transformation results in no outliers present. Please note that the location data relating to latitude and longitude have skewed distributions; however, since negative values persist, normalizing these distributions with the Box-Cox method is impossible, so the model does not transform these skewed columns. Overall, these transformations help enhance the model predictivity.

Factors

Since the factor data contains few unique values, the model does not "factor lump" the data. Factor lumping is the process of reducing the number of unique factors within a certain variable. If there were many unique values in related variables, then the model would factor lump to help fit the models more efficiently.

Modeling Choices

The models that are used for this dataset are as follows:

- Ordinary Least Squares (OLS)
- Multivariate Adaptive Regression Splines (MARS)
- Elastic Net
- Principal Component Regression (PCR)

- Decision Trees using Classification and Regression Trees (CART) algorithm
- Random Forests
- Gradient Boosting

Using the OLS model, a stepwise variable selection process is carried out to simplify the model without impacting much of the performance. All the models use the same set of predictors on the training data set to determine the best model using Root Mean Squared Error (RMSE) and R^2 statistical measures.

Model Validation Methods

Except for OLS and PCR modeling approach, all other models are created for the data set using a repeated 5-fold cross-validation technique, repeated once for model validation and hyper tuning purposes.

Results

Model Performance Summary

Model	Function or Package	Hyperparameters	$RMSE$	$Adj.R^2$
Random Forest	rf	mtry=6	0.649	0.464
Gradient Boost	xgbTree	max_depth = 3 eta = 0.3 nrounds = 150	0.669	0.6751
MARS	caret and earth	Degree = 1 nprune = 23	0.761	0.580
OLS	lm	N/A	0.786	0.551
Elastic Net	caret and elasticnet	Alpha = 0.4 Lambda = 0.0011129	0.787	0.551
PCR	pcr	ncomp = 15	0.787	0.551
Decision Tree	rpart	cp = 0.0065568	0.840	0.487

The Random Forest model with six random variables selected at each split gives the minimum out-of-bag error rate. This model outperforms other models based on the RMSE value of 0.65 which explains 46% of the variance in the target variable using the predictors of the data.

Conclusion

The purpose of this analysis is to provide more accurate predictions for estimating delivery times. Companies that provide delivery services like Amazon and DoorDash often use an estimated delivery time so their customers can know when to expect their products. Accurate prediction times may lead to the retention of customers. Using various machine learning techniques, the analysis aimed to predict minutes taken to deliver food using nineteen different variables. Data cleaning, imputing missing factors and numerical values using K-Nearest Neighbors and predictive-mean matching, and normalizing the highly skewed distributions. Using RMSE and Adjusted R^2 as performance metrics, seven different models were developed for the data set. Out of the different tree structures that are modeled, a single decision tree achieved the highest R^2 of 0.840, and the random forest achieved the lowest RMSE of 0.6489.

One of the major limitations encountered is getting access to the test data set to evaluate the models' performance on this dataset. The test dataset from Kaggle did not have the target variable values in the file, therefore the comparison of the predicted results of the test set to its actual values cannot be performed. Another limitation encountered is the lack of data to develop a neural network. A neural network was initially run on the data, however, only two variables qualified to be used as predictors. There were issues with the neural network not being able to reach target parameters. When the parameter expectations were reduced, the neural network was able to fit the data, but there was no test dataset to calculate the model performance.

References

[1] https://www.kaggle.com/datasets/radadiyamohit/time-taken-by-delivery-person

[2] http://www.sthda.com/english/articles/35-statistical-machine-learning-essentials/139-gradient-boosting-essentials-in-r-using-xgboost/

[3] https://medium.com/codex/data-science-tutorials-training-a-decision-tree-using-r-d6266936d86

[4] https://builtin.com/machine-learning/food-delivery-time-prediction

Appendix

Model Code

OLS:

```
# Fit Initial model

Ols <- lm(Target_Variable ~ ., data = df.clean)

# Use stepAIC to discover better model

fit.ols.stepAIC <- stepAIC(fit.ols, direction = "both")

# Final model

fit.ols <- lm(Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
              Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
              Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
              Festival + City + Time_Order_Picked,
            data = df.clean)

# Get the RMSE and R Squared of the model

ols.rmse    <- rmse(actual=df.clean$Target_Variable, predicted=fit.ols$fitted.values)

ols.summary <- summary(fit.ols)


# Key diagnostics

keyDiagnostics.ols <- data.frame(Model    = 'OLS',
                         Notes    = 'lm',
                         Hyperparameters = 'N/A',
                         RMSE     = ols.rmse,
                         Rsquared = ols.summary$adj.r.squared)
```

MARS:

```
# Model tuning controls

ctrl <- trainControl(method  = "repeatedcv",
                     number  = 5, # 5 fold cross validation
                     repeats = 1  # 1 repeats
                     )


# Fit the model
```

```
fit.mars <- train(data = df.clean,
                  Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
                   Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
                   Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
                   Festival + City + Time_Order_Picked,
                  method     = "earth",           # Earth is for MARS models
                  tuneLength = 9,                 # 9 values of the cost function
                  preProc    = c("center","scale"), # Center and scale data
                  trControl  = ctrl
                 )
# Get the RMSE and R Squared of the model
hyperparameters.mars = list('degree' = fit.mars[["bestTune"]][["degree"]],
                            'nprune' = fit.mars[["bestTune"]][["nprune"]])


keyDiagnostics.mars <- data.frame(Model   = 'MARS',
                                  Notes    = 'caret and earth',
                                  Hyperparameters = paste('Degree =', hyperparameters.mars$degree, ',',
                                                          'nprune =', hyperparameters.mars$nprune)
                                  )


keyDiagnostics.mars <- cbind(keyDiagnostics.mars,
                             fit.mars$results %>%
                              filter(degree == hyperparameters.mars$degree,
                                     nprune == hyperparameters.mars$nprune) %>%
                              dplyr::select(RMSE, Rsquared)
                            )
```

Elastic Net:

```
# Fit the model
fit.elasticnet <- train(data = df.clean,
                  Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
                   Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
                   Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
                   Festival + City + Time_Order_Picked,
```

```
            method     = "glmnet",          # Elastic net
            tuneLength = 10,                # 9 values of the cost function
            preProc    = c("center","scale"), # Center and scale data
            trControl  = ctrl
           )
# Function to get the best hypertuned parameters
get_best_result = function(caret_fit) {
 best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
 best_result = caret_fit$results[best, ]
 rownames(best_result) = NULL
 best_result
}
result.elasticnet <- get_best_result(fit.elasticnet)

# Gather key diagnostics for summary table
# Get the RMSE and R Squared of the model
hyperparameters.elasticnet = list('Alpha'  = result.elasticnet$alpha,
                      'Lambda' = result.elasticnet$lambda)


keyDiagnostics.elasticnet <- data.frame(Model    = 'Elastic Net',
                           Notes    = 'caret and elasticnet',
                           Hyperparameters = paste('Alpha =',
                                   hyperparameters.elasticnet$Alpha, ',',
                                   'Lambda =',
                                   hyperparameters.elasticnet$Lambda),
                           RMSE     = result.elasticnet$RMSE,
                           Rsquared = result.elasticnet$Rsquared
                           )
```

PCR:

```
# Fit the model
fit.pcr <- mvr(data = df.clean,
            Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
```

```
    Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
    Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
    Festival + City + Time_Order_Picked,
  center = TRUE,
  scale  = TRUE,
  validation = "CV"
  )
# See the summary output
summary(fit.pcr)
validationplot(fit.pcr)
validationplot(fit.pcr, val.type = 'R2')
# Gather key diagnostics for summary table
# Get the RMSE and R Squared of the model
# Key diagnostics for PCR final summary table
RMSE.pcr <- RMSEP(fit.pcr, ncomp=15)
R2.pcr <- R2(fit.pcr, ncomp = 1:15)

# Get the RMSE and R Squared of the model
keyDiagnostics.pcr <- data.frame(Model    = 'PCR',
                  Notes    = 'pcr',
                  Hyperparameters = paste('ncomp = ', 15),
                  RMSE     = min(RMSE.pcr$val),
                  Rsquared = max(R2.pcr$val) )
```

Decision Tree:

```
#using data without any pre-processing
fit.cart <- train(data = df.clean,
    Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
    Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
    Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
    Festival + City + Time_Order_Picked,
  method="rpart",
  tuneLength = 10,          # 10 values of the cost function
  #preProc    = c("center","scale"),
```

```
            trControl=ctrl)
# Get the RMSE and R Squared of the model
hyperparameters.cart = list('cp' = fit.cart[["bestTune"]][["cp"]])
keyDiagnostics.cart <- data.frame(Model    = 'CART',
                                  Notes    = 'rpart',
                                  Hyperparameters = paste('cp =',
                                                          hyperparameters.cart$cp),
                                  RMSE     = fit.cart$results[1,'RMSE'],
                                  Rsquared = fit.cart$results[1, 'Rsquared']
                                  )
```

Random Forest:

```
fit.rf <- train(data = df.clean,
                Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
                  Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
                  Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
                  Festival + City + Time_Order_Picked,
                method="rf",
                tuneLength = 10,                # 9 values of the cost function
                #preProc    = c("center","scale"),
                trControl=ctrl,
                allowParallel = TRUE)
# Get the RMSE and R Squared of the model
hyperparameters.rf = list('mtry' = fit.rf[["bestTune"]][["mtry"]])
keyDiagnostics.rf <- data.frame(Model    = 'Random Forest',
                                  Notes    = 'rf',
                                  Hyperparameters = paste('mtry =',
                                                          hyperparameters.rf$mtry),
                                  RMSE     = fit.rf$results[3,'RMSE'],
                                  Rsquared = fit.cart$results[3, 'Rsquared'])
```

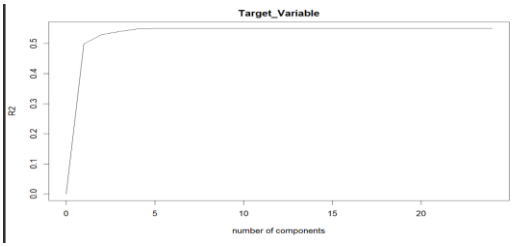
Gradient Boost Tree:

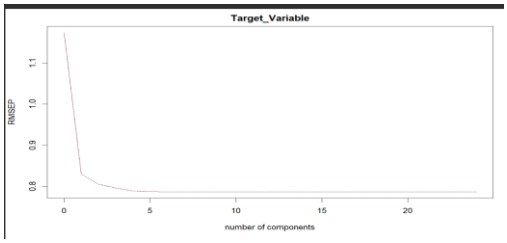
```
## gradient boosting
fit.grboost <- train(data = df.clean,
                     Target_Variable ~ Delivery_Person_Age + Delivery_Person_Ratings +
```

```
                Restaurant_Latitude + Weather_Conditions + Road_Traffic_Density +
                Vehicle_Condition + Type_Of_Vehicle + Multiple_Deliveries +
                Festival + City + Time_Order_Picked,
              method = "xgbTree",
              trControl = ctrl
 )
result.grboost = fit.grboost$results %>% arrange(fit.grboost$results['RMSE'])
hyperparameters.grboost = list('max_depth' = fit.grboost[["bestTune"]][["max_depth"]],
                    'eta' = fit.grboost[["bestTune"]][["eta"]],
                    'nrounds' = fit.grboost[["bestTune"]][["nrounds"]])
# Key diagnostics
keyDiagnostics.grboost <- data.frame(Model    = 'Gradient boost',
                       Notes    = 'xgbTree',
                       Hyperparameters = paste('max_depth =', hyperparameters.grboost$max_depth, ',',
                                'eta =', hyperparameters.grboost$eta, ',',
                              'nrounds=', hyperparameters.grboost$nrounds),
                       RMSE = result.grboost[1,'RMSE'],
                       Rsquared = result.grboost[1,'Rsquared'])
```

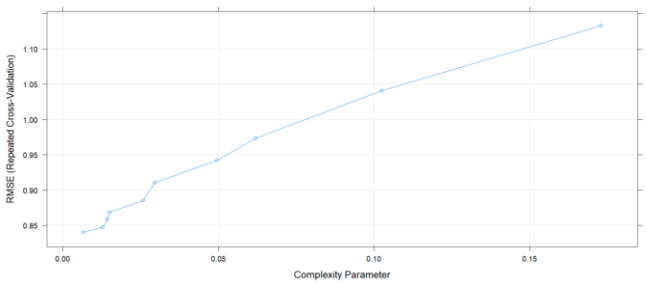
Plots

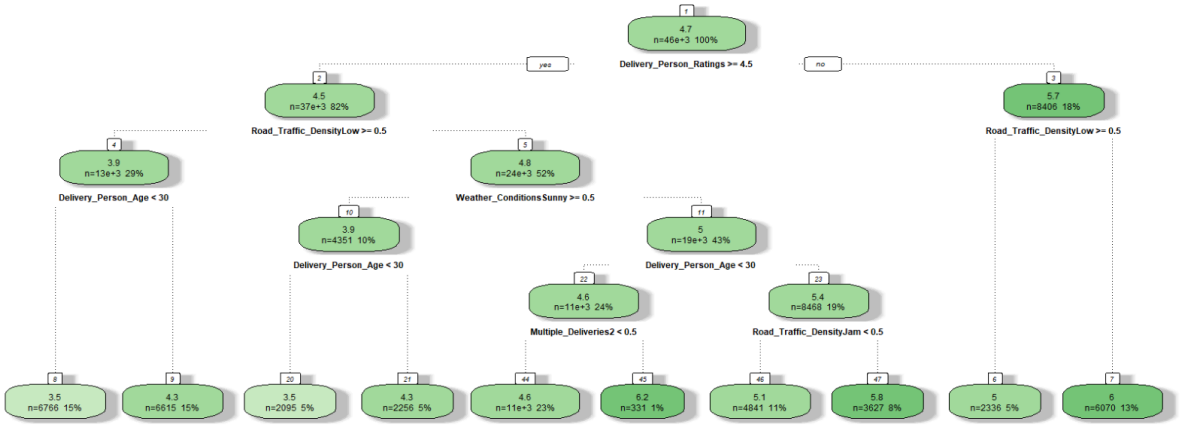
PCR:



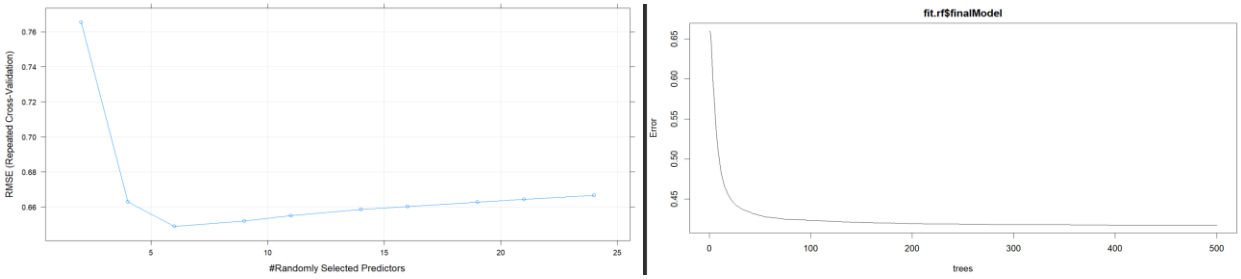


Decision Tree:





Random Forest:



Gradient Boost: