Titanic Exploratory Analysis

Brian Moore

Amalysis Scenario

- Let's assumes we're responsible for supporting the Titanic disaster investigation using data insights.
- In this exploratory report we'll use data analysis and modeling techniques to see if certain factors might have correlated with passengers being more likely to perish.

```
options(scipen=999)

# warning: running this will install packages on your machine if they aren't present
required_packages <- c('tidyverse', 'caret', 'faraway', 'pdp', 'ranger', 'Hmisc')
for(p in required_packages) {
  if(!require(p,character.only = TRUE))
        install.packages(p, repos = "http://cran.us.r-project.org")
  library(p,character.only = TRUE)
}
pct_formater_1 <- scales::label_percent(accuracy = 1)
```

Access the Titanic data

```
### read in the titanic dataset from URL location then store CSV in working dir
### only need to run once
url <- c("http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3.csv")
# t3_df_raw <- read_csv(url)
# write_csv(t3_df_raw, "titanic_3_dataset.csv")

### drop boat and body variables as these variables leak
### information about the target variable survival
### instead, we use information one might have had prior to the disaster start
df_raw <- read_csv("titanic_3_dataset.csv") %>%
  dplyr::select(-boat, - body)
```

Check for missing data

```
df_raw %>%
  map_df(~sum(is.na(.))) %>%
  gather(key="variable", value="NA_count") %>%
  filter(NA_count>=1) %>%
  mutate(percent_total_rows_NA = pct_formater_1(NA_count/nrow(df_raw))) %>%
  arrange(desc(percent_total_rows_NA))
```

```
## # A tibble: 5 x 3
##   variable  NA_count percent_total_rows_NA
##   <chr>        <int> <chr>
## 1 cabin         1014 77%
```

```
## 2 home.dest      564 43%
## 3 age            263 20%
## 4 fare             1 0%
## 5 embarked         2 0%
```

Handling NAs

- when Cabin or home.dest is NA then replace with missing
- when fare missing then replace with median for now
- replace embarked NA with most common embarked location

```
df <- df_raw %>%
  mutate_at(c("cabin", "home.dest"), replace_na, "missing") %>%
  mutate_at(c("embarked"), replace_na, "S") %>%
  mutate(fare = if_else(is.na(fare), median(df_raw$fare, na.rm = T), fare))

# check NAs
# df  %>%
#   map_df(~sum(is.na(.))) %>%
#   gather(key="variable", value="NA_count") %>%
#   filter(NA_count>=1) %>%
#   mutate(percent_total_rows_NA = pct_formater_1(NA_count/nrow(df))) %>%
#   arrange(desc(percent_total_rows_NA))
```

Adding derived new features

- Add a column for cabin deck (m will be returned for observations where cabin in missing).
- Hypothesis: iceberg crash occurred late at night. decks closer to lifeboats could have better chance for survival?
- Use passenger name to derive new name features surname and title.

```
df <- df %>%
  rowwise() %>%
  mutate(cabin_deck = str_extract(cabin, "^.{1}"),
         cabin_prefix_2 = replace_na(str_extract(cabin, "^.{2}"), "Other"),
         cabin_count = str_count(cabin, ' '),
         surname = strsplit(name, split=",")[[1]][1],
         title = strsplit(strsplit(name, split=",")[[1]][2], split="\\.")[[1]][1]) %>%
  ungroup()
```

- Surname count: could potentially help capture signal for families

```
df <- df %>%
  group_by(surname) %>%
  mutate(surname_count = n()) %>%
  ungroup()
```

- Create new ticket feature to see if there's some signal in the ticket variable.
- This might result in a random feature or could contribute to overfitting.

```
df <- df %>%
  mutate(ticket_prefix_1 = substring(ticket, 1, 1),
         ticket_prefix_2 = substring(ticket, 1, 2))
```

- Predict age when missing and add in age bucket (for visualizations)

```
# add observation id
df <- df %>% mutate(id = row_number())

# get data to train model to predict age
age_df <- df %>%
  filter(!is.na(age)) %>%
  select(age, sibsp, sex, fare, embarked, title, pclass)

# linear model to predict age for missing records
age_mod <- train(age ~ .,
        data=age_df,
        method="lm",
        trControl=trainControl(method = "cv", number = 3))

# age preds for missing records
missing_age <- df %>%
  filter(is.na(age)) %>%
  mutate(age_pred = predict(age_mod, newdata = df %>% filter(is.na(age))))

# join in age preds with full dataset
df <- df %>%
  left_join(missing_age %>% select(id, age_pred ),
            by=c("id")) %>%
  mutate(age = ifelse(is.na(age), age_pred, age),
         age_bucket = case_when(
          age <= 12 ~ 'Baby to Gradeschool',
          age <= 19 ~ 'Teen',
          age <= 29 ~ 'Twenties',
          age <= 39 ~ 'Thirties',
          age <= 49 ~ 'Forties',
          age <= 59 ~ 'Fifties',
          age > 59 ~ 'Senior',
          ),
         age_bucket = factor(age_bucket, levels = c('Baby to Gradeschool',
                'Teen', 'Twenties', 'Thirties',
                'Forties', 'Fifties', 'Senior'))
         ) %>%
  select(-id, -age_pred)
```

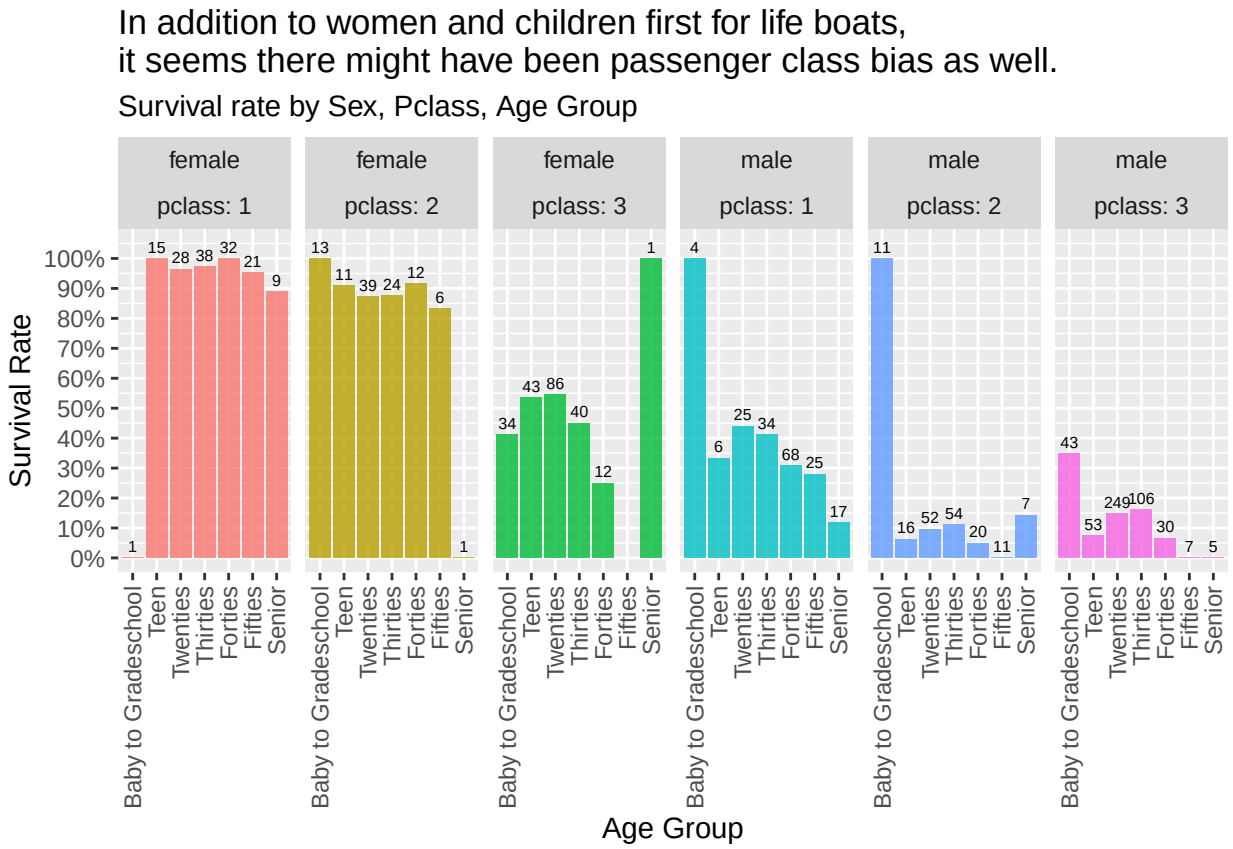
Exploratory Analysis

- top visualizations from EDA iterations

Sex, age, pclass

```
df %>%
  group_by(sex, age_bucket, pclass) %>%
  summarise(passenger_count = n(),
            survival_rate = mean(survived)) %>%
  ungroup() %>%
  mutate(pclass_2 = paste0("pclass: ", pclass),
    fill_key = paste0(sex, "_", pclass)) %>%
  ggplot(aes(x=age_bucket, y=survival_rate)) +
  geom_col(aes(fill=fill_key), alpha=0.8) +
```

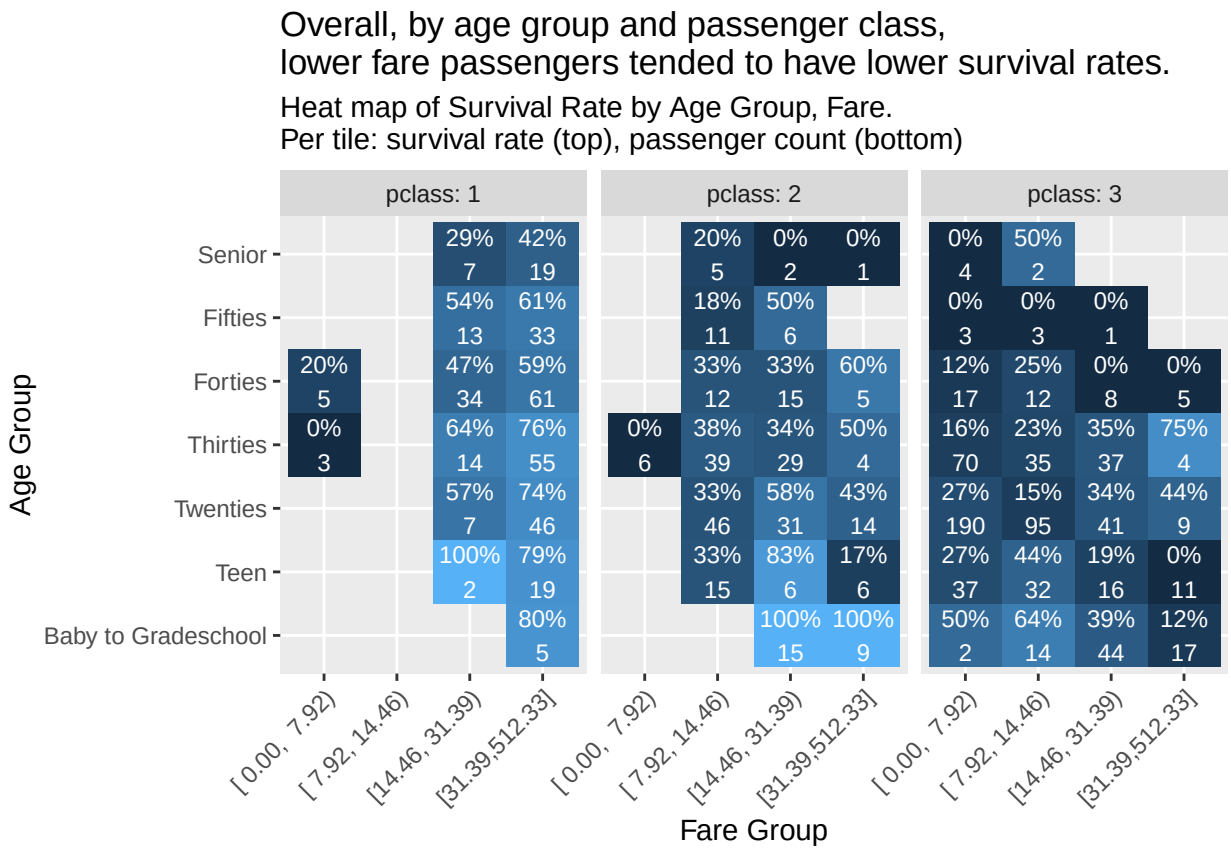
```
  geom_text(aes(label=passenger_count), vjust=-0.5, size=2) +
  facet_grid(. ~ sex + pclass_2) +
  scale_y_continuous(expand = expansion(mult = c(0.05, .1)),
                     labels = scales::percent_format(accuracy=1),
                     breaks = seq(0,1,by=0.1)) +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(title="In addition to women and children first for life boats,
it seems there might have been passenger class bias as well.",
       subtitle = "Survival rate by Sex, Pclass, Age Group",
       y="Survival Rate",
       x="Age Group")
```



Pclass, age, fare

```
df %>%
  group_by(pclass, age_bucket, fare_bucket = cut2(fare, g=4)) %>%
  summarise(passenger_count = n(),
            survival_rate = mean(survived)) %>%
  ggplot(aes(x=fare_bucket, y=age_bucket)) +
  geom_tile(aes(fill=survival_rate)) +
  geom_text(aes(label=pct_formater_1(survival_rate)), size=3,
            vjust=-0.5, color="white") +
  geom_text(aes(label=passenger_count), size=3, vjust=1.5, color="white") +
  facet_grid(. ~ paste0("pclass: ", pclass)) +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
```

```
    labs(title="Overall, by age group and passenger class,
lower fare passengers tended to have lower survival rates.",
       subtitle = "Heat map of Survival Rate by Age Group, Fare.
Per tile: survival rate (top), passenger count (bottom)",
       y="Age Group",
       x="Fare Group")
```



Prep Data for Modeling - using cross validation to estimate test error - not using separate test set as it reduces the training data size - for this analysis we're looking to surface trends in the passenger data vs make future predictions (hence why we use the full dataset for training)

```
# drop variables that would need further transformation before being useful to the model
df_final <- df %>%
  ### model binary factor var
  mutate(survived_fct = factor(survived)) %>%
  select(-name, -ticket, -cabin, -home.dest, -surname, -survived, -age_bucket) %>%
  mutate_if(is.character, as.factor)

# labeling sparse factor levels to other
# helps prevent model errors downstream
df_final <- df_final %>%
  mutate_if(is.factor, list(~fct_lump_min(f=., min=25)))

# drop near zero variable features
df_final  <- df_final[, -nearZeroVar(df_final)]
```

Modeling

- 5 fold cross validation to assess model accuracy vs null model
- use the same seed before each model run so the model trains on same folds for each model

```
ctrl <- trainControl(method = "cv", number = 5)
```

Create a baseline model that only uses the sex variable to predict survival.

```
# ~78% training data accuracy predicting all females survive
set.seed(123)
baseline_mod <- train(survived_fct ~ .,
                      data=df_final %>% dplyr::select(sex, survived_fct),
                      method="rpart",
                      tuneLength = 1,
                      tuneGrid = data.frame(cp=0.1),
                      trControl = ctrl)
baseline_mod
```

```
## CART
##
## 1309 samples
##    1 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1048, 1047, 1047, 1047, 1047
## Resampling results:
##
##   Accuracy   Kappa
##   0.7800006  0.5276824
##
## Tuning parameter 'cp' was held constant at a value of 0.1
```

Logistic Regression

```
set.seed(123)
logistic_reg_mod <- train(survived_fct ~ .,
                          data=df_final,
                          method="glm",
                          family="binomial",
                          preProcess=c("center", "scale"),
                          trControl = ctrl)

# warning: most VIF scores are very HIGH so multicollinearity should be considered
# vif(logistic_reg_mod$finalModel)
```

Lasso Logistic Regression - Ridge regression (or alpha = 0) - Lasso regression (or alpha = 1) - Lambda: the strength of the penalty on the coefficients. Larger the lambda value the more freedom to shrink coefficients towards zero which could end up resulting in an intercept only model.

```
set.seed(123)
lasso_logistic_reg_mod <- train(survived_fct ~ .,
                      data=df_final,
                      method="glmnet",
                      tuneGrid=expand.grid(.alpha=1,
                                           .lambda=10^seq(5, -5, length = 20)),
```

```
                    preProcess=c("center", "scale"),
                    trControl = ctrl)
```

Ridge Logistic Regression

```
set.seed(123)
ridge_logistic_reg_mod <- train(survived_fct ~ .,
                    data=df_final,
                    method="glmnet",
                    tuneGrid=expand.grid(.alpha=0,
                                         .lambda=10^seq(5, -5, length = 20)),
                    preProcess=c("center", "scale"),
                    trControl = ctrl)
```

Mix of Lasso and Ridge Logistic Regression

```
set.seed(123)
glmnet_mod <- train(survived_fct ~ .,
                    data=df_final,
                    method="glmnet",
                    tuneGrid=expand.grid(.alpha=seq(0,1,by=0.1),
                                         .lambda=10^seq(1, -5, length = 10)),
                    preProcess=c("center", "scale"),
                    trControl = ctrl)
```

LDA

```
set.seed(123)
lda_mod <- train(survived_fct ~ .,
                   data=df_final,
                   method="lda",
                   preProcess=c("center", "scale"),
                   trControl = ctrl)
```

KNN

```
set.seed(123)
knn_mod <- train(survived_fct ~ .,
                  data=df_final,
                  method = "knn",
                  trControl = ctrl,
                  preProcess = c("center","scale"),
                  tuneLength = 20)
```

Basic rpart model

```
set.seed(123)
rpart_mod <- train(survived_fct ~ .,
                   data=df_final,
                   method="rpart",
                   tuneLength = 5,
                   trControl = ctrl)

# visualize the tree
# plot(rpart_mod$finalModel)
# text(rpart_mod$finalModel)
```

Ctree

```
set.seed(123)
ctree_mod <- train(survived_fct ~ .,
                   data=df_final,
                   method="ctree",
                   trControl = ctrl)

# visualize the tree
# plot(ctree_mod$finalModel, type = "simple")
# print(ctree_mod$finalModel)
```

Random Forest

```
rf_grid <- tgrid <- expand.grid(
                      mtry = seq(5, 15, by=2),
                      splitrule = "gini",
                      min.node.size = c(15, 20, 25, 30))

set.seed(123)
rf_mod <- train(survived_fct ~ .,
                data=df_final,
                method="ranger",
                tuneGrid = rf_grid,
                trControl = ctrl)
```

XGB

```
grid_default <- expand.grid(
  nrounds = c(50, 100, 150),
  max_depth = c(1, 3, 6),
  eta = c(0.025, 0.05, 0.1, 0.3),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

set.seed(123)
xgb_mod <- train(survived_fct ~ .,
                data=df_final,
                method="xgbTree",
                trControl=ctrl,
                tuneGrid=grid_default)
```

Compare models

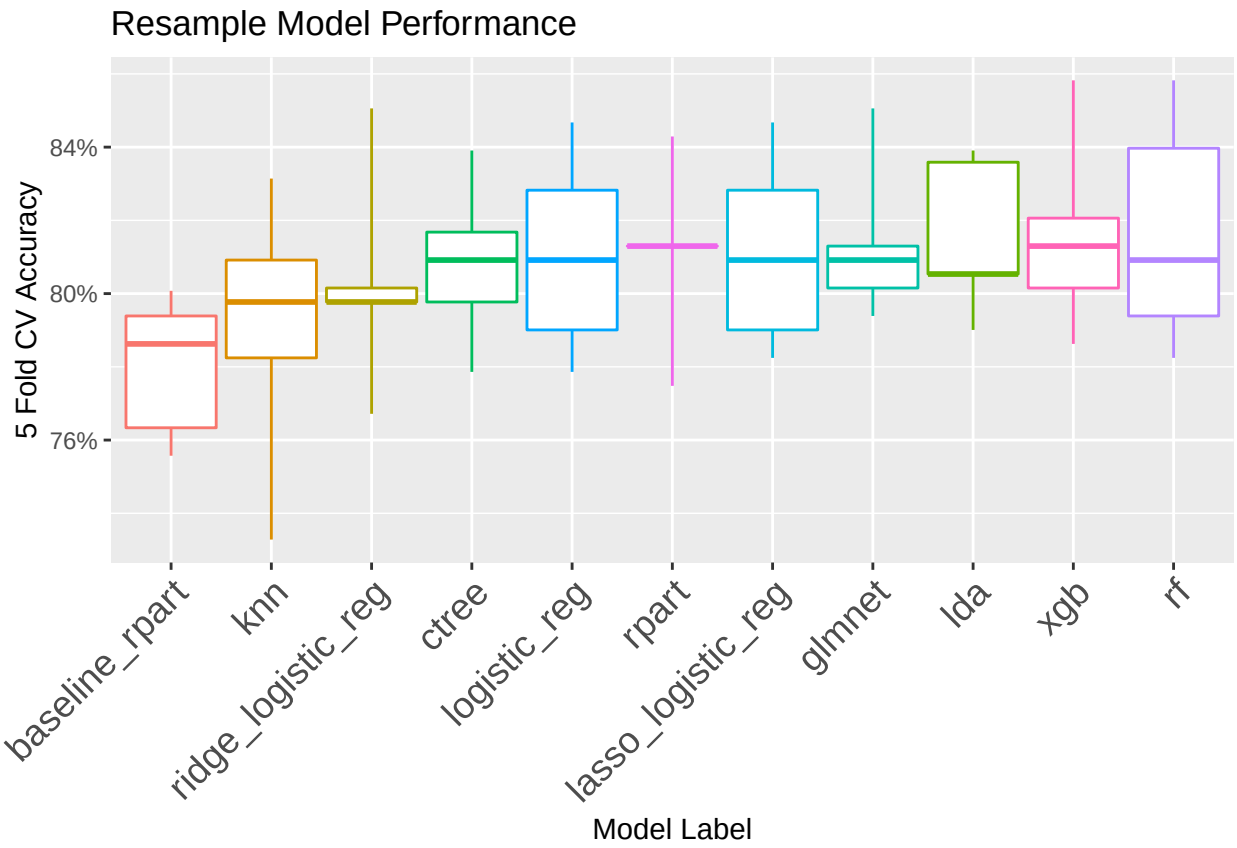
- Most of the models trend slightly above the baseline simple model.
- We'll select the Random Forest model as the final model for this analysis (RF has the highest average accuracy across the folds).
- Other models could be selected as there looks to be similar performance.
- KNN looks to have the worst performance on the resampling data.

```
# collect resample results (results for the 5 folds)
results <- resamples(list(baseline_rpart = baseline_mod,
                          logistic_reg=logistic_reg_mod,
                          lasso_logistic_reg = lasso_logistic_reg_mod,
```

```
                          ridge_logistic_reg = ridge_logistic_reg_mod,
                          glmnet = glmnet_mod,
                          lda = lda_mod,
                          knn = knn_mod,
                          rpart = rpart_mod,
                          ctree = ctree_mod,
                          rf = rf_mod,
                          xgb = xgb_mod))

# build df to plot resample perf
accuracy_df <- data.frame(summary(results)$statistics$Accuracy)
accuracy_df <- accuracy_df %>%
  as_tibble(rownames = NA) %>%
  rownames_to_column(var="model")

# resample model plot
accuracy_df %>%
  ggplot(aes(x=reorder(model, Mean), color=reorder(model, Median))) +
  geom_boxplot(
   aes(ymin = Min., lower = X1st.Qu., middle = Median, upper = X3rd.Qu., ymax = Max.),
   stat = "identity"
  ) +
  scale_y_continuous(labels = scales::percent_format(accuracy=1)) +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1, size=14)) +
  labs(title="Resample Model Performance",
       y="5 Fold CV Accuracy",
       x="Model Label")
```



Compare selected model vs baseline - Compare the results of the baseline model vs random forest. - RF has CV 5 fold accuracy results that are better than baseline and statistically significant. - This conclusion ties with the boxplots above (much of the RF distribution doesn't overlap with the baseline model).

```
compare_models(rf_mod, baseline_mod)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 4.1784, df = 4, p-value = 0.01394
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.01230903 0.06106115
## sample estimates:
##  mean of x
## 0.03668509
```

Train model on full training dataset

```
# update levels so ranger uses survived 1 as positive class
df_final <- df_final %>% mutate(survived_fct = factor(survived_fct, levels=c("1","0")))

final_rf_mod <- ranger(survived_fct ~ .,
                       data=df_final,
                       num.trees = rf_mod$finalModel$num.trees,
                       min.node.size = rf_mod$finalModel$min.node.size,
                       mtry = rf_mod$finalModel$mtry,
```

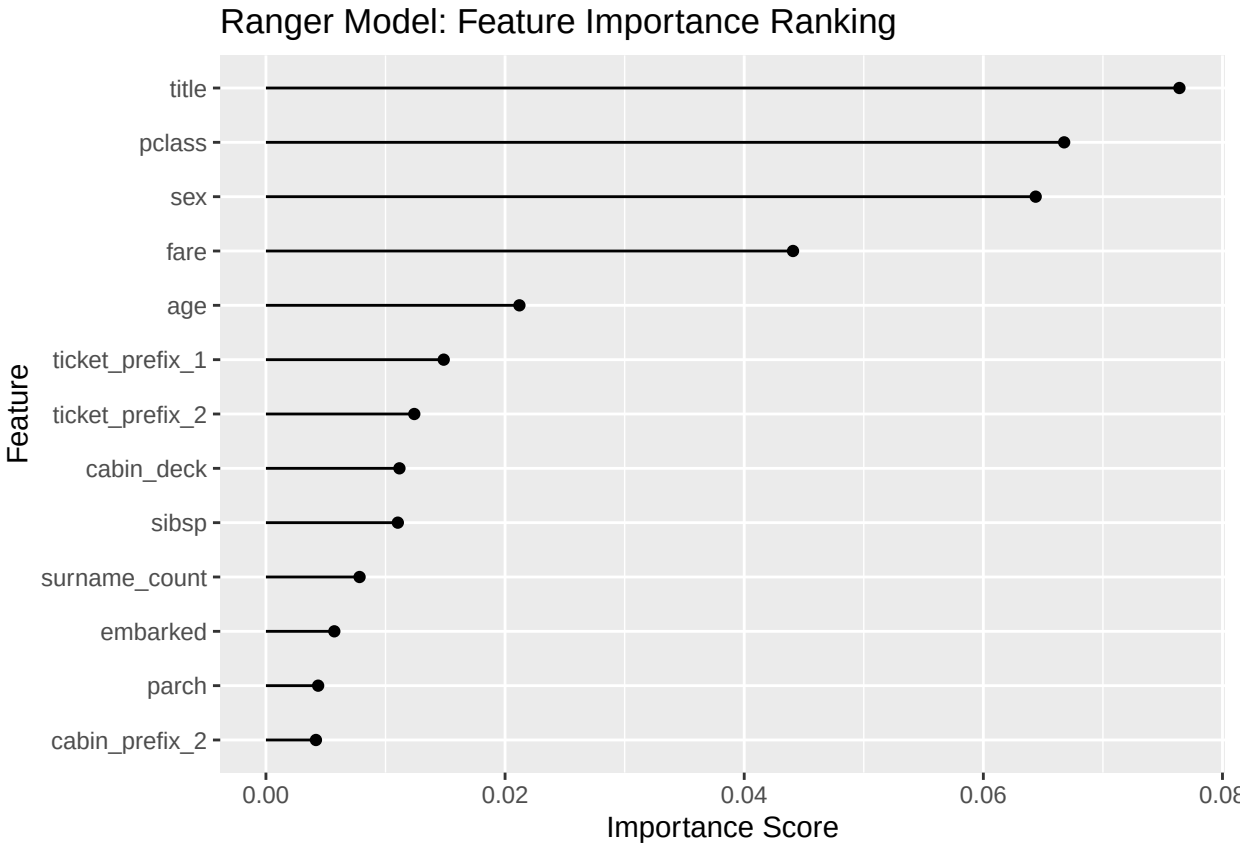
```
                        respect.unordered.factors = TRUE,
                        splitrule = "gini",
                        importance = "permutation",
                        probability = TRUE)
final_rf_mod
```

```
## Ranger result
##
## Call:
##  ranger(survived_fct ~ ., data = df_final, num.trees = rf_mod$finalModel$num.trees,      min.node.si
##
## Type:                             Probability estimation
## Number of trees:                  500
## Sample size:                      1309
## Number of independent variables:  13
## Mtry:                             11
## Target node size:                 15
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.1339837
```

Investigate feature importance

- Using the permutation method which shuffles a features values then checks error increases
- Features that result in higher error after being shuffled are features that score higher on the importance rank (e.g. these features are more importance to reducing model error)

```
data.frame(score = final_rf_mod$variable.importance) %>%
  rownames_to_column(var="feature") %>%
  mutate(feature = fct_reorder(factor(feature), score, max)) %>%
  ggplot(aes(x=score, y=feature)) +
  geom_segment(aes(y = feature,
                   x = 0,
                   yend = feature,
                   xend = score),
               color = "black") +
  geom_point() +
  labs(title="Ranger Model: Feature Importance Ranking",
       y="Feature",
       x="Importance Score")
```

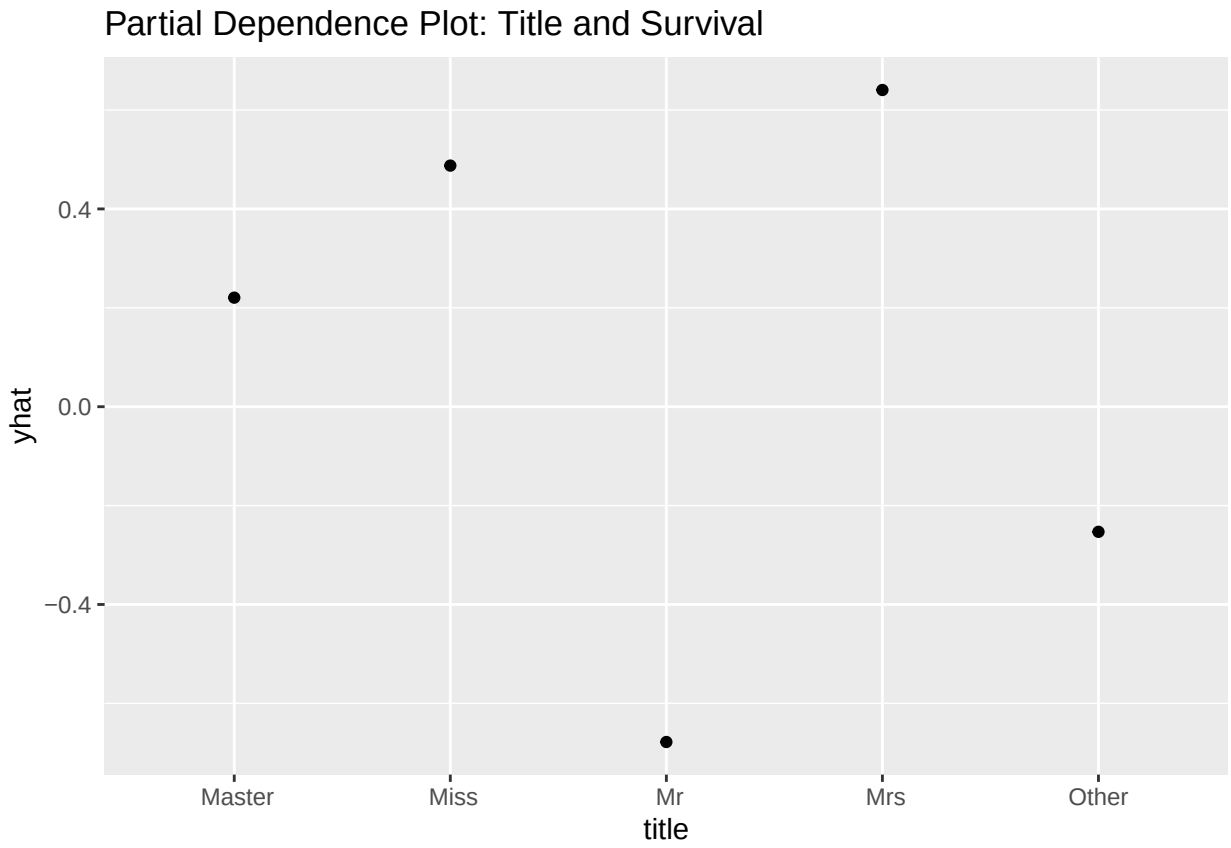


Partial Dependence Plots

- We can use partial dependence plots to gain intuition on the direction of the top features
- PDPs: visualize the marginal effect of a given feature on the outcome variable for different values of the feature
- Individual Conditional Expectation plots can also be used to see how predictions change at the observation level when a feature value changes.

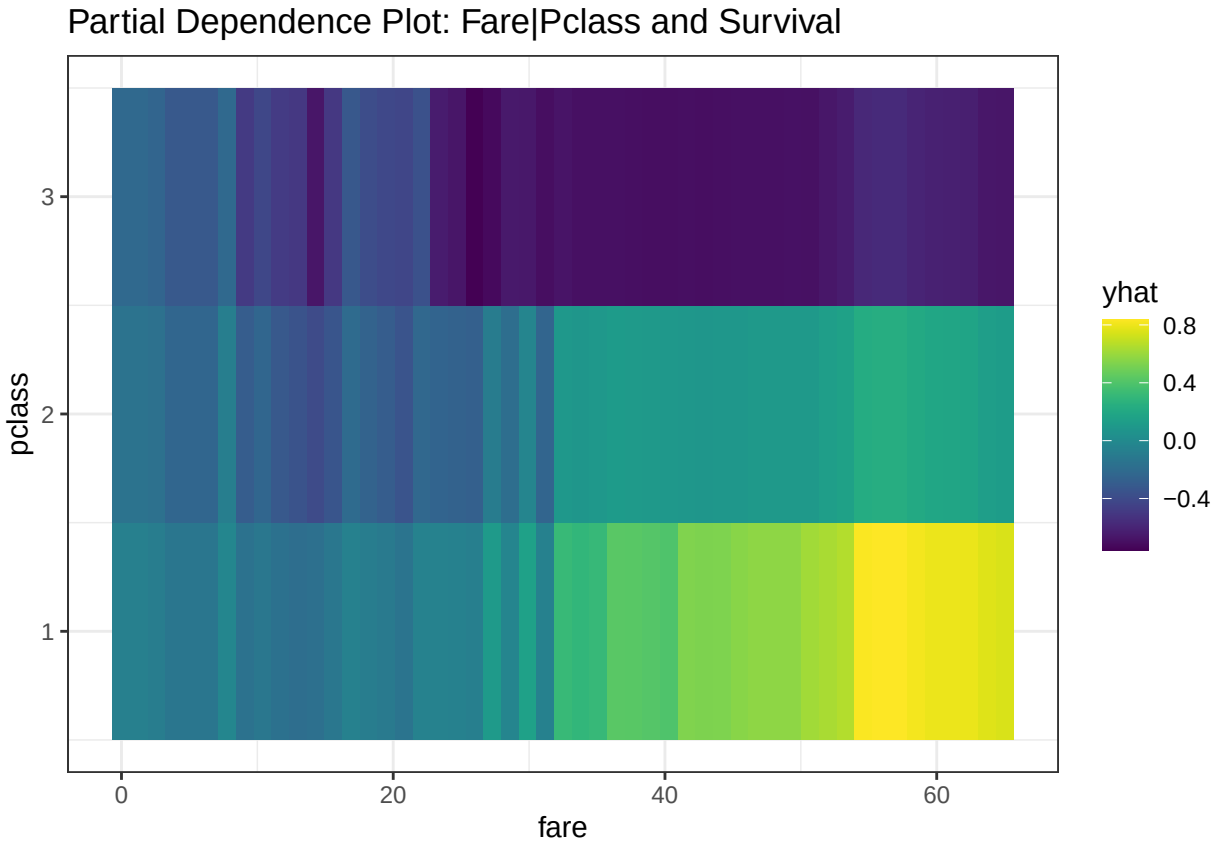
Overall, the final model predicts that survival is most likely for passengers
with Mrs title and survival is likely for Mr titles.

```
partial(final_rf_mod,
        pred.var = c("title"),
        trim.outliers = TRUE,
        rug = TRUE,
        plot=TRUE, plot.engine = "ggplot",
        paropts = list(.packages = "ranger")) +
  labs(title="Partial Dependence Plot: Title and Survival")
```



Overall, final model predicts that survival is more likely for older passengers in first or second class vs third class.

```
partial(final_rf_mod,
        pred.var = c("fare", "pclass"),
        trim.outliers = TRUE,
        rug = TRUE,
        plot=TRUE, plot.engine = "ggplot",
        paropts = list(.packages = "ranger")) +
  labs(title="Partial Dependence Plot: Fare|Pclass and Survival")
```



Conclusions

- For this analysis, exploratory analysis visualizations seemed more valuable than modeling. Given the relatively small feature set, visualizations can be used to investigate combinations of features and the relationship with survival.
- Robust models only performed slightly better than a simple prediction that all females survived.
- Based on the Titanic literature, it seems women and children were given priority to life boats first (this trend is present in the data). Additionally, there looks to have been a class bias.
- Potentially passengers who were in higher social class groups were given priority in line for life boats or closer to the life boats.

Areas for further investigation

- Additional feature engineering and model tuning
- Include individual conditional expectation plots in addition to PDP plots
- Investigate how to reduce multicollinearity for logistic regression approaches (then use for inference)
- More in depth write up could be included in the report
- We could run statistical tests to make stronger claims between the differences of passengers that survived vs perished