IDV Learner Capstone 2021

Michael Jaeger

7/27/2021

Contents

1 Introduction

This R markdown is the product of the Harvard edX Data Science Professional Certificate program Capstone. The Capstone course was designed from the course lectures on Machine Learning. The goal of the project is to build a machine learning algorithm that predicts a certain hours' load level classification based only on observations of price, congestion, dewpoint and air temperature.

2 Setup, Methods and Analysis

There are minimal packages required to load into the R environment as noted in the p_load function.

I created the dataset used for the analysis. It consist of hourly weather and integrated load values for hour ending 16 (3:00 - 3:59pm EST) for the New York City load zone in the New York Independent System Operator footprint.

The weather was scraped from NOAA using another script I have work purposes. I combined the weather data and load data in another script and created a csv file called "finaldata.csv".

Initially, I wanted to run a time series forecast, but quickly realized that I was in over my head in that I hadn't studied this realm of machine learning yet. The data was not standardized. I then turned to classification, where I classified the load values into 5 bins based on the values: "High", "Med-high", "Med", "Med-low" and "Low". I trained on this classification.

```
# Note: this process could take a couple of minutes
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(caret,data.table,recommenderlab,heatmaply,
               tidyverse, tinytex,readr, tidyquant)


rm(list=ls())
# Load data
data <- read_csv("finaldata.csv", col_types = cols(date = col_date(format = "%Y-%m-%d")))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

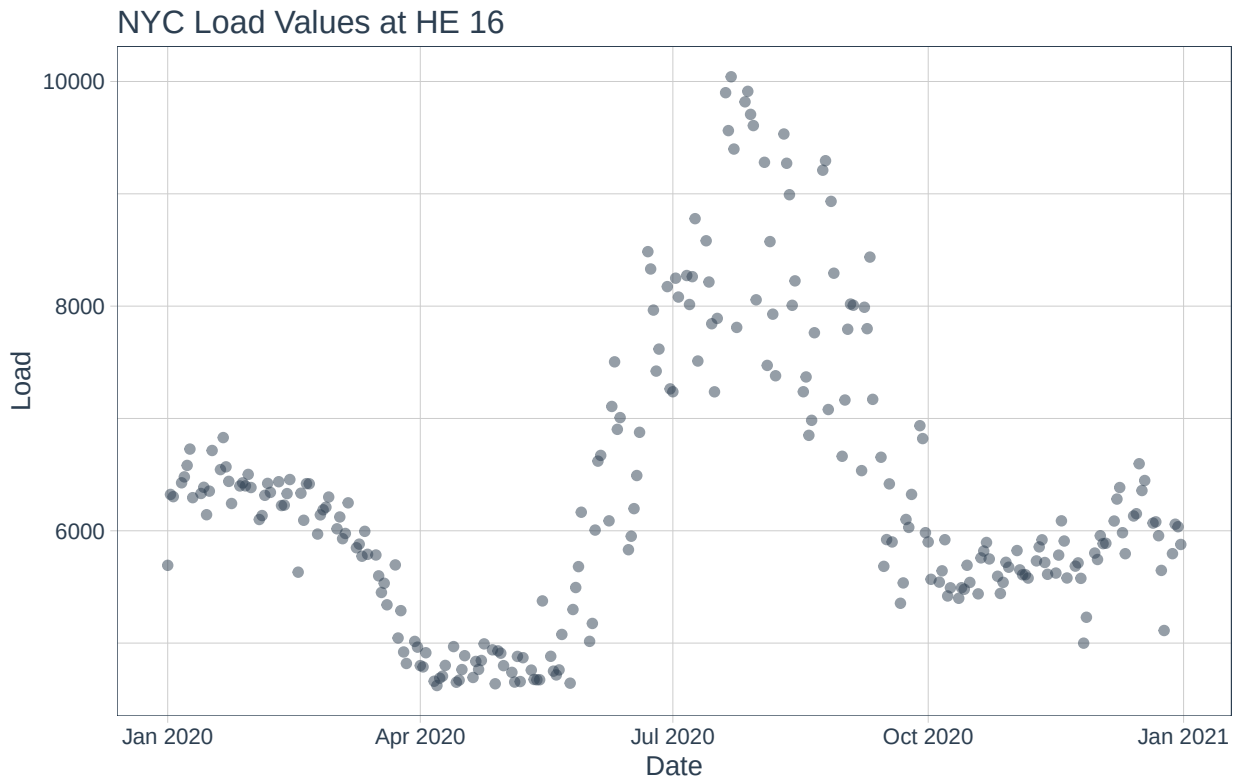
The supplied column names are not frinedly, lest change them.

```
# Rename some columns for friendliness
colnames(data)[3] <- "Price"
colnames(data)[4] <- "Load"
colnames(data)[5] <- "Congestion"
```

Lets look at the Hour 16 loads across NYC load zone.

```
# visualize the dataset to predict on
visualModelData <- data %>%
      ggplot(aes(x = date, y = Load)) +
      geom_point(alpha = 0.5, color = palette_light()[[1]]) +
      labs(title = "NYC Load Values at HE 16", x = "Date") +
      theme_tq()

visualModelData
```



We need to classify the loads into categories.

```
# Classify the load levels into 5 bins, High, med-high, med, med-low, low
data$loadClass <- ifelse(data$Load > 9500,"High",
                      ifelse(data$Load > 7500 & data$Load <= 9500,"Med-high",
                             ifelse(data$Load > 6200 & data$Load <= 7500, "Normal",
                                    ifelse(data$Load > 5000 & data$Load <= 6200, "Med-Low",
                                           ifelse(data$Load > 3800 & data$Load <= 5000, "Low",""")))))
```

Grab the columns we need:

```
# Select the final variables for the training set
data <- data %>%
      select(Price, Load, Congestion, Air_Temp, Dewpoint,loadClass)
```

2.1 Methods and Analysis

Split into test and train datasets, first set seed:

```
# Split data into a test index, then into training and test sets
set.seed(1972)
test_index <- createDataPartition(y = data$loadClass, times = 1, p = 0.1, list = FALSE)
train <- data[-test_index, ]
test <- data[test_index, ]
```

Now to set the control parameters to cross validate 10 and set metric to accuracy.

```
# Set up the models to use 10-fold cross validation and Accuracy metrics
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"
```

Run the five models chosen for classification, LDA, Cart, KNN, SVM, RF:

```
# 1 - linear algorithm
set.seed(1972)
lda <- train(loadClass~., data=train, method="lda", metric=metric, trControl=control)

# 2 - nonlinear algorithm
set.seed(1972)
cart <- train(loadClass~., data=train, method="rpart", metric=metric, trControl=control)

# 3 - kernel nearest neighbor
set.seed(1972)
knn <- train(loadClass~., data=train, method="knn", metric=metric, trControl=control)

# 4 - SVM
set.seed(1972)
svm <- train(loadClass~., data=train, method="svmRadial", metric=metric, trControl=control)

# 5 - Random Forest
set.seed(1972)
rf <- train(loadClass~., data=train, method="rf", metric=metric, trControl=control)
```

3 Results

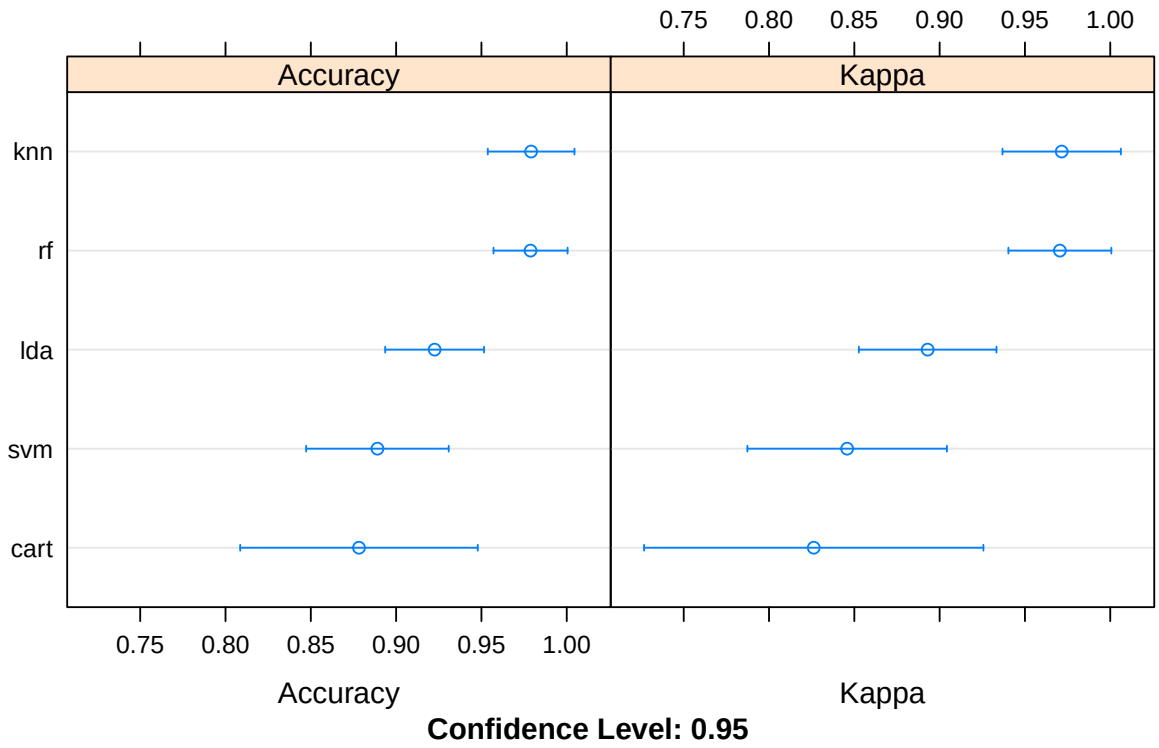
Now it's time show the results of the models.

```
# summarize accuracy of models
results <- resamples(list(lda=lda, cart=cart, knn=knn, svm=svm, rf=rf))
# Show results
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.8260870 0.9130435 0.9165217 0.9225685 0.9565217 0.9600000    0
## cart 0.7500000 0.7869565 0.9100791 0.8782332 0.9565217 1.0000000    0
## knn  0.9130435 0.9687500 1.0000000 0.9791377 1.0000000 1.0000000    0
## svm  0.7826087 0.8695652 0.8775000 0.8890415 0.9459091 0.9565217    0
## rf   0.9130435 0.9587500 1.0000000 0.9787899 1.0000000 1.0000000    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.7591623 0.8805969 0.8875631 0.8929669 0.9400842 0.9458874    0
## cart 0.6372796 0.6982250 0.8661990 0.8262009 0.9396211 1.0000000    0
## knn  0.8795812 0.9572447 1.0000000 0.9715278 1.0000000 1.0000000    0
## svm  0.6916890 0.8177692 0.8329094 0.8457827 0.9220924 0.9399478    0
## rf   0.8795812 0.9425106 1.0000000 0.9704702 1.0000000 1.0000000    0
```

KNN has the highest mean accuracy value of .9791, but it is difficult to see the best outcome in the table. Lets plot it:

```
# Hard to tell from table, lets visualize the results in a dotplot
dotplot(results)
```



Now we have our best model, lets see the summary:

```
# summarize Best Model
print(knn)
```

```
## k-Nearest Neighbors
##
## 234 samples
##   5 predictor
##   5 classes: 'High', 'Low', 'Med-high', 'Med-Low', 'Normal'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 211, 211, 212, 209, 211, 210, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9698966  0.9575028
##   7  0.9791377  0.9715278
##   9  0.9622754  0.9483541
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

Let's make our predictions against the hold-out set:

```r
# Now that we have our model, lets predict using the test set
predictions <- predict(knn, test)
```

A confustion matrix is ideal for showing the results of the predictions.

```r
#Show results of predictions against validation/hold-out set.
confusionMatrix(predictions, factor(test$loadClass))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low Med-high Med-Low Normal
##   High        1   0        0       0      0
##   Low         0   5        0       0      0
##   Med-high    0   0        4       0      0
##   Med-Low     0   0        0      11      0
##   Normal      0   0        0       0      7
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8766, 1)
##     No Information Rate : 0.3929
##     P-Value [Acc > NIR] : 4.351e-12
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Med-high Class: Med-Low
## Sensitivity              1.00000     1.0000          1.0000         1.0000
## Specificity              1.00000     1.0000          1.0000         1.0000
## Pos Pred Value           1.00000     1.0000          1.0000         1.0000
## Neg Pred Value           1.00000     1.0000          1.0000         1.0000
## Prevalence               0.03571     0.1786          0.1429         0.3929
## Detection Rate           0.03571     0.1786          0.1429         0.3929
## Detection Prevalence     0.03571     0.1786          0.1429         0.3929
## Balanced Accuracy        1.00000     1.0000          1.0000         1.0000
##                      Class: Normal
## Sensitivity                   1.00
## Specificity                   1.00
## Pos Pred Value                1.00
## Neg Pred Value                1.00
## Prevalence                    0.25
## Detection Rate                0.25
## Detection Prevalence          0.25
## Balanced Accuracy             1.00
```

4 Conclusions

The model had an easy time of predicting the classification due to the variables chosen in the data. All of these variables have a high correlation to the load value. For a more 'fair' test, a larger sample size should have been made, and had variables that had less correlation such as wind direction. Unfortunately there were personal constraints that kept me from spending more time on the dataset creation.