SurvMEth 640 Final Project Report

Cheng, Chia Wen

2023-04-21

Setup

```r
library(caret)
library(randomForest)
library(partykit)
library(pdp)
library(iml)
library(fastAdaboost)
library(dplyr)
library(MLmetrics)
library(ranger)
library(ada)
library(mlforsocialscience)
library(DataExplorer)
library(MASS)
```

Question of Interest

Predicted probabilities of days when the charging stations are utilized more frequently.

Electric Vehicle (EV) is creating a larger marketshare annually in many countries. As electric vehicle companies plan for more supplies with more EV products designed, manufactured, and publicized, they should also consider the demand from both existed and potential customers, and try to be as align with the needs as possible, so that the demand and the supply remain balanced, and the economic activity cycle continues.

One of the consumer demands centers around the use of the charging stations. I would like to use machine learning methodology to predict days when customers charge their cars at the scoped stations the most. This analysis will benefit both the suppliers–knowing which days would be the best for maintenance work, and the customers–being least interrupted of their charging needs due to maintenance.

The variable of interest, 'weekday,' is categorical, representing the day the recorded drivers charge their cars. Input variables contian both categorical and continuous variables, such as dollars they spent for charging, the total energy use of a given EV charging session, the type of facility a station is installed at, etc. I would like to build models, applying ordinal logit regression, several decision trees, and K-fold Cross-Validation Random Forest to predict the probabilities of stations being used for charging each day in a week. I will then compare the prediction performances of each model, and make recommendations based on the results.

The prediction results will focus on answering which days do users charge their EVs at stations the most on average. This analysis can make meaningful contribution for EV suppliers to decide days in a week to conduct regular maintenance–which are preferably to be on the least demanded days. Prediction is important to this research question of interest because it helps analyze the frequencies of usage along with the features of charging stations. Prediction also allows the suppliers to select areas to add more stations if a pattern of specifically high demand is observed in an area. In addition, the stations recorded in this data is only a portion of the charging sites. The models and the prediction process can be applied to the expanded data of collection. It is also more flexible for suppliers if they are to publish EVs with new features. Simply looking at historic data does not help to efficiently make decisions because it is not flexible enough for new features of EVs.

Data being analyzed is collected from November 2014 to October 2015, which may raise the concern of being out of the date. However, similar analytic methods can be applied to the latest data, and recommendations made from this dataset should remain valid in predicting and planning a long-term business plan.

Descriptive Statistics and Visualizations

First, I load in data, drop uninterested variables, and look at the classes of the remaining variables. I drop 'dollars' since there are many zeros in it, I assume that this variable will not help distinct users or stations. I also drop 'distance' because 31.37% of the values are missing in this variable. 'UserID,' 'StationID,' and 'LocationID' are just identifiers that do not help learning features and making predictions, and thus they are dropped. Variables regarding charging start time and terminate time are dropped because I do not wish to make prediction analysis on small-scale time within a day.

```
setwd("G:/My Drive/0. study abroad/academic/9. 2023 Winter/2. SurvMeth 640 Machine Learning in Social S
data <- read.csv("station_data_dataverse.csv")
data1 <- data[, c(1, 2, 8, 9, 10, 15:24)]
names(data1)
```

```
##  [1] "sessionId"      "kwhTotal"       "chargeTimeHrs"  "weekday"
##  [5] "platform"       "managerVehicle" "facilityType"   "Mon"
##  [9] "Tues"           "Wed"            "Thurs"          "Fri"
## [13] "Sat"            "Sun"            "reportedZip"
```

```
str(data1)
```

```
## 'data.frame':    3395 obs. of  15 variables:
##  $ sessionId     : int  1366563 3075723 4228788 3173284 3266500 4099366 5084244 2948436 3515913 84900
##  $ kwhTotal      : num  7.78 9.74 6.76 6.17 0.93 2.14 0.3 1.82 0.81 1.98 ...
##  $ chargeTimeHrs : num  1.511 2.177 4.672 1.768 0.299 ...
##  $ weekday       : chr  "Tue" "Wed" "Fri" "Wed" ...
##  $ platform      : chr  "android" "android" "android" "android" ...
##  $ managerVehicle: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ facilityType  : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Mon           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Tues          : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ Wed           : int  0 1 0 1 0 0 0 1 0 0 ...
##  $ Thurs         : int  0 0 0 0 1 0 0 0 1 1 ...
##  $ Fri           : int  0 0 1 0 0 1 1 0 0 0 ...
##  $ Sat           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Sun           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ reportedZip   : int  0 0 0 0 0 0 0 0 0 0 ...
```

There are 15 variables left in the data. All but 'weekday' and 'platform' are numeric. However, since 'managerVehicle,' 'facilityType,' 'reportedZip,' and 'Mon' through 'Sun' are binary variables, where 1 indicates effective and 0 ineffective, I make sure the programming algorithm understands them as factors.

I split the data, setting the seed at 98, with 20% of the data become test dataset, while the rest being training dataset.

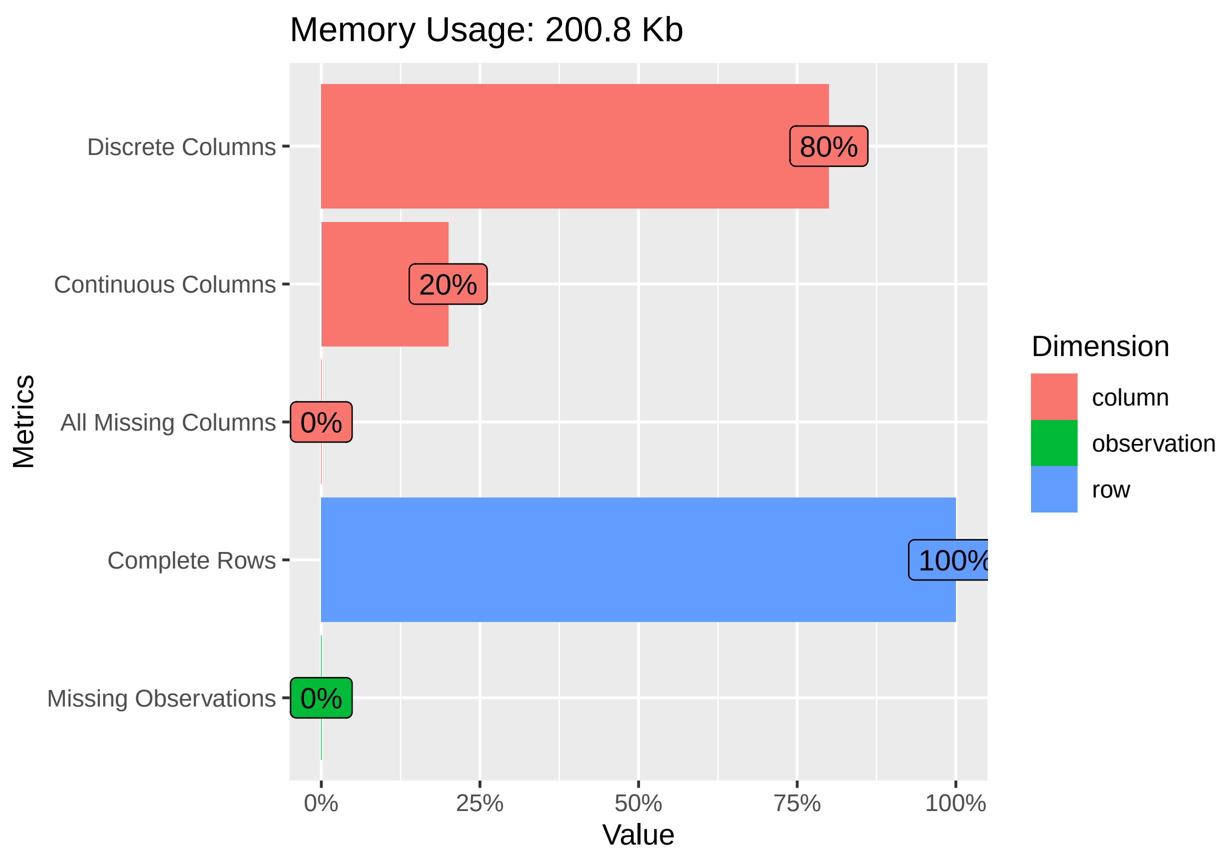
I encode 'platform' with levels from 1 to 3, representing 'ios,' 'android,' and 'web' respectively, and make sure they are factors. I also encode each day in a week to numbers, from 1 to 7.

```
data1$Mon <- as.factor(data1$Mon)
data1$Tues <- as.factor(data1$Tues)
data1$Wed <- as.factor(data1$Wed)
data1$Thurs <- as.factor(data1$Thurs)
data1$Fri <- as.factor(data1$Fri)
data1$Sat <- as.factor(data1$Sat)
data1$Sun <- as.factor(data1$Sun)
data1$managerVehicle <- as.factor(data1$managerVehicle)
data1$facilityType <- as.factor(data1$facilityType)
data1$reportedZip <- as.factor(data1$reportedZip)
data1$platform[data1$platform == "ios"] <- 1
data1$platform[data1$platform == "android"] <- 2
data1$platform[data1$platform == "web"] <- 3
data1$platform <- as.factor(data1$platform)

# Split data into training and test datasets
set.seed(98)
train <- sample(1:nrow(data1), 0.8*nrow(data1))
data_train <- data1[train,]
data_train1 <- data_train[, -c(1, 4)]
data_test <- data1[-train,]
data_test1 <- data_test[, -c(1, 4)]

data_train_glm <- data_train[, -c(1, 8:14)]
data_test_glm <- data_test[, -c(1, 8:14)]

data_train_glm$weekday[data_train_glm$weekday == "Mon"] <- 1
data_train_glm$weekday[data_train_glm$weekday == "Tue"] <- 2
data_train_glm$weekday[data_train_glm$weekday == "Wed"] <- 3
data_train_glm$weekday[data_train_glm$weekday == "Thu"] <- 4
data_train_glm$weekday[data_train_glm$weekday == "Fri"] <- 5
data_train_glm$weekday[data_train_glm$weekday == "Sat"] <- 6
data_train_glm$weekday[data_train_glm$weekday == "Sun"] <- 7
data_test_glm$weekday[data_test_glm$weekday == "Mon"] <- 1
data_test_glm$weekday[data_test_glm$weekday == "Tue"] <- 2
data_test_glm$weekday[data_test_glm$weekday == "Wed"] <- 3
```

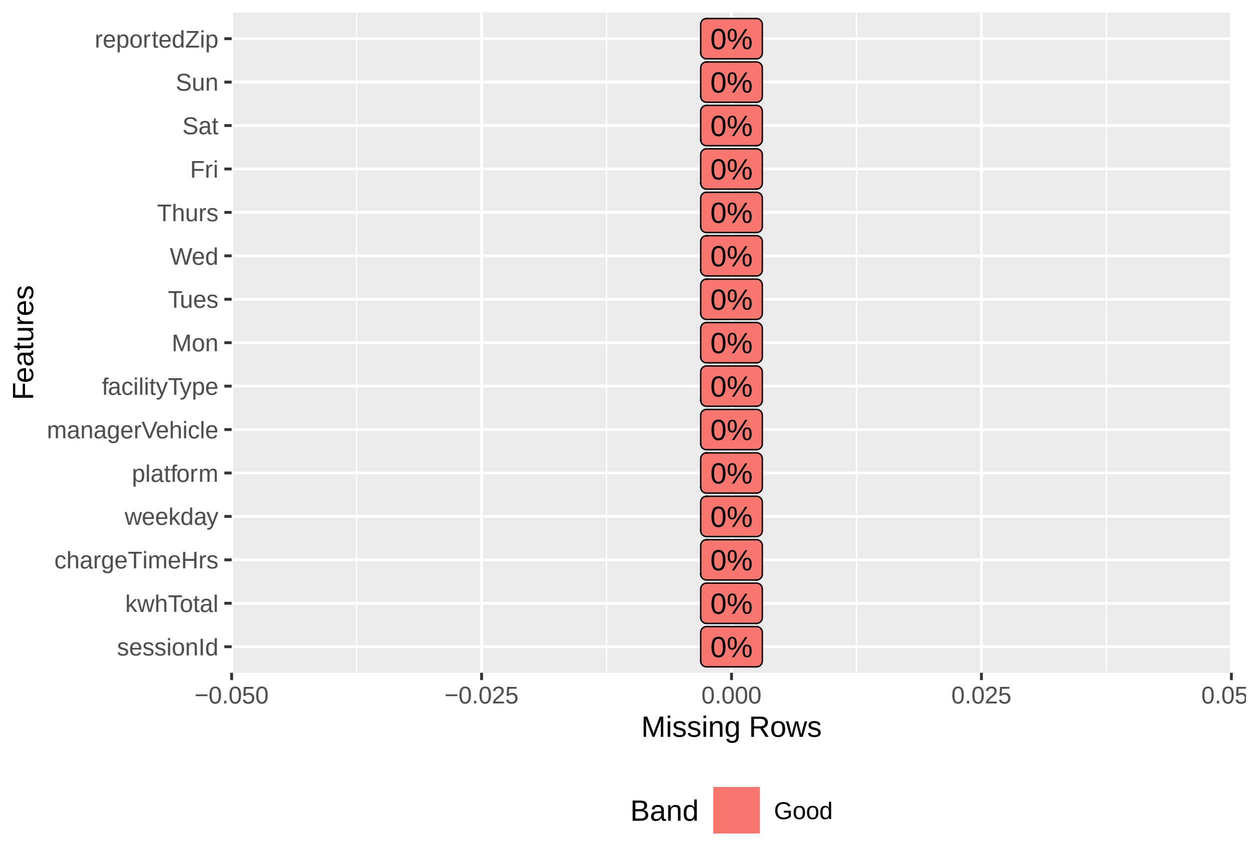
```
data_test_glm$weekday[data_test_glm$weekday == "Thu"] <- 4
data_test_glm$weekday[data_test_glm$weekday == "Fri"] <- 5
data_test_glm$weekday[data_test_glm$weekday == "Sat"] <- 6
data_test_glm$weekday[data_test_glm$weekday == "Sun"] <- 7
data_train_glm$weekday <- as.factor(data_train_glm$weekday)
data_test_glm$weekday <- as.factor(data_test_glm$weekday)

# Exploratory Analysis of Data
introduce(data_train) # no missing values
```

```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1 2716      15               12                  3                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0          2716              40740       205640
```

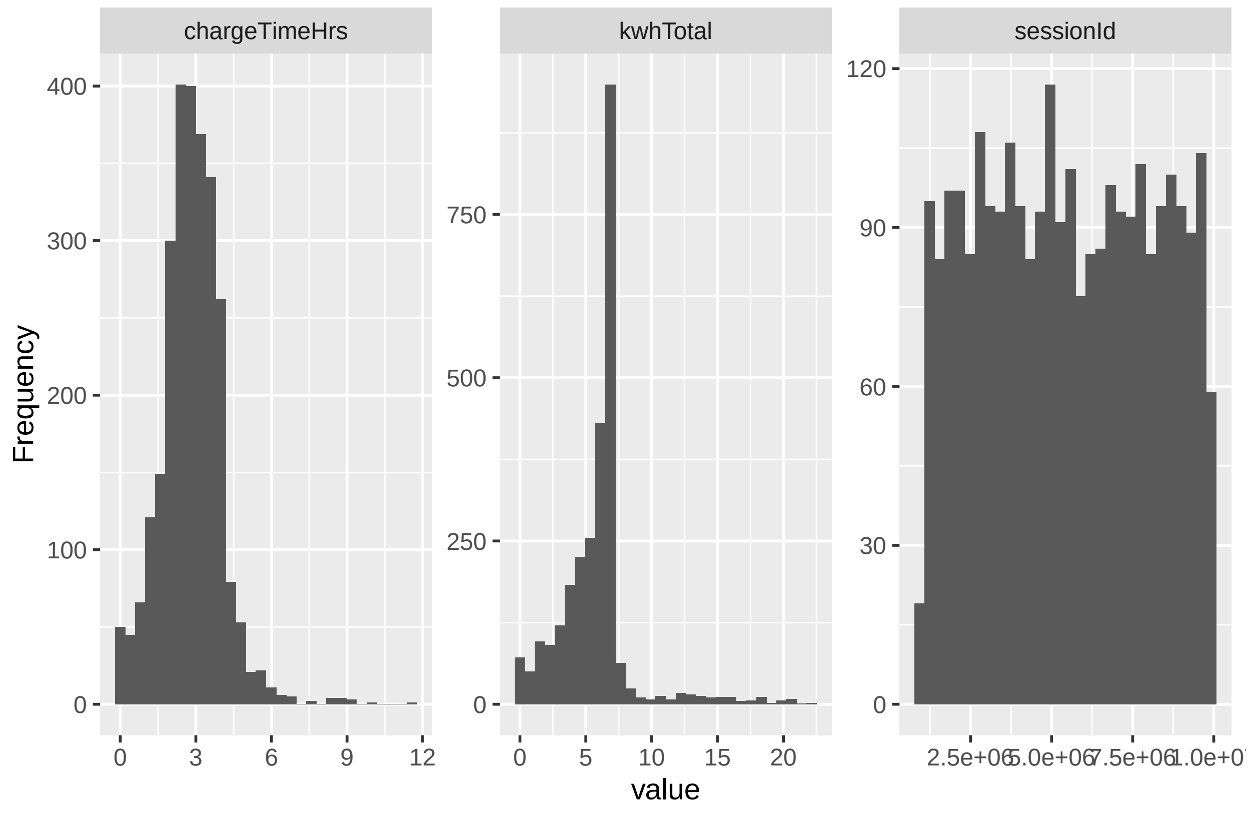
```
plot_intro(data_train)
```



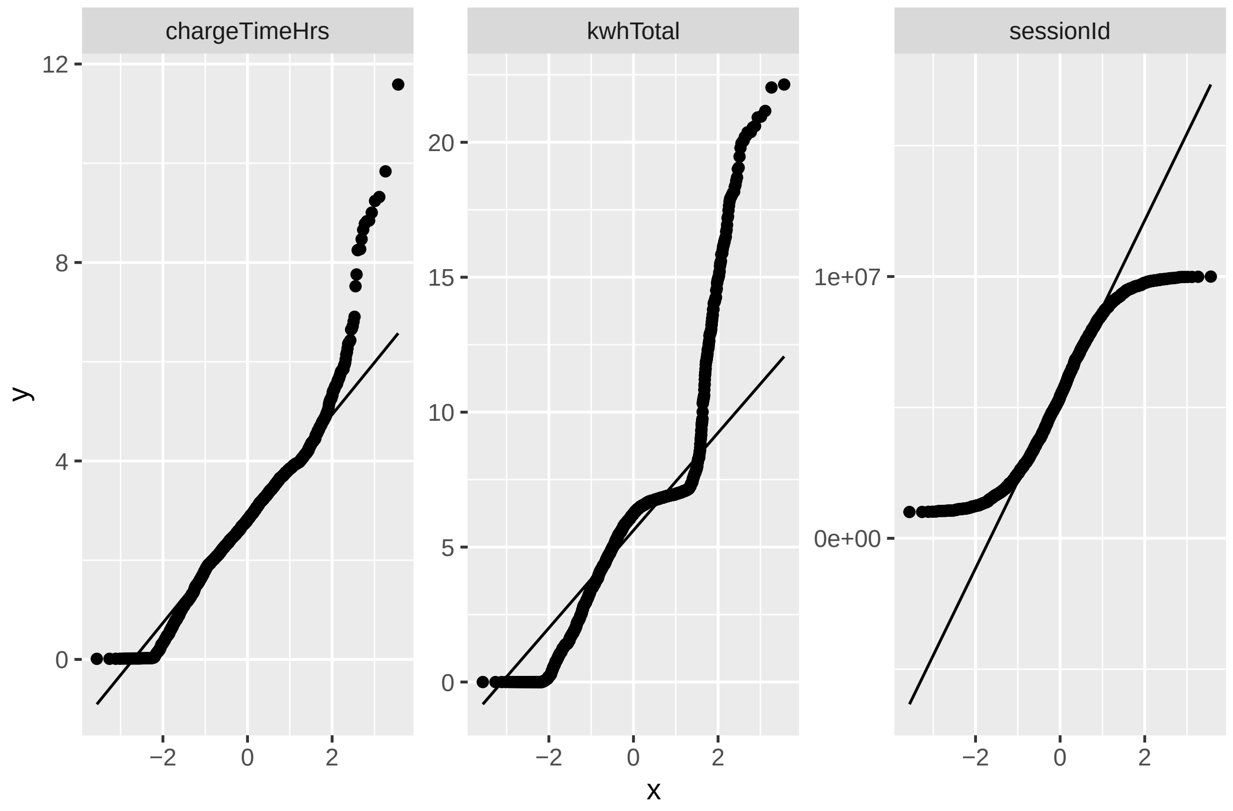
```
plot_missing(data_train)
```



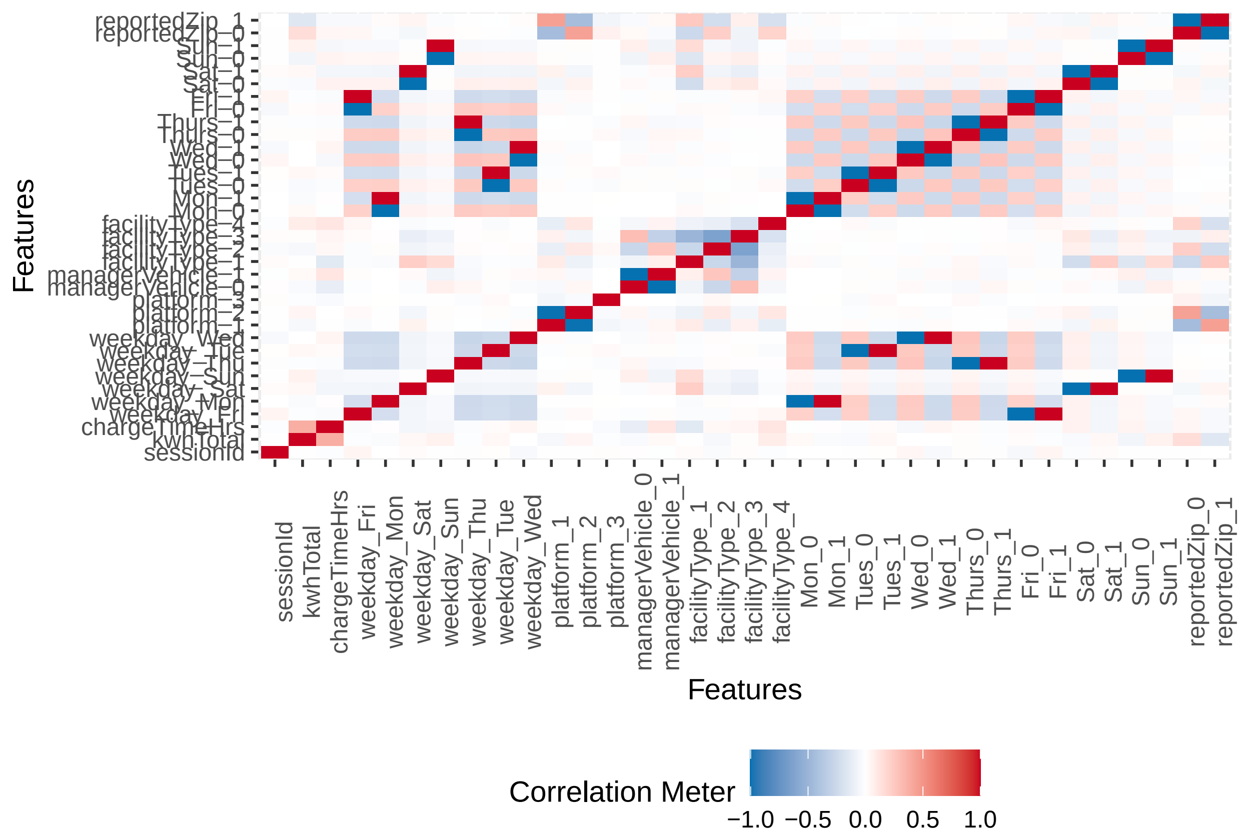
```
plot_histogram(data_train)
```



```
plot_qq(data_train)
```



```
plot_correlation(na.omit(data_train))
```



From the exploratory analysis, there are no missing values in the training and test datasets. 80% of the variables are categorical, and 20% are continuous. 'ChargeTimeHrs' is right-skewed, indicating that it has a mean greater than its median, and still greater than its mode. Its relationship between residuals and fitted values are more of an exponential distribution. 'KwhTotal' is also right-skewed, with a extreme peak at kwhTotal approximately of 7. It has a cubic-function-like relationship between its residuals and fitted values. There are several strong correlations between the features.

Prediction Models

```r
# Ordinal Logit Regression Model
o.logit <- polr(weekday ~ ., data = data_train_glm)
summary(o.logit)
```

```
## Call:
## polr(formula = weekday ~ ., data = data_train_glm)
##
## Coefficients:
##                      Value Std. Error t value
## kwhTotal           0.01554    0.01306  1.1898
## chargeTimeHrs     -0.07847    0.03173 -2.4729
## platform2         -0.05829    0.08027 -0.7262
## platform3         -0.70323    0.78043 -0.9011
## managerVehicle1 -0.04457    0.07456 -0.5978
## facilityType2   -0.47297    0.11605 -4.0754
## facilityType3   -0.39904    0.10227 -3.9017
## facilityType4   -0.16479    0.22606 -0.7290
## reportedZip1    -0.19530    0.08931 -2.1867
##
## Intercepts:
##     Value    Std. Error t value
## 1|2  -2.1813   0.1685   -12.9478
## 2|3  -1.1971   0.1651    -7.2521
## 3|4  -0.3095   0.1638    -1.8894
## 4|5   0.7277   0.1650     4.4105
## 5|6   3.0310   0.1990    15.2291
## 6|7   4.2648   0.2727    15.6374
##
## Residual Deviance: 9184.826
## AIC: 9214.826
```

```
data_train_glm %>% group_by(weekday) %>% count(weekday)
```

```
## # A tibble: 7 x 2
## # Groups:   weekday [7]
##   weekday     n
##   <fct>   <int>
## 1 1         492
## 2 2         516
## 3 3         589
## 4 4         576
## 5 5         476
## 6 6          47
## 7 7          20
```

I first apply the ordinal logit regression model to compare its performance with other models. Since the outcome variable of interest is with 7 levels, ordinary logit regression model helps predict the probabilities of each level.

Most of the variables have positive associations with 'weekday,' after transforming the coefficients to non-exponentiated ones. The intercepts indicate the differences between the two listed levels, when holding all other regressors constant.

```
# Comparing Models with caretList
library(caretEnsemble)
ctrl_ensemble <- suppressWarnings(trainControl(method = "cv",
                              number = 5,
                              index = createFolds(data_train_glm$weekday, 5),
                              savePredictions = "final",
                              returnData = FALSE,
                              verboseIter = FALSE))

mods <- c('treebag', 'ranger', 'xgbTree')

model_list <- caretList(weekday ~ .,
                        data = data_train_glm,
                        trControl = ctrl_ensemble,
                        metric = "Accuracy",
                        methodList = mods)
```

I also run a Bagging model, a Random Forest model, and an XGBoost (Extreme Gardient Boosting) model, using the same CV splits with 5-fold CV. This step allows me to compare their prediction performances in the next section.

```
# K-fold Cross-Validation Random Forest
ctrl  <- trainControl(method = "cv",
                      number = 10,
                      summaryFunction = multiClassSummary,
                      classProbs = TRUE)
grid <- expand.grid(mtry = c(sqrt(ncol(data_train_glm)), # number randomly variable selected is mtry
                             log(ncol(data_train_glm))),
                    splitrule = c("gini"),
                    min.node.size = 10)
levels(data_train_glm$weekday) <- c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
set.seed(98)
rf <- train(weekday ~ .,
            data = data_train_glm,
            method = "ranger",
            trControl = ctrl,
            tuneGrid = grid,
            metric = "ROC",
            importance = "impurity")
rf
```

```
## Random Forest
##
## 2716 samples
##    6 predictor
##    7 classes: 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2444, 2444, 2444, 2445, 2444, 2445, ...
## Resampling results across tuning parameters:
##
##   mtry      logLoss   AUC        prAUC      Accuracy   Kappa        Mean_F1
##   1.945910  1.673905  0.6234665  0.1852967  0.2176172  0.005238325        NaN
##   2.645751  1.666047  0.6276946  0.1843252  0.2194527  0.017748330  0.2530543
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
##   0.1460240         0.8578683               NaN            0.8579917
##   0.2055769         0.8594766         0.2869873            0.8591333
##   Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##         NaN       0.1460240    0.03108817           0.5019462
##   0.2869873       0.2055769    0.03135039           0.5325268
##
## Tuning parameter 'splitrule' was held constant at a value of gini
##
## Tuning parameter 'min.node.size' was held constant at a value of 10
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 2.645751, splitrule = gini
##  and min.node.size = 10.
```

Last, I build a K-fold Cross-Validation Random Forest Model with 10 folders. The model with the highest accuracy is the one using 'mtry = 2.645751.'

Conclusion

```
# Ordinal Logit Regression Model
data_test_glm$olog_pred <- predict(o.logit, newdata = data_test_glm)
xtab2 <- table(data_test_glm$weekday, data_test_glm$olog_pred)
xtab2
```

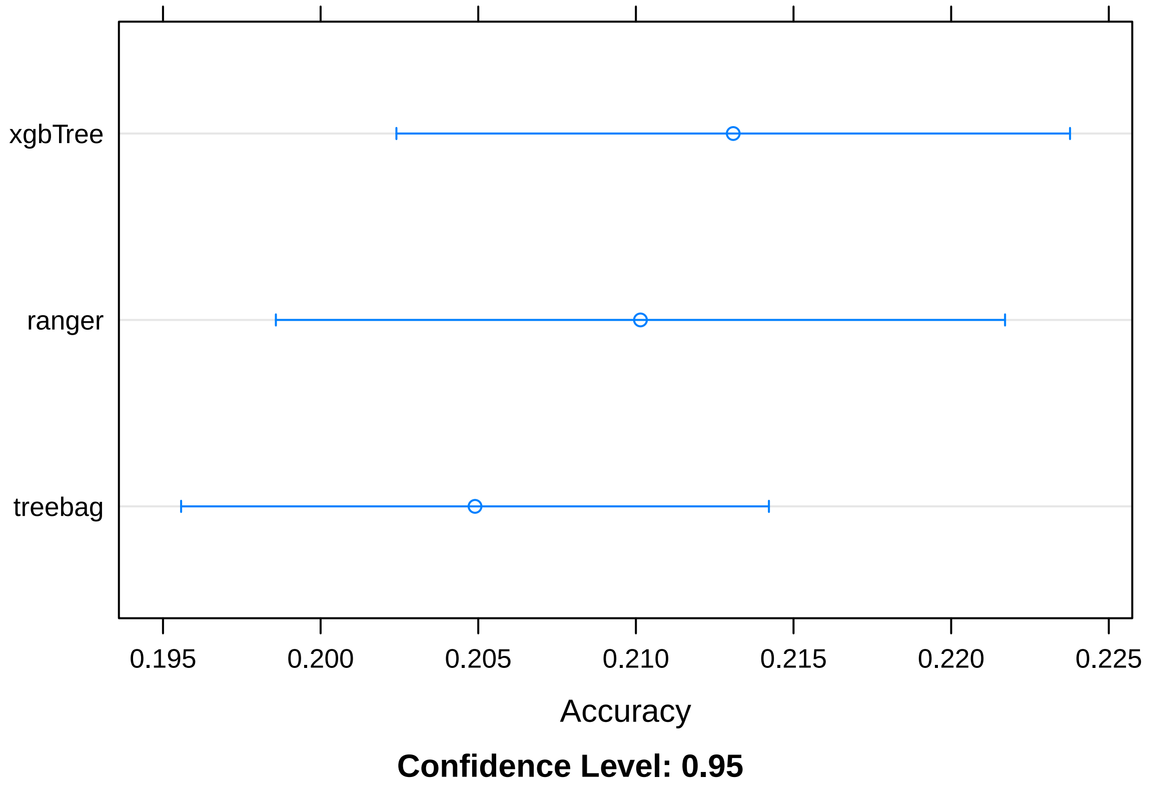
```
##
##       1   2   3   4   5   6   7
##   1   6   0  74  43   1   0   0
##   2   6   0  66  47   0   0   0
##   3  12   0  79  33   0   0   0
##   4   9   0 107  43   0   0   0
##   5  10   0  83  39   2   0   0
##   6   0   0   1  13   1   0   0
##   7   0   0   0   3   1   0   0
```

```
confusionMatrix(xtab2)
```

```
## Confusion Matrix and Statistics
##
##
##       1   2   3   4   5   6   7
##   1   6   0  74  43   1   0   0
##   2   6   0  66  47   0   0   0
##   3  12   0  79  33   0   0   0
##   4   9   0 107  43   0   0   0
##   5  10   0  83  39   2   0   0
##   6   0   0   1  13   1   0   0
##   7   0   0   0   3   1   0   0
##
## Overall Statistics
##
##                Accuracy : 0.1915
##                  95% CI : (0.1625, 0.2231)
##     No Information Rate : 0.6038
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0101
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity          0.139535       NA   0.1927  0.19457 0.400000       NA
## Specificity          0.814465   0.8247   0.8327  0.74672 0.804154  0.97791
## Pos Pred Value       0.048387       NA   0.6371  0.27044 0.014925       NA
## Neg Pred Value       0.933333       NA   0.4036  0.65769 0.994495       NA
## Prevalence           0.063328   0.0000   0.6038  0.32548 0.007364  0.00000
## Detection Rate       0.008837   0.0000   0.1163  0.06333 0.002946  0.00000
## Detection Prevalence 0.182622   0.1753   0.1826  0.23417 0.197349  0.02209
## Balanced Accuracy    0.477000       NA   0.5127  0.47065 0.602077       NA
##                      Class: 7
## Sensitivity                NA
## Specificity          0.994109
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence           0.000000
## Detection Rate       0.000000
## Detection Prevalence 0.005891
## Balanced Accuracy          NA
```

```
# Comparing Models with caretList
dotplot(resamples(model_list), metric = 'Accuracy')
```



```r
# K-fold Cross-Validation Random Forest
levels(data_test_glm$weekday) <- c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
y <- predict(rf, newdata = data_test_glm)
y_prob <- predict(rf, newdata = data_test_glm, type = "prob")
confusionMatrix(y, data_test_glm$weekday, mode = "everything", positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Mon Tue Wed Thu Fri Sat Sun
##        Mon  13  11  11   8  10   0   0
##        Tue  13  11  17  20  11   1   0
##        Wed  49  50  44  63  55   5   1
##        Thu  38  40  43  54  39   5   2
##        Fri  10   7   9  14  19   3   0
##        Sat   1   0   0   0   0   1   0
##        Sun   0   0   0   0   0   0   1
##
## Overall Statistics
##
##                Accuracy : 0.2106
##                  95% CI : (0.1805, 0.2432)
##     No Information Rate : 0.2342
##     P-Value [Acc > NIR] : 0.934
##
##                   Kappa : 0.0142
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Mon Class: Tue Class: Wed Class: Thu Class: Fri
## Sensitivity             0.10484    0.09244     0.3548    0.33962    0.14179
## Specificity             0.92793    0.88929     0.5982    0.67885    0.92110
## Pos Pred Value          0.24528    0.15068     0.1648    0.24434    0.30645
## Neg Pred Value          0.82268    0.82178     0.8058    0.77074    0.81361
## Precision               0.24528    0.15068     0.1648    0.24434    0.30645
## Recall                  0.10484    0.09244     0.3548    0.33962    0.14179
## F1                      0.14689    0.11458     0.2251    0.28421    0.19388
## Prevalence              0.18262    0.17526     0.1826    0.23417    0.19735
## Detection Rate          0.01915    0.01620     0.0648    0.07953    0.02798
## Detection Prevalence    0.07806    0.10751     0.3932    0.32548    0.09131
## Balanced Accuracy       0.51638    0.49086     0.4765    0.50923    0.53145
##                      Class: Sat Class: Sun
## Sensitivity            0.066667   0.250000
## Specificity            0.998494   1.000000
## Pos Pred Value         0.500000   1.000000
## Neg Pred Value         0.979321   0.995575
## Precision              0.500000   1.000000
## Recall                 0.066667   0.250000
## F1                     0.117647   0.400000
## Prevalence             0.022091   0.005891
## Detection Rate         0.001473   0.001473
## Detection Prevalence   0.002946   0.001473
## Balanced Accuracy      0.532580   0.625000
```

For the ordinal logit regression and the K-fold Cross-Validation Random Forest Model, the latter one has a higher accuracy of 0.1988, which is more preferable. 'Sat' and 'Sun' are rarely predicted from the ordinal logit regression. But since they are also least used days of charging stations, they have the highest specificity scores among the weekdays, which means the number of the true negative counts is high.

The set of decision tree models has a result that the xgbTree model has the highest accuracy of above 0.22 among the three models, while the random forest model has a still higher accuracy comparing to the bagging tree model. Thus, I conclude that the xgbTree model performs the best in prediction performance.

All models have poor prediction performances. While independent variables show a dgree of correlations, it seems like that the distinction between this sample is not sufficient enough for the machine to learn to predict well, instead of the possibility that variances across this sample is too large to come to similar conclusions of predictions.

Despite the poor accuracy of performance, this prediction can be applied to determine on which days should the maintenance take place. It would be better to collect more data to make samples distinguishable before coming up with a business plan.

References

1. Electric Vehicle Charging Dataset: https://www.kaggle.com/datasets/michaelbryantds/electric-vehicle-charging-dataset?select=station_data_dataverse.csv
2. Data Dictionary: https://github.com/asensio-lab/workplace-charging-experiment (the unbolded ones will be dropped before splitting the training and testing datasets)