STAT652 Midterm Report

Chun Yin Kong

3/10/2021

Initial Setup

Loading Data and Library

```
library(tictoc)
library(pacman)
p_load(titanic, tidyverse, janitor, naniar, DataExplorer, tidymodels)
data(titanic_train)
data(titanic_test)
set.seed(4)
```

Study the data pattern of both training and testing data set:

```
head(titanic_train)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                  Name    Sex Age SibSp Parch
## 1                             Braund, Mr. Owen Harris   male  22     1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                              Heikkinen, Miss. Laina female  26     0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 5                            Allen, Mr. William Henry   male  35     0     0
## 6                                    Moran, Mr. James   male  NA     0     0
##             Ticket    Fare Cabin Embarked
## 1        A/5 21171  7.2500              S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250              S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500              S
## 6           330877  8.4583              Q
```

```
head(titanic_test)
```

```
##   PassengerId Pclass                                         Name    Sex  Age
## 1         892      3                             Kelly, Mr. James   male 34.5
## 2         893      3             Wilkes, Mrs. James (Ellen Needs) female 47.0
## 3         894      2                    Myles, Mr. Thomas Francis   male 62.0
## 4         895      3                             Wirz, Mr. Albert   male 27.0
```

```
## 5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0
## 6         897      3                   Svensson, Mr. Johan Cervin   male 14.0
##   SibSp Parch  Ticket    Fare Cabin Embarked
## 1     0     0  330911  7.8292              Q
## 2     1     0  363272  7.0000              S
## 3     0     0  240276  9.6875              Q
## 4     0     0  315154  8.6625              S
## 5     1     1 3101298 12.2875              S
## 6     0     0    7538  9.2250              S
```

Preparing Data for model training

Training and Testing Data

```r
titanic_train <- titanic_train %>% clean_names()
#head(titanic_train)
```

Check for duplicate records/examples/rows in your dataset.

```r
#get_dupes(titanic_train)
```

```r
titanic_train2 <- titanic_train %>% select(-passenger_id, -name, -ticket, -cabin) %>%
  mutate(
    survived = as_factor(survived),
    pclass = as_factor(pclass),
    sex = as_factor(sex),
    embarked = as_factor(embarked)
  )

#head(titanic_train2)
```

```r
titanic_test <- titanic_test %>% clean_names()

#head(titanic_test)
```

Check for duplicate records/examples/rows in your dataset.

```r
#get_dupes(titanic_test)
```

```r
titanic_test2 <- titanic_test %>% select(-passenger_id, -name, -ticket, -cabin) %>%
  mutate(
    pclass = as_factor(pclass),
    sex = as_factor(sex),
    embarked = as_factor(embarked)
  )

#head(titanic_test2)
```

Make the first split with 80% of the data being in the training data set.

```r
titanic_train2_split <- initial_split(titanic_train2, prop = 0.8)
#titanic_train2_split
```

Create the recipe for applying the preprocessing. Note the use of step_nzv(), which removes any columns that have very low variability, and the use of the step_meanimpute() function, which fills in the cells that are missing with the mean of the column.

```
titanic_train2_recipe <- training(titanic_train2_split) %>%
  recipe(survived ~ .) %>%
  step_rm(pclass, sex, embarked) %>%
  step_nzv(all_predictors()) %>%
  step_meanimpute(age) %>%
  prep()

#summary(titanic_train2_recipe)

#tidy(titanic_train2_recipe)
```

Apply the receipe, so the *age* variable should be complete after the imputation.

```
titanic_train2_testing <- titanic_train2_recipe %>%
  bake(testing(titanic_train2_split))

#titanic_train2_testing
```

```
titanic_train2_training <- juice(titanic_train2_recipe)

#titanic_train2_training
```

Model Comparison

In this report, I will directly apply code with tuning in order to find the best parameters for each algorithms yielding the best result for model comparision. Then I will pick the model which perform the best to rerun the full titanic_train dataset and produce prediction for the titanic_test dataset.

0. Null Model

Training Model

```
model_null <- logistic_reg(mode = "classification") %>%
  set_engine("glm") %>%
  fit(survived ~ 1, data = titanic_train2_training)
```

Testing Model with Prediction

```
library(yardstick)
pred <- titanic_train2_testing %>%
  select(survived) %>%
  bind_cols(
  predict(model_null, new_data = titanic_train2_testing, type = "class")
  ) %>%
  rename(survived_null = .pred_class)
```

Evaluating the model performance

```
model_null %>%
  predict(titanic_train2_testing) %>%
  bind_cols(titanic_train2_testing) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 113  65
##          1   0   0
```

Accuracy and Metrics:

```r
result_null_2 <- metrics(pred, survived, survived_null) %>%
  mutate(algorithm = "Null Model")

metrics(pred, survived, survived_null)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.635
## 2 kap      binary         0
```

1. kNN with normalization

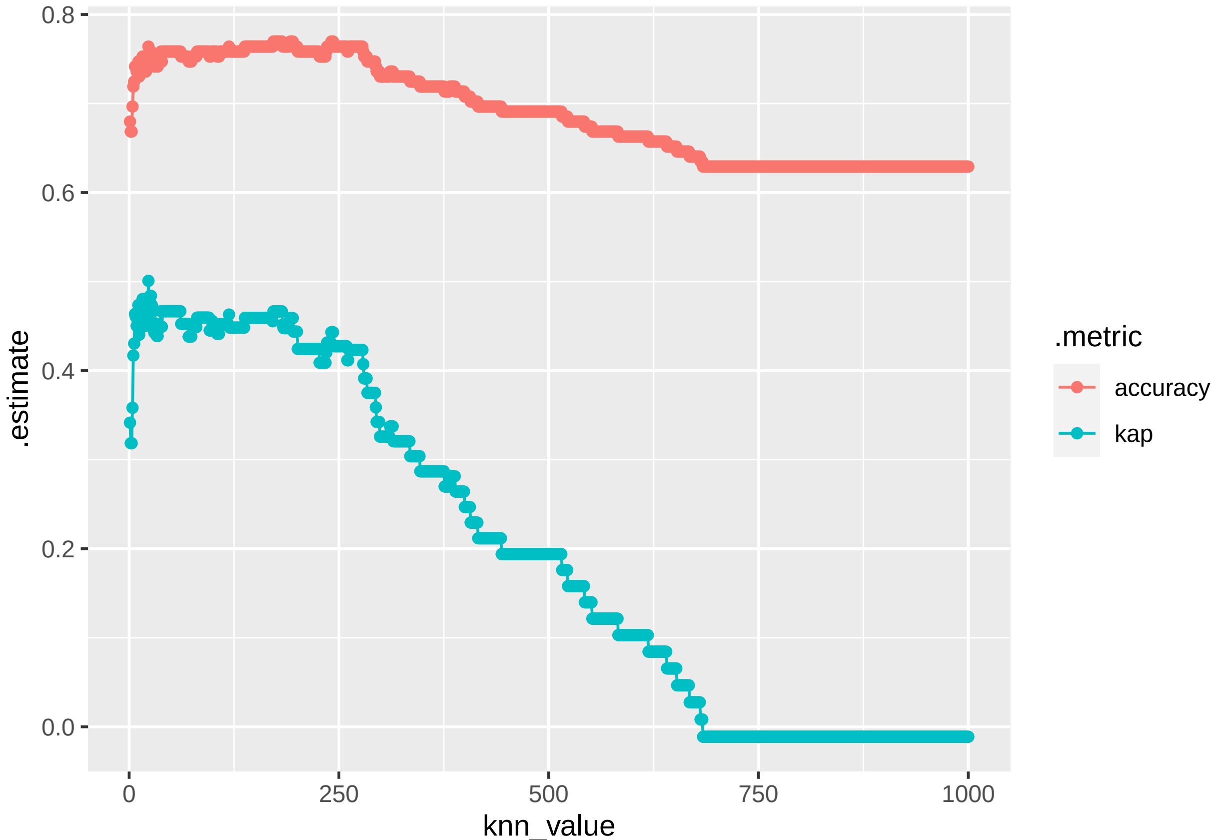
Tuning K parameter

In here I use for loop to test the k-nearest neighbor value from 1 to 1000

```
tic()
result_kNN <- data.frame()
for (i in 1:1000){

kNN_value <- c(i, i)

titanic_train2a_knn <- nearest_neighbor(neighbors = i) %>%
  set_engine("kknn", scale=TRUE) %>%
  set_mode("classification") %>%
  parsnip::fit(survived ~ ., data = titanic_train2_training)

result_temp <- titanic_train2a_knn %>%
  predict(titanic_train2_testing) %>%
  bind_cols(titanic_train2_testing) %>%
  metrics(truth = survived, estimate = .pred_class)

result_temp <- bind_cols(kNN_value, result_temp)

result_kNN <- bind_rows(result_kNN, result_temp)

}

result_kNN <- result_kNN %>%
  rename(knn_value = ...1)
toc()
```

```
## 200.184 sec elapsed
```

```
library(ggplot2)
ggplot(data = result_kNN, aes(y = .estimate, x = knn_value, col=.metric)) +
  geom_point() +
  geom_line()
```



Finding the k value yield the best accuracy and kappa:

```
result_kNN %>%
  filter(.estimate == max(.estimate))
```

```
##    knn_value  .metric .estimator .estimate
## 1        172 accuracy     binary 0.7696629
## 2        173 accuracy     binary 0.7696629
## 3        174 accuracy     binary 0.7696629
## 4        175 accuracy     binary 0.7696629
## 5        176 accuracy     binary 0.7696629
## 6        177 accuracy     binary 0.7696629
## 7        178 accuracy     binary 0.7696629
## 8        179 accuracy     binary 0.7696629
## 9        180 accuracy     binary 0.7696629
## 10       181 accuracy     binary 0.7696629
## 11       182 accuracy     binary 0.7696629
## 12       192 accuracy     binary 0.7696629
## 13       193 accuracy     binary 0.7696629
## 14       194 accuracy     binary 0.7696629
## 15       195 accuracy     binary 0.7696629
## 16       241 accuracy     binary 0.7696629
## 17       242 accuracy     binary 0.7696629
## 18       243 accuracy     binary 0.7696629
```

kNN Model with best k-value

```r
result_kNN_2 <- result_kNN %>%
  filter(knn_value == "173") %>%
  mutate(algorithm = "kNN")

result_kNN_2
```

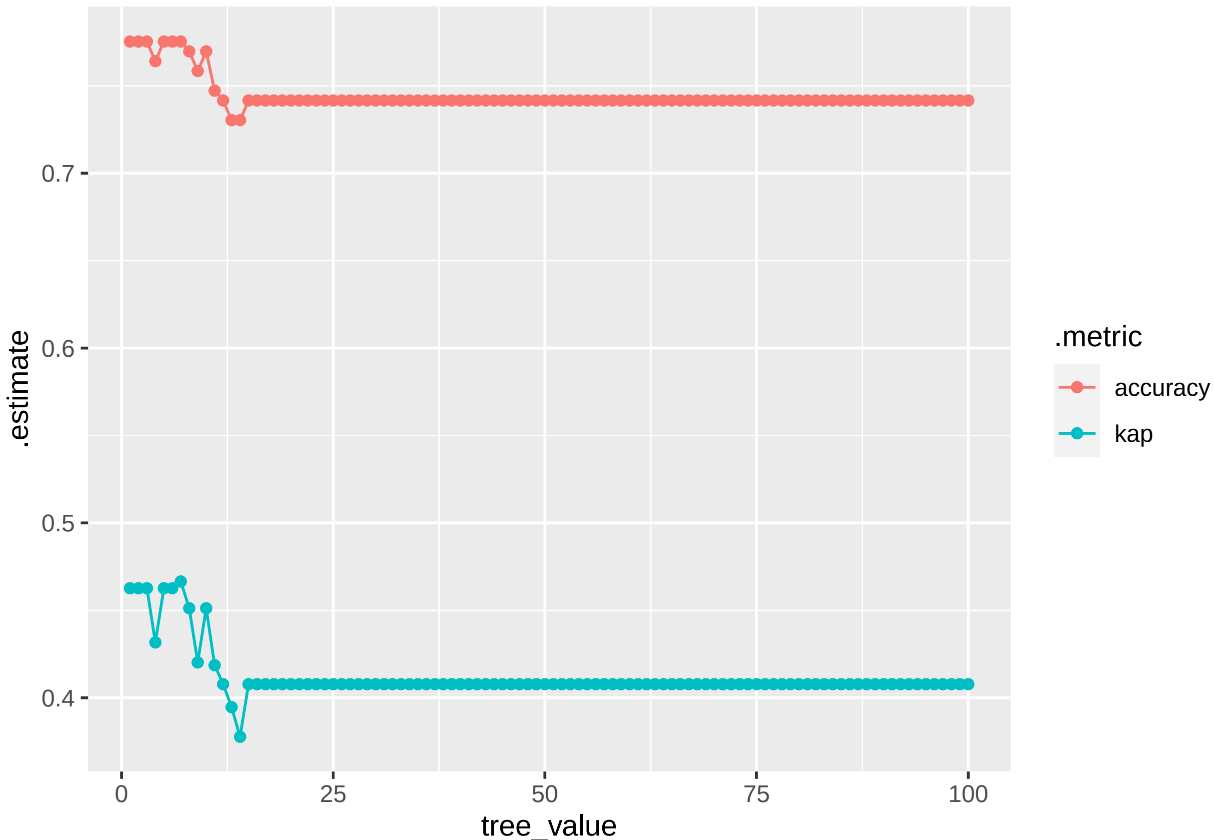
```
##   knn_value  .metric .estimator .estimate algorithm
## 1       173 accuracy     binary 0.7696629       kNN
## 2       173      kap     binary 0.4665985       kNN
```

2. Boosted C5.0

Tuning Number of Trees in Boosted C5.0

```
result_boosted_c50 <- data.frame()

for (i in 1:100){
  tree_value <- c(i, i)
  titanic_train2a_C50 <- boost_tree(trees = i) %>%
  set_engine("C5.0") %>%
  set_mode("classification") %>%
  fit(survived ~ ., data = titanic_train2_training)

  result_temp <- titanic_train2a_C50 %>%
    predict(titanic_train2_testing) %>%
    bind_cols(titanic_train2_testing) %>%
    metrics(truth = survived, estimate = .pred_class)

  result_temp <- bind_cols(tree_value, result_temp)

  result_boosted_c50 <- bind_rows(result_boosted_c50, result_temp)

}

result_boosted_c50 <- result_boosted_c50 %>%
  rename(tree_value = ...1)
```

```
ggplot(data = result_boosted_c50, aes(y = .estimate, x = tree_value, col=.metric)) +
  geom_point() +
  geom_line()
```



Boosted Tree model with best number of tree

Finding the number of trees value yield the best accuracy and kappa:

```
result_boosted_c50 %>%
  filter(.estimate == max(.estimate))
```

```
##   tree_value  .metric .estimator .estimate
## 1          1 accuracy     binary 0.7752809
## 2          2 accuracy     binary 0.7752809
## 3          3 accuracy     binary 0.7752809
## 4          5 accuracy     binary 0.7752809
## 5          6 accuracy     binary 0.7752809
## 6          7 accuracy     binary 0.7752809
```

```
result_boosted_c50_2 <- result_boosted_c50 %>%
  filter(tree_value == "6") %>%
  mutate(algorithm = "Boosted C5.0")

result_boosted_c50_2
```

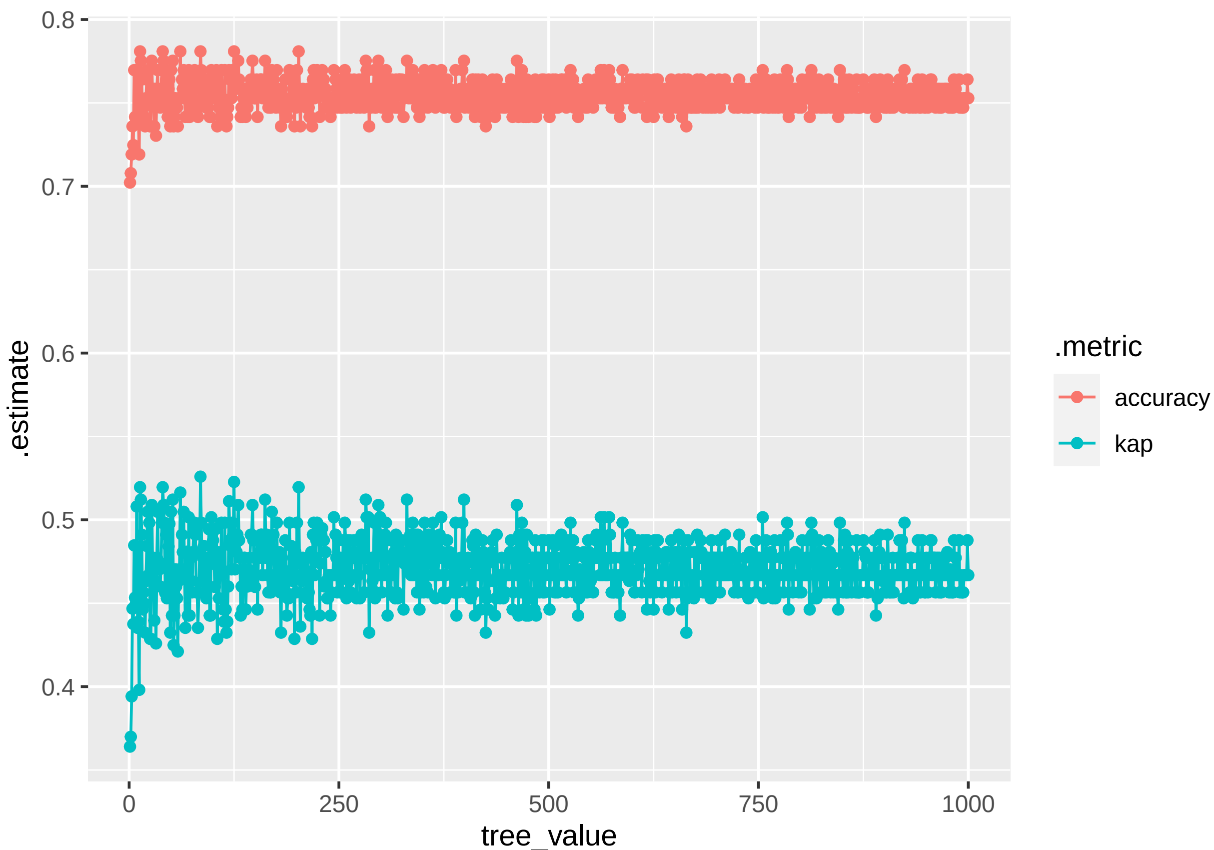
```
##   tree_value  .metric .estimator .estimate    algorithm
## 1          6 accuracy     binary 0.7752809 Boosted C5.0
## 2          6      kap     binary 0.4626415 Boosted C5.0
```

3. Random Forest

Tuning Number of Trees in Random Forest

```
result_random_forest <- data.frame()

for (i in 1:1000){
  tree_value <- c(i, i)
  titanic_train2a_ranger <- rand_forest(trees = i) %>%
  set_engine("ranger") %>%
  set_mode("classification") %>%
  fit(survived ~ ., data = titanic_train2_training)

  result_temp <- titanic_train2a_ranger %>%
    predict(titanic_train2_testing) %>%
    bind_cols(titanic_train2_testing) %>%
    metrics(truth = survived, estimate = .pred_class)

  result_temp <- bind_cols(tree_value, result_temp)

  result_random_forest <- bind_rows(result_random_forest, result_temp)

}

result_random_forest <- result_random_forest %>%
  rename(tree_value = ...1)
```

```
ggplot(data = result_random_forest, aes(y = .estimate, x = tree_value, col=.metric)) +
  geom_point() +
  geom_line()
```



Finding the number of trees value yield the best accuracy and kappa:

```
result_random_forest %>%
  filter(.estimate == max(.estimate))
```

```
##   tree_value  .metric .estimator .estimate
## 1         13 accuracy     binary 0.7808989
## 2         40 accuracy     binary 0.7808989
## 3         61 accuracy     binary 0.7808989
## 4         85 accuracy     binary 0.7808989
## 5        125 accuracy     binary 0.7808989
## 6        202 accuracy     binary 0.7808989
```

```
result_random_forest_2 <- result_random_forest %>%
  filter(tree_value == "85") %>%
  mutate(algorithm = "Random Forest")

result_random_forest_2
```

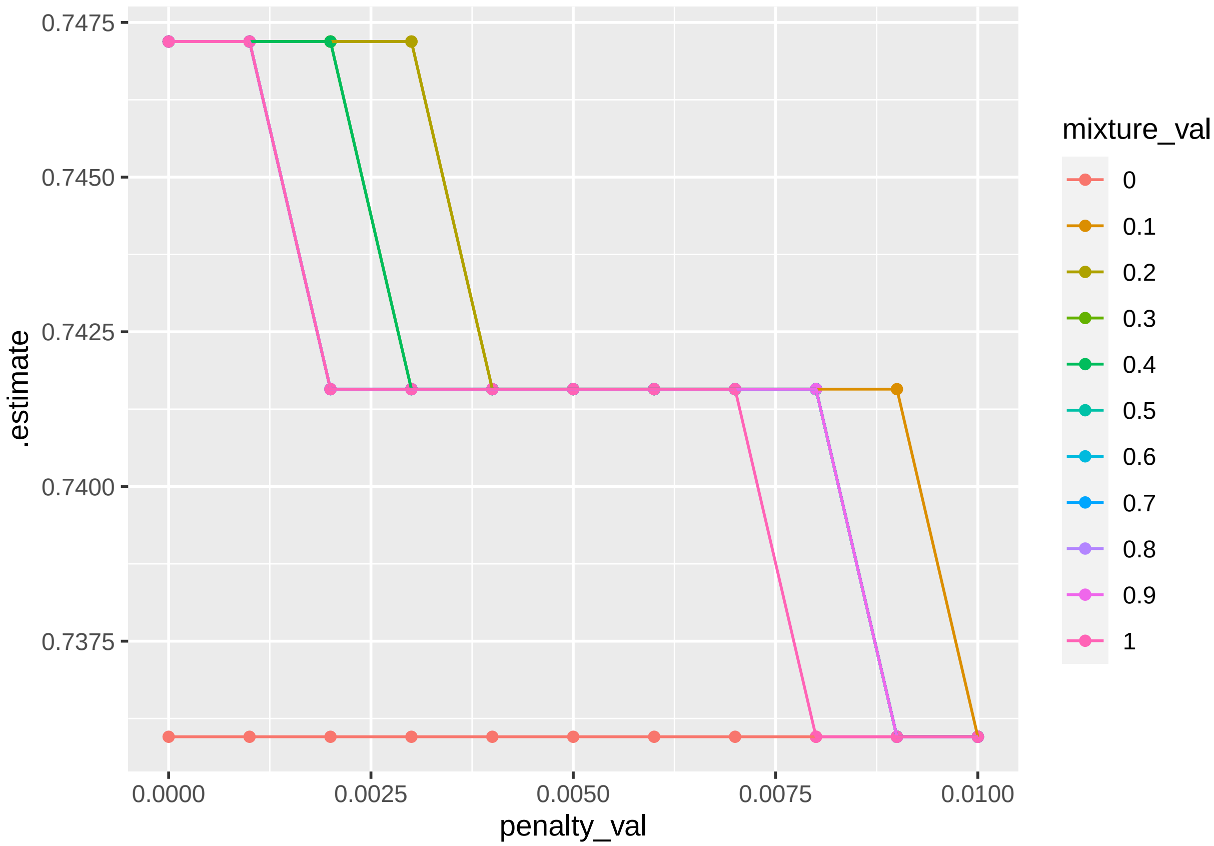
```
##   tree_value  .metric .estimator .estimate     algorithm
## 1         85 accuracy     binary 0.7808989 Random Forest
## 2         85      kap     binary 0.5258844 Random Forest
```

4. Logistic Regression using regularization

Tuning penalty and mixture

```
library(discrim)
result_log_reg <- data.frame()

for (j in seq(0, 1, by=0.1)){
for (i in seq(0, 0.01, by=0.001)){
  penalty_val <- c(i, i)
  mixture_val <- c(j, j)
  titanic_train2a_glm <- logistic_reg(penalty = i, mixture = j) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(survived ~ ., data = titanic_train2_training)

  result_temp <- titanic_train2a_glm %>%
    predict(titanic_train2_testing) %>%
    bind_cols(titanic_train2_testing) %>%
    metrics(truth = survived, estimate = .pred_class)

  result_temp <- bind_cols(penalty_val, mixture_val, result_temp)

  result_log_reg <- bind_rows(result_log_reg, result_temp)

}
}

result_log_reg <- result_log_reg %>%
  rename(penalty_val = ...1) %>%
  rename(mixture_val = ...2) %>%
  mutate(mixture_val = as.factor(mixture_val))
```

```
result_log_reg1a <- result_log_reg %>%
  filter(.metric == "accuracy")

ggplot(data = result_log_reg1a, aes(y = .estimate, x = penalty_val, col=mixture_val)) +
  geom_point() +
  geom_line()
```



From the above short comparison, we find that although changing the penalty and mixture value of the logistic regression will affect the accuracy, but the difference in accuracy value is so small that it is not so significant in changing the penalty and mixture in this data set. Hence below is a random pick of penalty to produce the accuracy result. For the value of mixture (L1 Regularization), it will change when it is greater than 0.5 or less than 0.5 for each penalty value. i.e. the tuning point of accuracy of each penalty value is in mixture value = 0.5. Hence I set the mixture value to 0.5 to give the highest accuracy for the logistic regression prediction.

```
result_log_reg_2 <- logistic_reg(penalty = 0.001, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(survived ~ ., data = titanic_train2_training) %>%
  predict(titanic_train2_testing) %>%
  bind_cols(titanic_train2_testing) %>%
  metrics(truth = survived, estimate = .pred_class) %>%
  mutate(penalty_val = 0.001,
         mixture_val = 0.5,
         algorithm = "Logistic Regression with Regularization")

result_log_reg_2
```

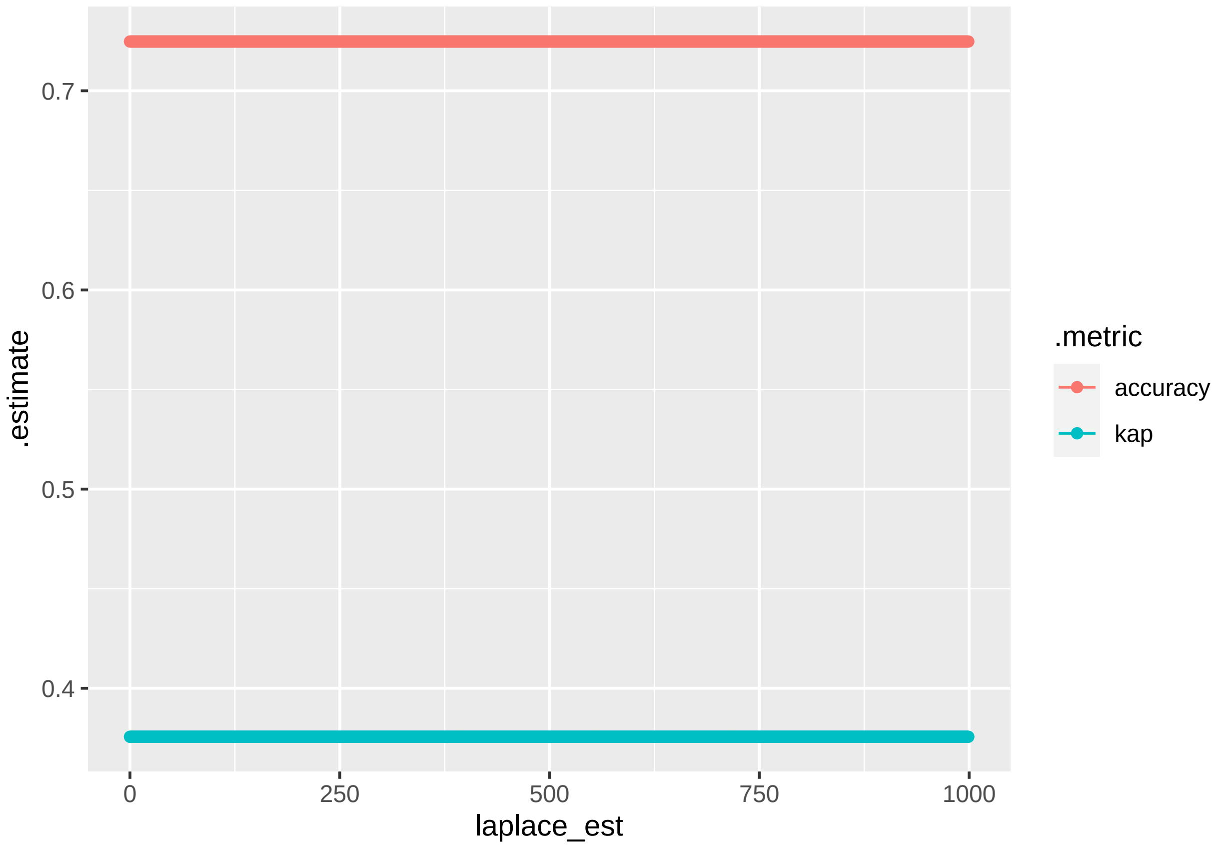
```
## # A tibble: 2 x 6
##   .metric  .estimator .estimate penalty_val mixture_val algorithm
##   <chr>    <chr>          <dbl>       <dbl>       <dbl> <chr>
## 1 accuracy binary         0.747       0.001         0.5 Logistic Regression wit~
## 2 kap      binary         0.375       0.001         0.5 Logistic Regression wit~
```

5. Naive Bayes

Tuning Laplace Estimator

```
library(discrim)
result_naive_bayes <- data.frame()

for (i in 0:999){
  laplace_est <- c(i, i)
  titanic_train2a_nb <- naive_Bayes(Laplace = i) %>%
  set_engine("klaR") %>%
  set_mode("classification") %>%
  fit(survived ~ ., data = titanic_train2_training)

  result_temp <- titanic_train2a_nb %>%
    predict(titanic_train2_testing) %>%
    bind_cols(titanic_train2_testing) %>%
    metrics(truth = survived, estimate = .pred_class)

  result_temp <- bind_cols(laplace_est, result_temp)

  result_naive_bayes <- bind_rows(result_naive_bayes, result_temp)

}

result_naive_bayes <- result_naive_bayes %>%
  rename(laplace_est = ...1)
```

```
ggplot(data = result_naive_bayes, aes(y = .estimate, x = laplace_est, col=.metric)) +
  geom_point() +
  geom_line()
```



From the graph above we can see that the accuracy doesn't change when the Laplace Esimator varies. Hence in this model I will pick the Laplace Estimator of 0 for the best parameter for this model, that giving best accuracy.

```
result_naive_bayes_2 <- result_naive_bayes %>%
  filter(laplace_est == "0") %>%
  mutate(algorithm = "Naive Bayes")

result_naive_bayes_2
```

```
##   laplace_est  .metric .estimator .estimate   algorithm
## 1           0 accuracy     binary 0.7247191 Naive Bayes
## 2           0      kap     binary 0.3756621 Naive Bayes
```

Comparing all models

```
result_all_models <- bind_rows(result_null_2, result_kNN_2, result_boosted_c50_2, result_random_forest_
result_all_models
```

```
## # A tibble: 12 x 9
##    .metric  .estimator .estimate algorithm      knn_value tree_value penalty_val
##    <chr>    <chr>          <dbl> <chr>              <int>      <int>       <dbl>
##  1 accuracy binary         0.635 Null Model            NA         NA       NA
##  2 kap      binary         0     Null Model            NA         NA       NA
##  3 accuracy binary         0.770 kNN                  173         NA       NA
##  4 kap      binary         0.467 kNN                  173         NA       NA
##  5 accuracy binary         0.775 Boosted C5.0          NA          6       NA
##  6 kap      binary         0.463 Boosted C5.0          NA          6       NA
```

```
##  7 accuracy binary         0.781 Random Forest         NA         85       NA
##  8 kap      binary         0.526 Random Forest         NA         85       NA
##  9 accuracy binary         0.747 Logistic Regr~        NA         NA        0.001
## 10 kap      binary         0.375 Logistic Regr~        NA         NA        0.001
## 11 accuracy binary         0.725 Naive Bayes           NA         NA       NA
## 12 kap      binary         0.376 Naive Bayes           NA         NA       NA
## # ... with 2 more variables: mixture_val <dbl>, laplace_est <int>
```

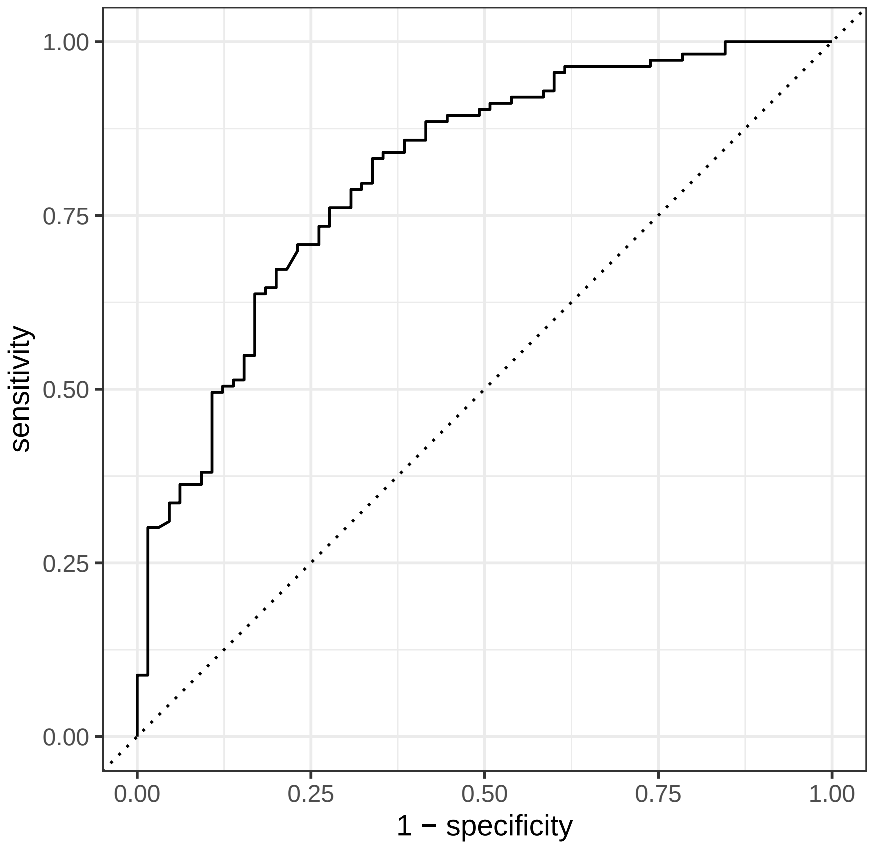
From the table above we see that after running all models, random forest algorithm gives the highest accuracy in the titanic data set. Hence in the following pages I will use random forest to rerun the full titanic data set and for model evaluation.

ROC Curves

```
null_roc <- model_null %>%
  predict(titanic_train2_testing, type = "prob") %>%
  bind_cols(titanic_train2_testing) %>%
  roc_curve(survived, .pred_0)

knn_roc <- titanic_train2a_knn %>%
  predict(titanic_train2_testing, type = "prob") %>%
  bind_cols(titanic_train2_testing) %>%
  roc_curve(survived, .pred_0)

c50_roc <- titanic_train2a_C50 %>%
  predict(titanic_train2_testing, type = "prob") %>%
  bind_cols(titanic_train2_testing) %>%
  roc_curve(survived, .pred_0)

rf_roc <- titanic_train2a_ranger %>%
  predict(titanic_train2_testing, type = "prob") %>%
  bind_cols(titanic_train2_testing) %>%
  roc_curve(survived, .pred_0)


glm_roc <- titanic_train2a_glm %>%
  predict(titanic_train2_testing, type = "prob") %>%
  bind_cols(titanic_train2_testing) %>%
  roc_curve(survived, .pred_0)


nb_roc <- titanic_train2a_nb %>%
  predict(titanic_train2_testing, type = "prob") %>%
  bind_cols(titanic_train2_testing) %>%
  roc_curve(survived, .pred_0)
```

ROC Curve for Best Model - Random Forest

```
autoplot(rf_roc)
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity

Rerun Full Titanic Dataset with Best Model

```
titanic_train3 <- titanic_train2 %>%
  na.omit()

titanic_test3 <- titanic_test2 %>%
  na.omit()

titanic_test4 <- titanic_test %>%
  na.omit() %>%
  select(passenger_id)

titanic_rf_full_model <- rand_forest(trees = 85) %>%
  set_engine("ranger") %>%
  set_mode("classification") %>%
  fit(survived ~ ., data = titanic_train3)

result_rf_full <- titanic_rf_full_model %>%
  predict(titanic_test3) %>%
  bind_cols(titanic_test4)
```

```
result_rf_full <- result_rf_full[, c(2, 1)]
result_rf_full <- result_rf_full %>%
  rename(PassengerId = passenger_id,
         Survived = .pred_class)

head(result_rf_full)
```

```
## # A tibble: 6 x 2
##   PassengerId Survived
##         <int> <fct>
## 1         892 0
## 2         893 0
## 3         894 0
## 4         895 0
## 5         896 0
## 6         897 0
```

```
write.csv(result_rf_full, "titanic_full_prediction.csv")
```