Classification

Justin Hardy & Benjamin Frenkel

The Data Set

Starting by reading in the data set. The data set we'll use for the assignment consists of data collected by an airline organization, over their customers' submitted satisfaction surveys, as well as relevant information about their flight and demographic.

If you want to see the data set for yourself, you access the raw data here, or the page where I collected it online.

Please note that this part of the assignment reuses the data set used in the Similarity assignment.

```r
# Read data set
CustomerData_raw <- read.csv("Invistico_Airline.csv")
```

We'll also remove 90% of the rows in the data set. We'll do this by removing 90% of the satisfied reviews, and 90% of the dissatisfied reviews, and combining what remains.

We're doing this purely to make SVM run quicker, as otherwise, it'll take way too long to run.

```r
# Cut down data set, aiming for equal divide between satisfied/dissatisfied
CustomerData_sat <- CustomerData_raw[CustomerData_raw$satisfaction == "satisfied",]
CustomerData_sat <- CustomerData_sat[sample(1:nrow(CustomerData_sat),
                                            nrow(CustomerData_sat)*0.1, replace=FALSE),]
CustomerData_dis <- CustomerData_raw[CustomerData_raw$satisfaction == "dissatisfied",]
CustomerData_dis <- CustomerData_dis[sample(1:nrow(CustomerData_dis),
                                            nrow(CustomerData_dis)*0.1, replace=FALSE),]
CustomerData <- rbind(CustomerData_sat[1:(nrow(CustomerData_sat)),],
                      CustomerData_dis[1:(nrow(CustomerData_dis)),])
```

Cleaning Up The Data Set

Cleaning up data set for logistic regression, by converting qualitative columns into factors.

```r
# Factor columns
CustomerData$satisfaction <- factor(CustomerData$satisfaction) # satisfaction
CustomerData$Gender <- factor(CustomerData$Gender) # gender
CustomerData$Customer.Type <- factor(CustomerData$Customer.Type) # customer type
CustomerData$Type.of.Travel <- factor(CustomerData$Type.of.Travel) # travel type
CustomerData$Class <- factor(CustomerData$Class) # class

# Normalize factor names
levels(CustomerData$satisfaction) <- c("Dissatisfied", "Satisfied")
levels(CustomerData$Customer.Type) <- c("Disloyal", "Loyal")
levels(CustomerData$Type.of.Travel) <- c("Business", "Personal")

# Continue factoring numeric finite columns
for(i in 8:21) {
  CustomerData[,i] <-
    factor(CustomerData[,i], levels=c(0,1,2,3,4,5)) # out-of-5 ratings
}

# Normalize column names
names(CustomerData) <- gsub("\\.", " ", names(CustomerData))
names(CustomerData) <- str_to_title(names(CustomerData))
names(CustomerData) <- gsub("\\ ", ".", names(CustomerData))

# Remove na rows
CustomerData <- CustomerData[complete.cases(CustomerData),]
```

Dividing Into Train/Test/Validate

Dividing the data set into train/test/validate…

```
#reset seed
set.seed(1234)

# train/test/validate division
groups <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(CustomerData),
                nrow(CustomerData)*cumsum(c(0,groups)), labels=names(groups)))
train <- CustomerData[i=="train",]
test <- CustomerData[i=="test",]
validate <- CustomerData[i=="validate",]
```

Data Exploration

For this entire section, I've simply reused the data exploration I did in the previous assignment, since it is of the same application.

Structure

Exploring the train data, we can see that each of our 0-5 Ratings were factored into levels of 6. The reason I opted to factor the data this way is because, although the values are numerical, they're a small finite set of integers. I also noticed higher accuracy in my results after factoring the data this way, which seems to confirm that this was a good decision.

```
##        Satisfaction     Gender      Customer.Type       Age
##  Dissatisfied:3478   Female:3894   Disloyal:1406   Min.   : 7.00
##  Satisfied   :4293   Male  :3877   Loyal   :6365   1st Qu.:27.00
##                                                    Median :39.00
##                                                    Mean   :39.14
##                                                     3rd Qu.:51.00
##                                                     Max.   :80.00
##   Type.Of.Travel      Class      Flight.Distance Seat.Comfort
##  Business:5441   Business:3784   Min.   :  52    0: 294
##  Personal:2330   Eco     :3443   1st Qu.:1392    1:1236
##                  Eco Plus: 544   Median :1943    2:1706
##                                  Mean   :2007    3:1762
##                                  3rd Qu.:2572    4:1695
##                                  Max.   :6537    5:1078
##  Departure.Arrival.Time.Convenient Food.And.Drink Gate.Location
##  0: 407                            0: 368         0:   0
##  1:1259                            1:1272         1:1364
##  2:1368                            2:1606         2:1469
##  3:1405                            3:1707         3:2010
##  4:1719                            4:1556         4:1768
##  5:1613                            5:1262         5:1160
##  Inflight.Wifi.Service Inflight.Entertainment Online.Support
##  0:   4                0: 189                 0:   0
##  1: 872                1: 697                 1: 830
##  2:1629                2:1136                 2:1023
##  3:1655                3:1466                 3:1257
##  4:1894                4:2468                 4:2509
##  5:1717                5:1815                 5:2152
##  Ease.Of.Online.Booking On.Board.Service Leg.Room.Service Baggage.Handling
##  0:   1                 0:   1           0:  25           0:   0
##  1: 785                 1: 728           1: 659           1: 465
##  2:1197                 2:1064           2:1319           2: 812
##  3:1307                 3:1567           3:1331           3:1464
##  4:2428                 4:2465           4:2322           4:2912
##  5:2053                 5:1946           5:2115           5:2118
##  Checkin.Service Cleanliness Online.Boarding Departure.Delay.In.Minutes
##  0:   0          0:   1      0:   0          Min.   :  0.00
##  1: 919          1: 438      1: 927          1st Qu.:  0.00
##  2: 959          2: 793      2:1106          Median :  0.00
##  3:2127          3:1447      3:1853          Mean   : 14.27
##  4:2183          4:2916      4:2131          3rd Qu.: 13.00
##  5:1583          5:2176      5:1754          Max.   :505.00
##  Arrival.Delay.In.Minutes
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 14.65
##  3rd Qu.: 13.00
##  Max.   :507.00
```

```
## 'data.frame':    7771 obs. of  23 variables:
##  $ Satisfaction                     : Factor w/ 2 levels "Dissatisfied",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Gender                           : Factor w/ 2 levels "Female","Male": 2 1 2 1 2 2 2 1 2 1 ...
##  $ Customer.Type                    : Factor w/ 2 levels "Disloyal","Loyal": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Age                              : int  28 55 54 25 51 39 49 32 49 15 ...
##  $ Type.Of.Travel                   : Factor w/ 2 levels "Business","Personal": 1 1 1 2 1 1 1 1 1 2
##  $ Class                            : Factor w/ 3 levels "Business","Eco",..: 1 2 1 3 1 1 1 1 1 2 ..
##  $ Flight.Distance                  : int  2483 1013 708 2521 2687 338 341 2679 1887 1061 ...
##  $ Seat.Comfort                     : Factor w/ 6 levels "0","1","2","3",..: 2 5 3 6 3 6 5 6 3 1 ...
##  $ Departure.Arrival.Time.Convenient: Factor w/ 6 levels "0","1","2","3",..: 2 4 2 3 3 6 5 6 3 5 ...
##  $ Food.And.Drink                   : Factor w/ 6 levels "0","1","2","3",..: 2 4 3 6 3 6 5 6 5 1 ...
##  $ Gate.Location                    : Factor w/ 6 levels "0","1","2","3",..: 2 4 3 5 3 6 5 6 3 4 ...
##  $ Inflight.Wifi.Service            : Factor w/ 6 levels "0","1","2","3",..: 5 4 6 6 6 4 6 5 5 2 ...
##  $ Inflight.Entertainment           : Factor w/ 6 levels "0","1","2","3",..: 5 5 5 6 5 4 5 5 5 1 ...
##  $ Online.Support                   : Factor w/ 6 levels "0","1","2","3",..: 5 4 5 6 5 2 5 5 5 2 ...
##  $ Ease.Of.Online.Booking           : Factor w/ 6 levels "0","1","2","3",..: 5 5 6 6 5 4 5 5 6 2 ...
##  $ On.Board.Service                 : Factor w/ 6 levels "0","1","2","3",..: 6 5 6 3 5 4 5 6 6 4 ...
##  $ Leg.Room.Service                 : Factor w/ 6 levels "0","1","2","3",..: 4 5 6 4 5 5 5 5 6 6 ...
##  $ Baggage.Handling                 : Factor w/ 6 levels "0","1","2","3",..: 2 5 6 5 5 4 5 5 6 5 ...
##  $ Checkin.Service                  : Factor w/ 6 levels "0","1","2","3",..: 4 4 4 3 4 3 5 5 4 6 ...
##  $ Cleanliness                      : Factor w/ 6 levels "0","1","2","3",..: 4 5 6 4 5 4 5 6 6 5 ...
##  $ Online.Boarding                  : Factor w/ 6 levels "0","1","2","3",..: 5 2 6 6 4 2 6 5 5 2 ...
##  $ Departure.Delay.In.Minutes       : int  0 0 0 0 0 9 0 0 0 16 ...
##  $ Arrival.Delay.In.Minutes         : int  24 0 0 0 4 16 0 0 8 6 ...

## [1] "Number of NAs: 0"
```

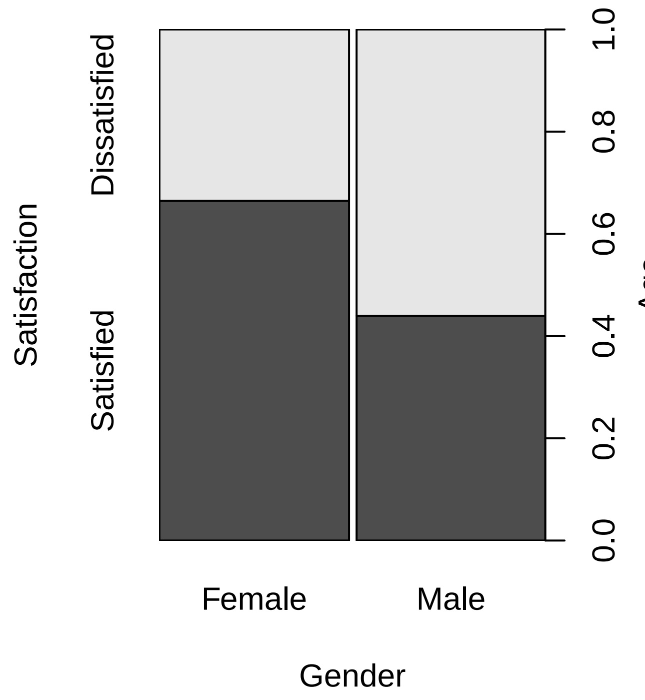
Graphs & Plots

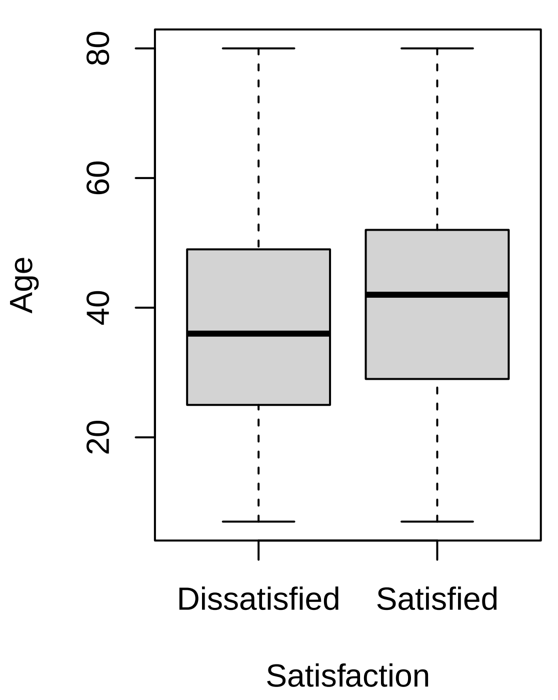
Plotting the data, we can see the relationships between various attributes (or lack thereof):

In the two graphs below, we are seeking to observe for a relationship between the customer's demographics and their satisfaction.

In the left-hand graph, we can observe that females were generally more satisfied with their flights than dissatisfied, as opposed to males who were generally more dissatisfied than satisfied. This may make for a good point of prediction.

In the right-hand graph, we can observe that those satisfied with their flight were, on average, older than those who were dissatisfied. However, the difference is very small, and the values fall within similar ranges, so it may not make for a good point of prediction.

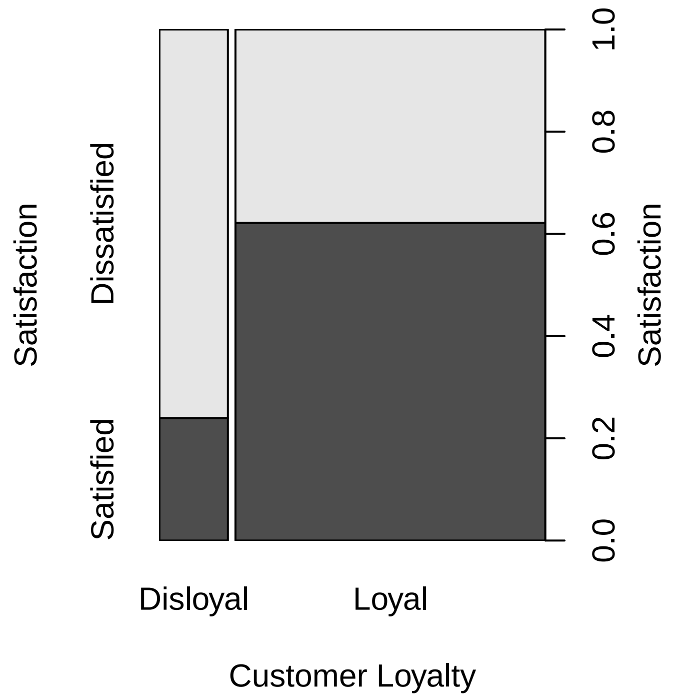


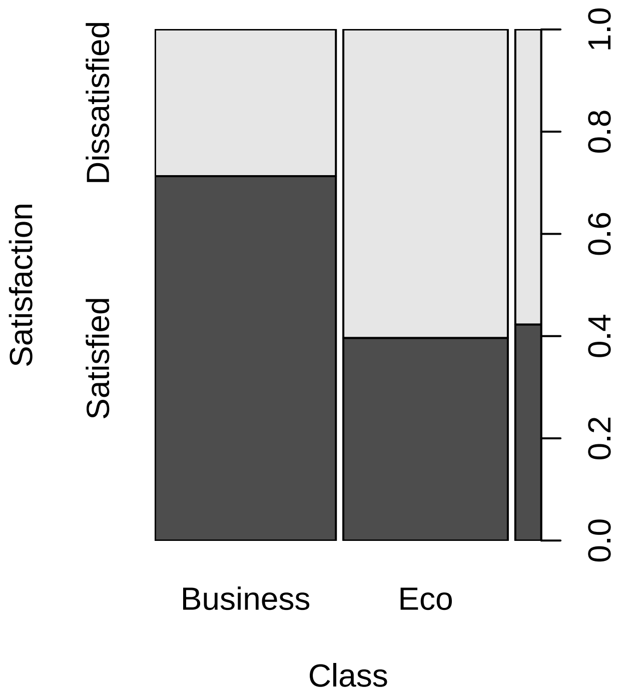


Furthermore, in the next two graphs below, we are seeking to determine if there is a observe for a relationship between the customer's classifications and their satisfaction.

In the left-hand graph, we can observe that loyal customers are significantly likely to be satisfied with their flight, while disloyal customers are significantly likely to be dissatisfied with their flight. The large difference may make a customer's loyalty a good predictor of satisfaction.

In the right-hand graph, we can observe that customers in the Business class are very likely to be satisfied with their flight, while customers in the Eco (Plus) classes are comparatively less likely to be satisfied with their flight. While Eco and Eco Plus lie more near the 50/50 mark, the comparative difference between their satisfaction and the Business class's satisfaction may make for a good point of prediction.

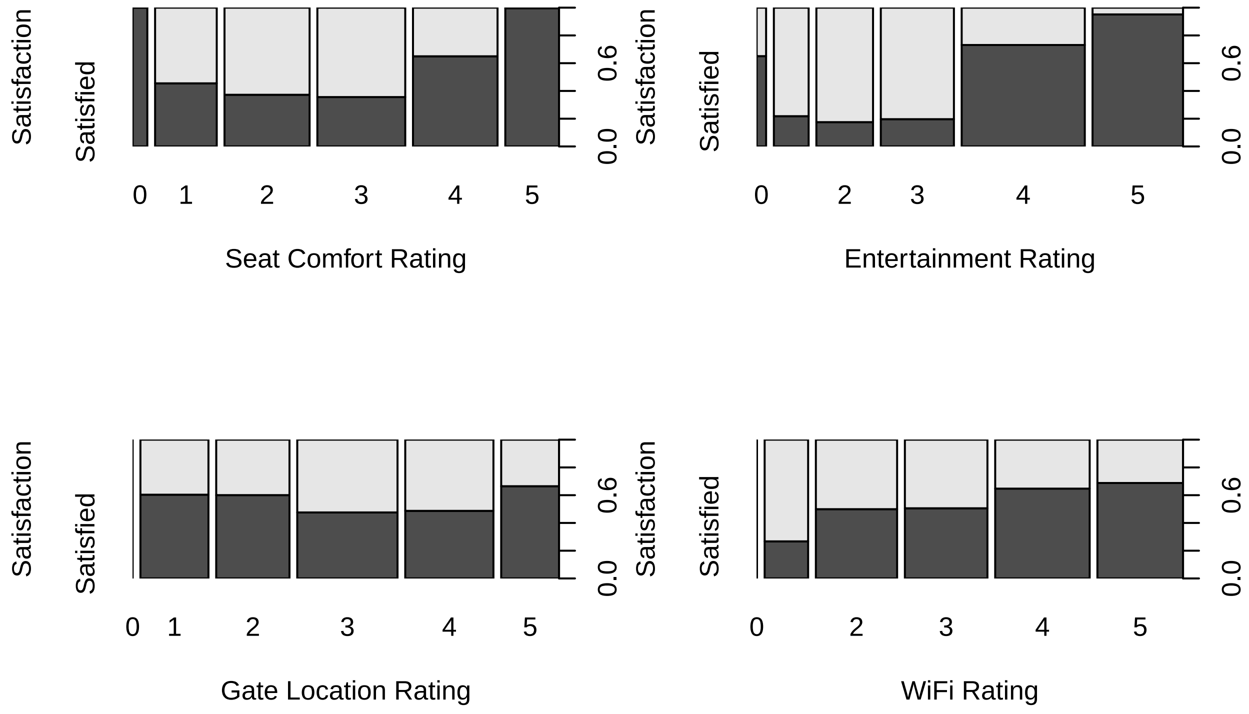




Finally, in the last four graphs below, we are seeking to determine if there is any correlation between the customer's review ratings and their satisfaction.

For obvious reasons, we can assume these will go hand-in-hand, but these graphs help show that generally, the lower the rating, the less likely people are to be satisfied, and the higher the rating, the more likely they are to be satisfied.

This is not true for *all* ratings, however. Such as the bottom-left graph, which implies that Gate Location has little effect on the customer's satisfaction with their flight.



Models

Model Training

SVM Linear

```r
svm1 <- svm(Satisfaction~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = Satisfaction ~ ., data = train, kernel = "linear",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  1745
##
```

```
##  ( 869 876 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dissatisfied Satisfied
```

SVM Polynomial

```r
svm2 <- svm(Satisfaction~., data=train, kernel="polynomial", cost=10, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = Satisfaction ~ ., data = train, kernel = "polynomial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  3635
##
##  ( 1792 1843 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dissatisfied Satisfied
```

SVM Radial

```r
svm3 <- svm(Satisfaction~., data=train, kernel="radial", cost=10, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = Satisfaction ~ ., data = train, kernel = "radial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  1808
##
##  ( 891 917 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dissatisfied Satisfied
```

Model Tuning

SVM Linear

```
# Tune model
tune_svm1 <- tune(svm, Satisfaction~., data=validate, kernel="linear",
                  ranges=list(cost=c(1, 5, 7, 10, 13, 15)))
summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1092233
##
## - Detailed performance results:
##   cost     error dispersion
## 1    1 0.1165563 0.02829712
## 2    5 0.1107677 0.02554768
## 3    7 0.1111538 0.02572608
## 4   10 0.1092233 0.02468746
## 5   13 0.1103816 0.02516481
## 6   15 0.1107677 0.02548276
```

```
# Extract best model
best_model_svm1 <- tune_svm1$best.model
summary(best_model_svm1)
```

```
##
## Call:
## best.tune(method = svm, train.x = Satisfaction ~ ., data = validate,
##     ranges = list(cost = c(1, 5, 7, 10, 13, 15)), kernel = "linear")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  657
##
##  ( 327 330 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dissatisfied Satisfied
```

SVM Polynomial

```
# Tune model
tune_svm2 <- tune(svm, Satisfaction~., data=validate, kernel="polynomial",
                 ranges=list(cost=c(0.01, 0.1, 0.5, 1, 2),
                             gamma=c(0.01, 0.025, 0.05, 0.075, 0.1)))
summary(tune_svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1 0.075
##
## - best performance: 0.09147164
##
## - Detailed performance results:
##    cost gamma      error dispersion
## 1  0.01 0.010 0.45079448 0.04145200
## 2  0.10 0.010 0.44963617 0.03987634
## 3  0.50 0.010 0.44886397 0.04149309
## 4  1.00 0.010 0.44809177 0.04190270
## 5  2.00 0.010 0.44423225 0.04233506
## 6  0.01 0.025 0.44963617 0.03987634
## 7  0.10 0.025 0.44616127 0.04128661
## 8  0.50 0.025 0.13816157 0.02182674
## 9  1.00 0.025 0.12619840 0.02140334
## 10 2.00 0.025 0.11307989 0.01771435
## 11 0.01 0.050 0.44770567 0.04120537
## 12 0.10 0.050 0.12967330 0.01939631
## 13 0.50 0.050 0.10497772 0.01799217
## 14 1.00 0.050 0.09494654 0.01526549
## 15 2.00 0.050 0.09224532 0.01732573
## 16 0.01 0.075 0.19336353 0.03299713
## 17 0.10 0.075 0.11076478 0.01854195
## 18 0.50 0.075 0.09185922 0.01617280
## 19 1.00 0.075 0.09147164 0.01918586
## 20 2.00 0.075 0.09841699 0.01875709
## 21 0.01 0.100 0.13391892 0.02005536
## 22 0.10 0.100 0.09726017 0.01724998
## 23 0.50 0.100 0.09185477 0.02007794
## 24 1.00 0.100 0.09995990 0.01894478
## 25 2.00 0.100 0.09957232 0.01887714
```

```
# Extract best model
best_model_svm2 <- tune_svm2$best.model
summary(best_model_svm2)
```

```
##
## Call:
## best.tune(method = svm, train.x = Satisfaction ~ ., data = validate,
##     ranges = list(cost = c(0.01, 0.1, 0.5, 1, 2), gamma = c(0.01,
##         0.025, 0.05, 0.075, 0.1)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  1166
##
##  ( 549 617 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dissatisfied Satisfied
```

SVM Radial

```
# Tune model
tune_svm3 <- tune(svm, Satisfaction~., data=validate, kernel="radial",
                  ranges=list(cost=c(1, 5, 7, 10, 13, 15),
                              gamma=c(0.01, 0.025, 0.05, 0.075, 0.1)))
summary(tune_svm3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
```

```
##
## - best parameters:
##  cost gamma
##    10 0.025
##
## - best performance: 0.08992426
##
## - Detailed performance results:
##    cost gamma      error dispersion
## 1     1 0.010 0.11386249 0.02048612
## 2     5 0.010 0.09919364 0.02155813
## 3     7 0.010 0.09610336 0.01981622
## 4    10 0.010 0.09532967 0.01968946
## 5    13 0.010 0.09494357 0.01986871
## 6    15 0.010 0.09494357 0.01884178
## 7     1 0.025 0.10034898 0.02281301
## 8     5 0.025 0.09455598 0.01708414
## 9     7 0.025 0.09030888 0.01336803
## 10   10 0.025 0.08992426 0.01466173
## 11   13 0.025 0.09339917 0.01418358
## 12   15 0.025 0.09532967 0.01576514
## 13    1 0.050 0.09494357 0.02367283
## 14    5 0.050 0.09417285 0.01458812
## 15    7 0.050 0.09533413 0.01729393
## 16   10 0.050 0.09495099 0.01751046
## 17   13 0.050 0.09533561 0.01611085
## 18   15 0.050 0.09572171 0.01698743
## 19    1 0.075 0.09610039 0.02484576
## 20    5 0.075 0.09417731 0.01851472
## 21    7 0.075 0.09340511 0.01746356
## 22   10 0.075 0.09224978 0.01753446
## 23   13 0.075 0.09186368 0.01803209
## 24   15 0.075 0.09186368 0.01803209
## 25    1 0.100 0.09648649 0.02488306
## 26    5 0.100 0.09263291 0.01893644
## 27    7 0.100 0.09224681 0.01871135
## 28   10 0.100 0.09224681 0.01871135
## 29   13 0.100 0.09224681 0.01871135
## 30   15 0.100 0.09224681 0.01871135
```

```
# Extract best model
best_model_svm3 <- tune_svm3$best.model
summary(best_model_svm3)
```

```
##
## Call:
## best.tune(method = svm, train.x = Satisfaction ~ ., data = validate,
##     ranges = list(cost = c(1, 5, 7, 10, 13, 15), gamma = c(0.01,
##         0.025, 0.05, 0.075, 0.1)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  864
##
##  ( 406 458 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dissatisfied Satisfied
```

Model Predictions

SVM Linear

Un-Tuned:

```
##
## pred_svm1      Dissatisfied Satisfied
##   Dissatisfied         1093       126
##   Satisfied             120      1251

## Accuracy:  0.905019305019305
## MCC:  0.809338812477781
```

Tuned:

```
##
## pred_svm1_t    Dissatisfied Satisfied
##   Dissatisfied         1097       129
##   Satisfied             116      1248

## Accuracy:  0.905405405405405
## MCC:  0.810211459466968
```

SVM Polynomial

Un-Tuned:

```
##
## pred_svm2      Dissatisfied Satisfied
##   Dissatisfied         1090       121
##   Satisfied             123      1256

## Accuracy:  0.905791505791506
## MCC:  0.810806914823568
```

Tuned:

```
##
## pred_svm2_t    Dissatisfied Satisfied
##   Dissatisfied         1101        85
##   Satisfied             112      1292
```

```
## Accuracy:  0.923938223938224
## MCC:  0.847247482030031
```

SVM Radial

Un-Tuned:

```
##
## pred_svm3      Dissatisfied Satisfied
##   Dissatisfied         1124        78
##   Satisfied              89      1299
```

```
## Accuracy:  0.935521235521236
## MCC:  0.87048507030758
```

Tuned:

```
##
## pred_svm3_t    Dissatisfied Satisfied
##   Dissatisfied         1103        82
##   Satisfied             110      1295
```

```
## Accuracy:  0.925868725868726
## MCC:  0.851136830836148
```

Analysis

Looking at each of the above prediction results, we can observe a number of things about each kernel mode. We'll discuss the results of each kernel in individually.

Linear

The un-tuned linear kernel SVM model utilized a cost of 10. We found that the model was able to predict with 90.5% accuracy in this un-tuned state.

In the tuned model, I ran the model through numerous cost values at first, covering a wider scope, and ended up simplifying it down to a narrower scope surrounding 10 (+/-5 from it). The best model uses a cost of 10 under the validation data. We can observe that there is a slight improvement to the model's accuracy after tuning- predicting with 90.54% accuracy - however the improvement is very minimal and can be considered the same as the accuracy in the un-tuned model.

The reason why there is even a difference at all is likely due to the fact that the validation data is smaller than that of the train data. It does seem to generalize hyperparameters well to the model, however, given the negligible difference in accuracy.

Polynomial

The un-tuned polynomial kernel SVM model utilized a cost of 10, as well as the default gamma value of 1. We found that the model was able to predict with 90.58% accuracy in this un-tuned state, being similar to that of the un-tuned linear model.

In the tuned model, I ran the model through numerous cost values at first, covering a wider scope, and ended up simplifying it down to a narrower scope, much like how I did for the previously discussed linear kernel. I did a similar thing for gamma values, and noticed the best value was more on the lower end (0.1), so I narrowed the range of it down from 0.01 through 0.1 for precision. The best model uses a cost of 1 and a gamma value of 0.075 under the validation data. We can observe that there is a notable improvement to the model's accuracy after tuning - predicting with 92.39% accuracy - and that there was a drop in both false positive and (especially) false negative rates.

It's worth noting that the improvements are notable this time likely due to the considerable difference in both cost and gamma values from the un-tuned model. If we were to re-run the new values on the un-tuned model, we can expect the rates to improve further from our tuned model, due to the transition from utilizing a smaller validation data, to a larger train data.

Radial

The un-tuned radial kernel SVM model utilized a cost of 10, as well as the default gamma value of 1. We found that the model was able to predict with 93.55% accuracy in this un-tuned state, which is a notable improvement to both of the previous un-tuned models.

In the tuned model, I approached tuning the same way as I did for the polynomial kernel - seeking to narrow the cost and gamma values down as I saw results, for increased accuracy - stopping at a reasonable place. The best model uses a cost of 10 and a gamma value of 0.025 under the validation data. We can observe that there is a decline in the model's accuracy after tuning - predicting with 92.59% accuracy - and that there was an increase in false negative and (especially) false positive rates.

I can only assume that the reason why the prediction accuracy is lower is because of the tuned model using a smaller selection of the data than the un-tuned model. I believe that if I were to run the un-tuned model using the best cost and gamma values observed by the tuned model, we'd observe an increase in model prediction accuracy rather than a decrease. I believe this tells us that for the radial kernel, the validation data does not do well at generalizing hyperparameters for the data as a whole.

Conclusion

It's clear that both the Polynomial and Radial kernels are best suited for creating an SVM model on the data set, with both of those kernels having a consistently better prediction accuracy over linear.

However, I believe Radial is best suitable overall due to the fact that, when you compare the un-tuned models and the tuned models separately between both kernels, you'll notice radial does consistently better at predicting overall. Additionally, the un-tuned radial model had the highest accuracy, which in my previous paragraph, I'd predicted would only be made higher by re-running the model using the better cost and gamma values that the tuned model utilized. Moreover, it makes sense for the Radial kernel to work best with the data set, due to the fact that the data set since the data is (presumably) not very linearly separable. Likely due to the nature of the out-of-5 rating data that's gathered, and how they're each fairly independent of one another.