HARVARD EXTENSION SCHOOL

EXT CSCI E-106 Model Data Class Group Project Template

Author: Dinesh Bedathuru   Author: Brian Calderon

Author: Jeisson Hernandez   Author: Hao Fu   Author: Derek Rush

Author: Jeremy Tajonera   Author: Catherine Tully

08 May 2025

In this project, our aim is to classify the probability of a passenger surviving the Titanic crash of 1912. We used a variety of linear and non-linear models to deduce the most accurate model and provide long-term stability in our predictions.

Table of contents

List of Figures

List of Tables

Classify whether a passenger on board the maiden voyage of the RMS Titanic in 1912 survived given their age, sex and class. Sample-Data-Titanic-Survival.csv to be used in the Final Project

Variable	Description
pclass	**Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)**
survived	**Survival (0 = No; 1 = Yes)**
name	**Name**
sex	**Sex**
age	**Age**
sibsp	**# of siblings / spouses aboard the Titanic**
parch	**# of parents / children aboard the Titanic**
ticket	**Ticket number**
fare	**Passenger fare**
cabin	**Cabin number**
embarked	**Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)**
boat	**Lifeboat ID, if passenger survived**
body	**Body number (if passenger did not survive and body was recovered**
home.dest	**The intended home destination of the passenger**

1 Instructions:

0. Join a team with your fellow students with appropriate size (Up to Nine Students total) If you have not group by the end of the week of April 11 you may present the project by yourself or I will randomly assign other stranded student to your group. I will let know the final groups in April 11.

1. Load and Review the dataset named "Titanic_Survival_Data.csv" 2. Create the train data set which contains 70% of the data and use set.seed (15). The remaining 30% will be your test data set.

3. Investigate the data and combine the level of categorical variables if needed and drop variables as needed. For example, you can drop id, Latitude, Longitude, etc.

4. Build appropriate model to predict the probability of survival.

5. Create scatter plots and a correlation matrix for the train data set. Interpret the possible relationship between the response.

6. Build the best models by using the appropriate selection method. Compare the performance of the best logistic linear models.

7. Make sure that model assumption(s) are checked for the final model. Apply remedy measures (transformation, etc.) that helps satisfy the assumptions.

8. Investigate unequal variances and multicollinearity.

9. Build an alternative to your model based on one of the following approaches as applicable to predict the probability of survival: logistic regression, classification Tree, NN, or SVM. Check the applicable model assumptions. Explore using a negative binomial regression and a Poisson regression.

10. Use the test data set to assess the model performances from above.

11. Based on the performances on both train and test data sets, determine your primary (champion) model and the other model which would be your benchmark model.

12. Create a model development document that describes the model following this template, input the name of the authors, Harvard IDs, the name of the Group, all of your code and calculations, etc..

Due Date: May 12 2025 1159 pm hours EST Notes No typographical errors, grammar mistakes, or misspelled words, use English language All tables need to be numbered and describe their content in the body of the document All figures/graphs need to be numbered and describe their content All results must be accurate and clearly explained for a casual reviewer to fully understand their purpose and impact Submit both the RMD markdown file and PDF with the sections with appropriate explanations. A more formal.

2 Executive Summary

This section will describe the model usage, your conclusions and any regulatory and internal requirements. In a real world scenario, this section is for senior management who do not need to know the details. They need to know high level (the purpose of the model, limitations of the model and any issues).

3 Introduction

This section needs to introduce the reader to the problem to be resolved, the purpose, and the scope of the statistical testing applied. What you are doing with your prediction? What is the purpose of the model? What methods were trained on the data, how large is the test sample, and how did you build the model?

The Titanic was a British-registered ship that set sail on its maiden voyage on April 10th, 1912 with 2,240 passengers and crew on board. On April 15th, 1912, the ship struck an iceberg, split in half, and sank to the bottom of the ocean (National Oceanic and Atmospheric Administration (NOAA), 2023). In this report, we are going to analyze the data in the Titanic.csv file and use it to determine the best model for predicting whether someone on board would live or die. By creating this model, we hope to understand what factors a passenger could have taken into account in order to reduce their risk of death during the trip. We cleaned the data and split into into a train/test split in order to properly train our models. We created simple linear models, multivariate linear models, logistic models (both binomial and poisson), a regression tree, and a neural network model. The train sample size was 916 data points (70.03%) and the test sample size was 392 data points (29.97%). We built the

models after examining the data and determining which predictor variables we thought would be most relevant for survival rate. Once we had our variables and training data, we created the models and examined the performance of the models on both training and testing data to determine if they were robust. We also examined if the model assumptions appeared to hold for each model.

4 Description of the data and quality

Based on the data cleaning we were able to only remove 2 rows from the data set. We used median imputation as well as KNN for various columns. We also dummified several categorical columns. We found that leaving sibsp and parch as continuous as opposed to categorical increased their contributions to the model performance Appendix A. Further, we also extracted the deck number and found that removing deck_G from the model increased its performance.

4.1 Loading the data

```
odata <- read.csv("../data/Titanic_Survival_Data.csv")
cat("Size of entire data set:", nrow(odata), "\n")
```

```
Size of entire data set: 1310
```

4.2 Removing un-needed columns

Name: Removing because names have no inference on surivival (inference)

ticket: Ticket No. will also likely not have an influence in survival

boat: This is highly correlated to the survival dependant variable since people who made it on a boat likely survived

body: This is highly correlated to the survival dependant variable since people who's body was recovered did not survive.

home.dest: The destination likely has nothing to do with the survival

```
data.clean = odata[, !(names(odata) %in% c("name", "ticket", "boat","body","home.dest"))]
```

4.3 Data Augmentation

We extracted the deck letter from the cabin since it could potentially correlate to the survival.

```
#Extract deck letter from cabin
data.clean$deck <- substr(data.clean$cabin, 1,1)
# Remove cabin col:
data.clean$cabin <- NULL
```

4.4 Initial Check for Missing values

We see that age and deck have the most amount of missing data, therefore we proceed to impute them.

```
print(plot_missing_barchart(data.clean))
```
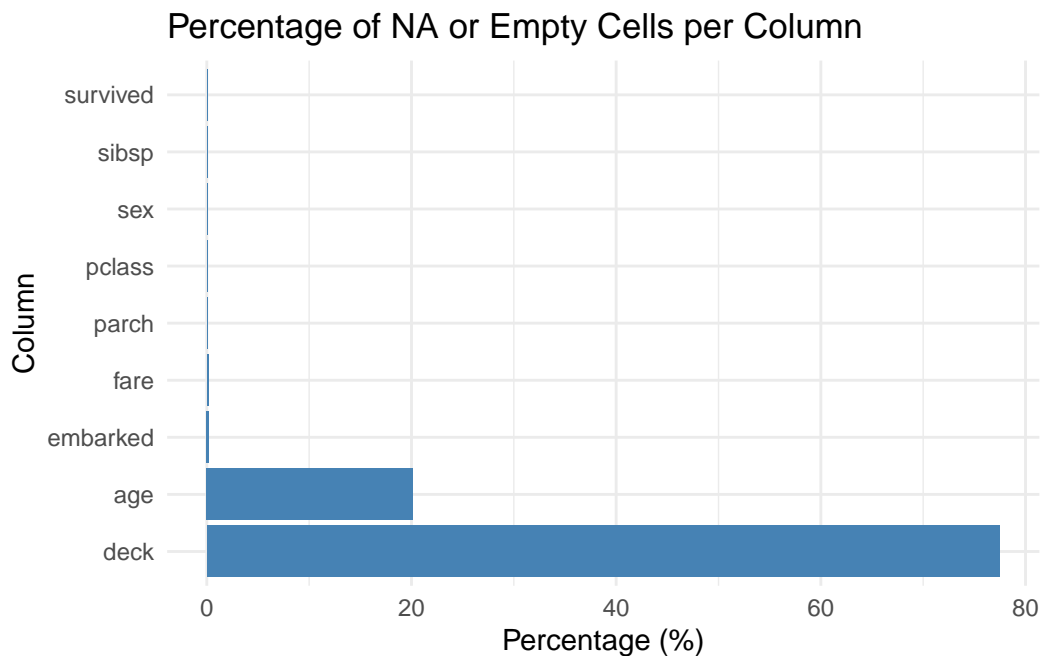

Figure 1: Percentage of Missing Values

4.5 Imputing data

Below we impute Age using the median value in that column.

For deck we use KNN to impute the missing deck values.

After imputing these two columns we can see that the largest amount of missing data is ~0.2% which is quite small and can be removed.

```
# ---- Age----
#Replace NAs in age column with Median value
median_age <- median(data.clean$age, na.rm = TRUE)
data.clean <- data.clean %>%
  mutate(age = ifelse(is.na(age), median_age, age))

# ---- deck----
# For deck, since its a category, we decided to use KNN  to impute the column:

# Install if not already installed
# install.packages("VIM")
library(VIM)
```

```
Warning: package 'VIM' was built under R version 4.4.3
```

Loading required package: colorspace

Loading required package: grid

VIM is ready to use.

Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

Attaching package: 'VIM'

The following object is masked from 'package:datasets':

    sleep

```r
# Replace "" with NA in the 'deck' column
data.clean$deck[data.clean$deck == ""] <- NA

# Convert 'cabin' to factor
data.clean$deck <- as.factor(data.clean$deck)

# Apply kNN imputation just to Cabin column
data.clean <- kNN(data.clean, variable = "deck", k = 5)

# Check that NAs were imputed
# sum(is.na(data.clean$deck))         # Original
# sum(is.na(data.clean.imputed$deck)) # After

# Remove indicator col:
data.clean$deck_imp <- NULL
```

```r
################################################################################
#          Check for Missing values after Imputation                        #
################################################################################

plot_missing_barchart(data.clean)
```
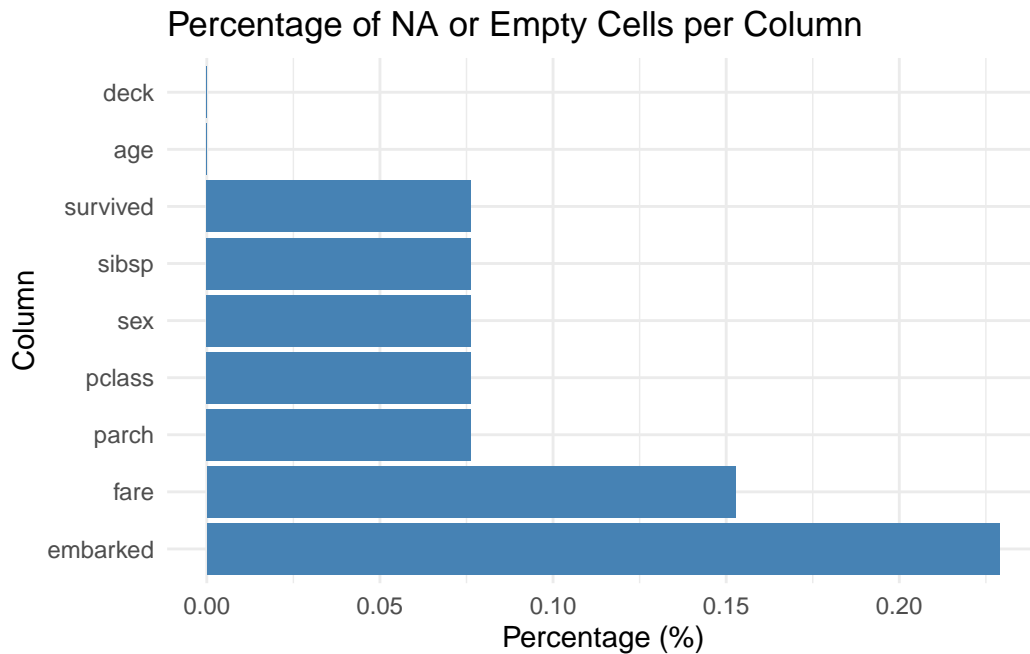
Figure 2: Percentage of Missing Values after Imputation

4.6 Dummifying Columns:

We dummify pclass, sex, embarked and deck. We leave sibsp and parch as continuous variables as we observed that dummifying these columns leads to smaller significance Appendix A, whilst leaving them as continuous maximizes their contributions to the models explanatory power.

```
# Dummifying pclass:
data.clean$pclass_1 = ifelse(data.clean$pclass == 1, 1, 0)
data.clean$pclass_2 = ifelse(data.clean$pclass == 2, 1, 0)

# Dummifying sex:
data.clean$sex_M = ifelse(data.clean$sex == 'male', 1, 0)

# Dummifying embarked:
data.clean$embarked_C = ifelse(data.clean$embarked == 'C', 1, 0)
data.clean$embarked_Q = ifelse(data.clean$embarked == 'Q', 1, 0)

# Dummifying deck:
data.clean$deck_A = ifelse(data.clean$deck == 'A', 1, 0)
data.clean$deck_B = ifelse(data.clean$deck == 'B', 1, 0)
data.clean$deck_C = ifelse(data.clean$deck == 'C', 1, 0)
data.clean$deck_D = ifelse(data.clean$deck == 'D', 1, 0)
data.clean$deck_E = ifelse(data.clean$deck == 'E', 1, 0)
data.clean$deck_F = ifelse(data.clean$deck == 'F', 1, 0)
data.clean$deck_G = ifelse(data.clean$deck == 'G', 1, 0)

# Removing Dummified cols:
data.clean = subset(data.clean, select  = -c(pclass, sex, embarked,deck))
```

4.7 Remove NA rows and deck_G

Below we remove NA rows, which turned out to be only 2 after proper cleaning and imputation. We also removed deck_G as we observed that it has a large skew in the data distribution with only 13 people allocated in this deck. It was observed that this variable lead to erroneous predictions in the model.

```
# Plot histogram of the 'values' column
hist(data.clean$deck_G,
     main = "Histogram of Values",
     xlab = "Values",
     col = "skyblue",
     border = "white")
```
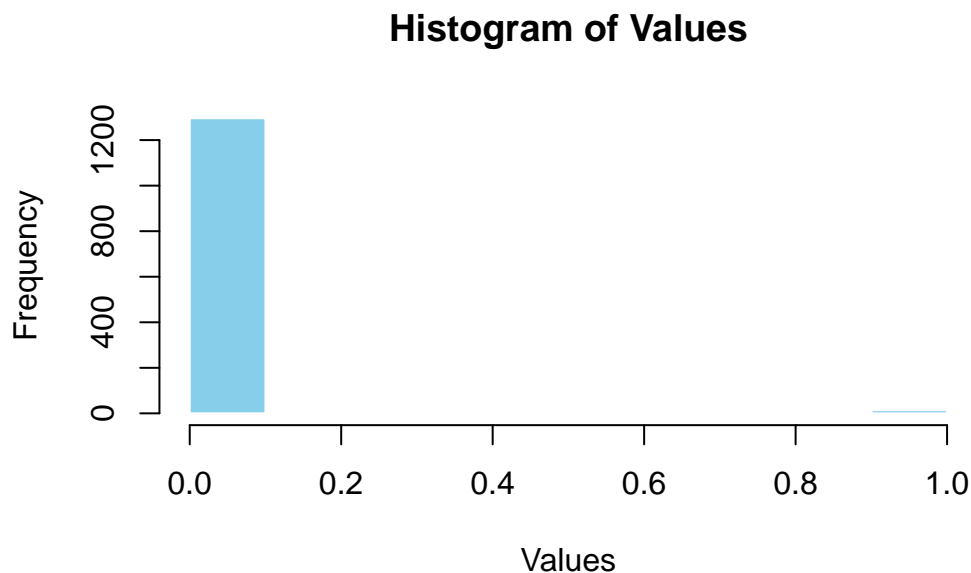

Figure 3: Histogram of Deck_G

```
# Removing deck_G col:
data.clean = subset(data.clean, select   = -c(deck_G))
```

```
data.clean = na.omit(data.clean)
cat(nrow(odata) - nrow(data.clean),'rows were removed from original dataset')
```

2 rows were removed from original dataset

4.8 Divide into Test / Train

Finally we divide into 70% training data and 30% test data.

```
set.seed (1023)
train_indices = sample(1 : nrow(data.clean), size = 0.7005*nrow(data.clean), replace = FALSE)
train = data.clean[train_indices,]
test = data.clean[-train_indices,]
cat("We are using:", nrow(train)/nrow(data.clean) * 100, '% of the data for training')
```

We are using: 70.03058 % of the data for training

4.9 EDA

Using the training data set we use a variety of method to draw some initial conclusions:

- Histogram: Showing that more people in their late teens up to late thirties survived.
- Bar chart showing that more people died than survived
- Bar chart showing that a higher number of people survived when they had less siblings on board.
- Correlation matrix shows that sex and Deck_F are highly negatively correlated to survival. There is a soft positive correlation to pclass_1.
- There is a high correlation between pclass_1 and fare, this justifies that one of these predictors can potentially be removed.
- The scatter plots did not give us much more information on the relation between the predictors and the dependent variable.

```
# Histogram showing that more people in their late teens up to late thirties survived.
ggplot(train, aes(age)) +
  geom_histogram(bins=30)
```
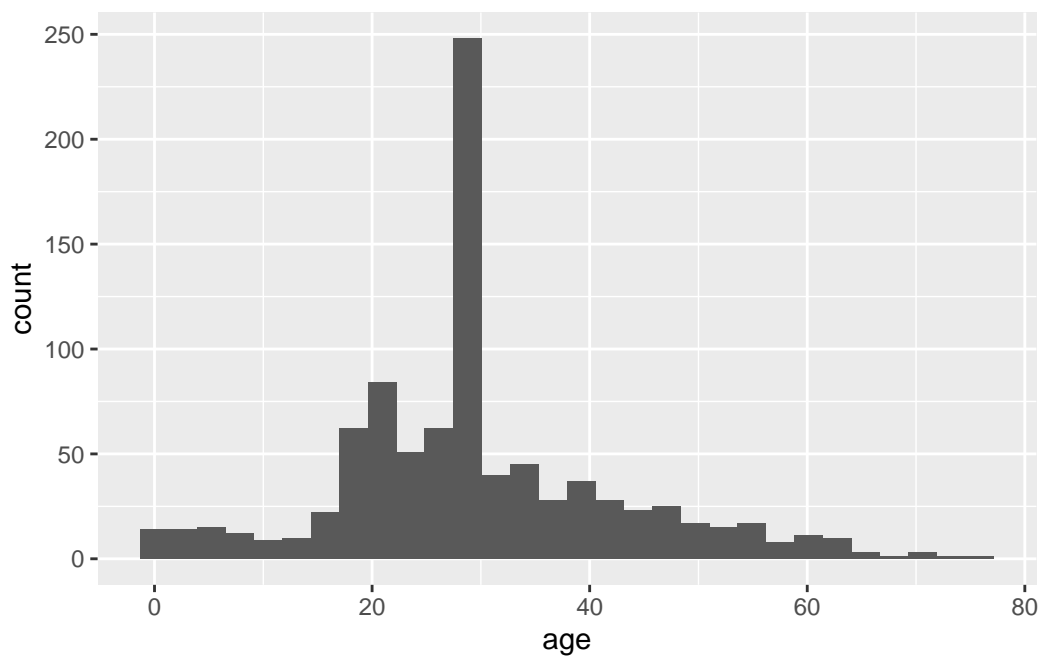

Figure 4: Histogram of survival vs age

```
# Bar chart showing that more people died than survived
ggplot(train, aes(survived)) +
  geom_bar()
```
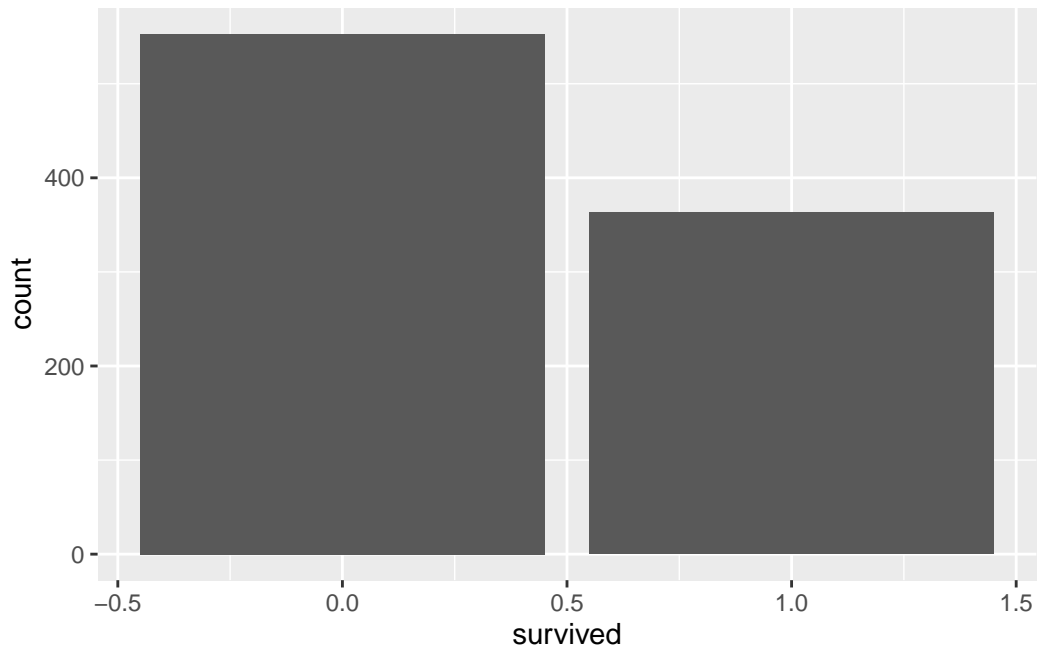
Figure 5: Barchart of survival

```
# Bar chart showing that a higher number of people survived when they had less
# siblings on board.
ggplot(train, aes(sibsp, survived)) +
  geom_bar(stat='identity')
```
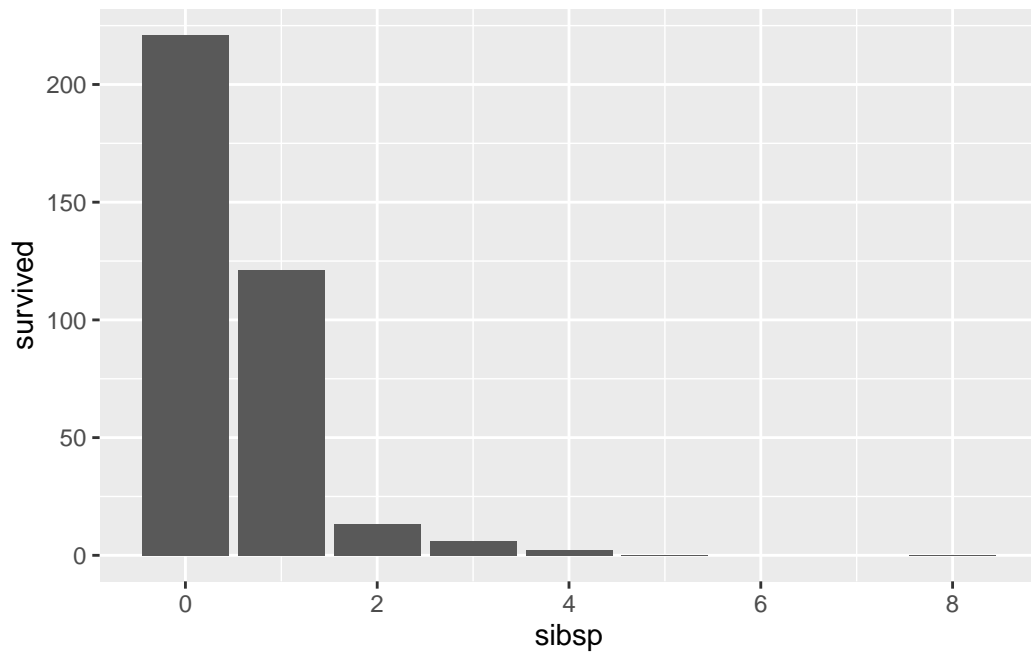

Figure 6: Barchart of survival vs Num. of siblings

```
cor(train)
```

	survived	age	sibsp	parch	fare

```
survived      1.00000000 -0.04599216 -0.027543645  0.095720424  0.249620417
age          -0.04599216  1.00000000 -0.154952394 -0.122885529  0.164477215
sibsp        -0.02754365 -0.15495239  1.000000000  0.355216328  0.165296185
parch         0.09572042 -0.12288553  0.355216328  1.000000000  0.209878267
fare          0.24962042  0.16447722  0.165296185  0.209878267  1.000000000
pclass_1      0.28432063  0.34847627 -0.022306615 -0.016451014  0.591163043
pclass_2      0.06118909  0.01898825 -0.069089243 -0.017696720 -0.125637353
sex_M        -0.53377358  0.05385130 -0.127066747 -0.243501842 -0.201776401
embarked_C    0.14757622  0.05701305 -0.061215242 -0.001165488  0.272087792
embarked_Q   -0.02542950 -0.03373092 -0.060221066 -0.093253422 -0.126992379
deck_A        0.02818626  0.12375210 -0.063208812 -0.046447535  0.060156186
deck_B        0.17102793  0.11384754 -0.018056185  0.073894445  0.458426488
deck_C        0.15622066  0.15548653  0.032030626 -0.054359774  0.314379308
deck_D        0.11169173  0.07390772 -0.003562241  0.002702881  0.007239167
deck_E        0.32472414  0.11407881 -0.082487921 -0.030305432 -0.017794021
deck_F       -0.47931430 -0.28313314  0.052474332  0.010856728 -0.420674788
                pclass_1    pclass_2        sex_M   embarked_C   embarked_Q
survived      0.28432063  0.06118909 -0.53377358  0.147576220 -0.02542950
age           0.34847627  0.01898825  0.05385130  0.057013047 -0.03373092
sibsp        -0.02230662 -0.06908924 -0.12706675 -0.061215242 -0.06022107
parch        -0.01645101 -0.01769672 -0.24350184 -0.001165488 -0.09325342
fare          0.59116304 -0.12563735 -0.20177640  0.272087792 -0.12699238
pclass_1      1.00000000 -0.30255006 -0.11706927  0.285611052 -0.16373859
pclass_2     -0.30255006  1.00000000 -0.01057862 -0.143425262 -0.14105035
sex_M        -0.11706927 -0.01057862  1.00000000 -0.040151739 -0.09800681
embarked_C    0.28561105 -0.14342526 -0.04015174  1.000000000 -0.16362864
embarked_Q   -0.16373859 -0.14105035 -0.09800681 -0.163628636  1.00000000
deck_A        0.27817486 -0.07214490  0.05726877  0.179563214 -0.06045425
deck_B        0.42157158 -0.12290262 -0.10513911  0.197253957 -0.08030780
deck_C        0.49783147 -0.09377853 -0.08157273  0.166128834 -0.07869333
deck_D        0.11604898  0.04117623 -0.06599395  0.288111512 -0.09280535
deck_E        0.06047826 -0.03335428 -0.07748047 -0.085668330 -0.08697346
deck_F       -0.70753043  0.15770708  0.15681273 -0.353103247  0.22529873
                  deck_A       deck_B       deck_C       deck_D       deck_E
survived      0.02818626  0.17102793  0.15622066  0.111691728  0.32472414
age           0.12375210  0.11384754  0.15548653  0.073907716  0.11407881
sibsp        -0.06320881 -0.01805619  0.03203063 -0.003562241 -0.08248792
parch        -0.04644753  0.07389444 -0.05435977  0.002702881 -0.03030543
fare          0.06015619  0.45842649  0.31437931  0.007239167 -0.01779402
pclass_1      0.27817486  0.42157158  0.49783147  0.116048985  0.06047826
pclass_2     -0.07214490 -0.12290262 -0.09377853  0.041176226 -0.03335428
sex_M         0.05726877 -0.10513911 -0.08157273 -0.065993951 -0.07748047
embarked_C    0.17956321  0.19725396  0.16612883  0.288111512 -0.08566833
embarked_Q   -0.06045425 -0.08030780 -0.07869333 -0.092805348 -0.08697346
deck_A        1.00000000 -0.04808710 -0.06241531 -0.055570446 -0.06660453
deck_B       -0.04808710  1.00000000 -0.08291288 -0.073820128 -0.08847788
deck_C       -0.06241531 -0.08291288  1.00000000 -0.095815832 -0.11484105
deck_D       -0.05557045 -0.07382013 -0.09581583  1.000000000 -0.10224685
deck_E       -0.06660453 -0.08847788 -0.11484105 -0.102246852  1.00000000
deck_F       -0.23590750 -0.31338100 -0.40675710 -0.362149521 -0.43405805
                  deck_F
survived     -0.47931430
age          -0.28313314
```

```
sibsp        0.05247433
parch        0.01085673
fare        -0.42067479
pclass_1    -0.70753043
pclass_2     0.15770708
sex_M        0.15681273
embarked_C  -0.35310325
embarked_Q   0.22529873
deck_A      -0.23590750
deck_B      -0.31338100
deck_C      -0.40675710
deck_D      -0.36214952
deck_E      -0.43405805
deck_F       1.00000000
```

```
pairs(train[c(1:4,7,13)])
```
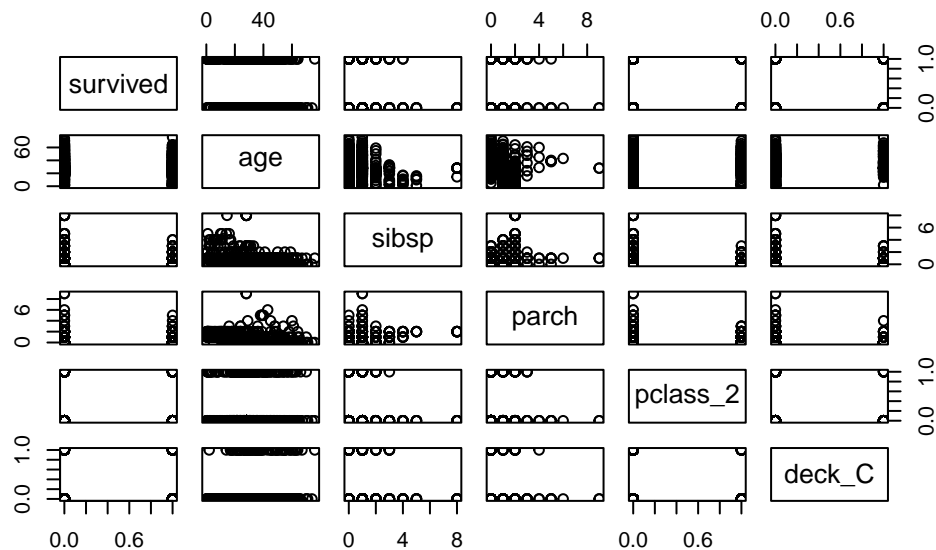

Figure 7: Scatter plots of all variables in train data

```
# Since this data is mainly categorical, the scatterplot and correlation matrix are not very useful.
```

(Statology, 2025) is used to develop the correlation values between our categorical columns. This describes the use of pysch and rcompanion.

```
#install.packages("psych")
library(psych) # [@statology2025] to understand how this works
```

```
Warning: package 'psych' was built under R version 4.4.3
```

```
Attaching package: 'psych'
```

The following objects are masked from 'package:ggplot2':

    %+%, alpha

```r
tetrachoric(train[, c("survived", "sex_M")])
```

```
Call: tetrachoric(x = train[, c("survived", "sex_M")])
tetrachoric correlation
        srvvd sex_M
survived  1.00
sex_M    -0.75  1.00

 with tau of
survived    sex_M
    0.26    -0.35
```

```r
tetrachoric(train[, c("survived", "pclass_1")])
```

```
Call: tetrachoric(x = train[, c("survived", "pclass_1")])
tetrachoric correlation
        srvvd pcl_1
survived 1.00
pclass_1 0.46  1.00

 with tau of
survived pclass_1
    0.26     0.69
```

```r
tetrachoric(train[, c("survived", "pclass_2")])
```

```
Call: tetrachoric(x = train[, c("survived", "pclass_2")])
tetrachoric correlation
        srvvd pcl_2
survived 1.00
pclass_2 0.11  1.00

 with tau of
survived pclass_2
    0.26     0.77
```

```r
#install.packages("rcompanion")
library(rcompanion) # Reference 4 to understand how this works.
```

```
Warning: package 'rcompanion' was built under R version 4.4.3
```

```
Attaching package: 'rcompanion'
```

The following object is masked from 'package:psych':

    phi

```
cramerV(train$survived, train$sex)
```

Cramer V
  0.5338

```
library(corrplot)
```

corrplot 0.95 loaded

```
cor_matrix <- cor(train)#[,1]
corrplot(cor_matrix, method = "circle")
```
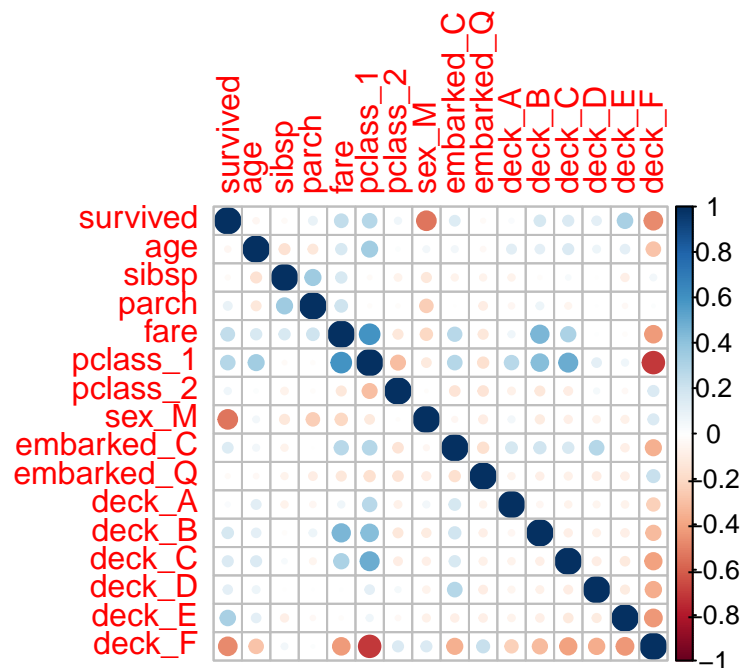

Figure 8: Correlation Matrix

5 Model Development Process

The data was properly cleaned and divided into train/test in the prior section.

Here we train a binary model. The Q-Q plot shows that the residuals are indeed normally distributed so a transformation is potentially not necessary.

The statistical comparison between test and train data shows that the model is very stable with an accuracy of ~84% for both.

We also analyzed the VIF and we see that there is high degree of correlation between the decks, this provides justification to remove some of the decks as predictors.

```r
library(car)
```

Warning: package 'car' was built under R version 4.4.3

Loading required package: carData

Warning: package 'carData' was built under R version 4.4.3

Attaching package: 'car'

The following object is masked from 'package:psych':

    logit

The following object is masked from 'package:dplyr':

    recode

The following object is masked from 'package:purrr':

    some

```r
# Log model on train data:
lmod <- glm(survived ~ ., family = binomial, data = train)
# summary(lmod)
vif(lmod)
```

```
        age        sibsp        parch         fare     pclass_1     pclass_2       sex_M
   1.493399     1.258340     1.281859     1.727034     4.267734     1.560100     1.604117
 embarked_C  embarked_Q       deck_A       deck_B       deck_C       deck_D       deck_E
   1.513511     1.397623     4.532575     4.848833     7.604802     6.662169     6.091048
     deck_F
  14.919819
```

```r
y_hat_log_train <- predict(lmod, data = train, type="response")
predictions_log_train <- ifelse(y_hat_log_train > 0.5, 1, 0)

y_hat_log_test<-predict(lmod, newdata = test, type="response")
predictions_log_test <- ifelse(y_hat_log_test > 0.5, 1, 0)

confusion_matrix_log_train <- confusionMatrix(as.factor(predictions_log_train), as.factor(train$survi
base.model.accuracy = confusion_matrix_log_train$overall['Accuracy']
base.model.f1 = confusion_matrix_log_train$byClass['F1']
base.model.train.summary = data.frame(
  Accuracy = base.model.accuracy,
  F1 = base.model.f1
)
row.names(base.model.train.summary) <- 'base.model.train'
```

```r
confusion_matrix_log_test <- confusionMatrix(as.factor(predictions_log_test), as.factor(test$survived
base.model.accuracy = confusion_matrix_log_test$overall['Accuracy']
base.model.f1 = confusion_matrix_log_test$byClass['F1']
base.model.test.summary = data.frame(
  Accuracy = base.model.accuracy,
  F1 = base.model.f1
)
row.names(base.model.test.summary) <- 'base.model.test'
```

```r
data.frame(rbind(base.model.train.summary, base.model.test.summary))
```

Table 2: Summary Stats. of base log. model

	Accuracy	F1
base.model.train	0.8471616	0.8071625
base.model.test	0.8367347	0.7730496

```r
summary(lmod)
```

```
Call:
glm(formula = survived ~ ., family = binomial, data = train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.3284111  0.8287318   6.430 1.28e-10 ***
age         -0.0557904  0.0090406  -6.171 6.78e-10 ***
sibsp       -0.2761926  0.1235912  -2.235  0.02544 *
parch        0.0250325  0.1186751   0.211  0.83294
fare         0.0001907  0.0023082   0.083  0.93415
pclass_1     0.8206018  0.4454290   1.842  0.06544 .
pclass_2     1.8580626  0.2991637   6.211 5.27e-10 ***
sex_M       -3.2522858  0.2555063 -12.729  < 2e-16 ***
embarked_C   0.6313152  0.2939442   2.148  0.03173 *
embarked_Q   0.8927717  0.3642358   2.451  0.01424 *
deck_A      -1.9122914  0.9044640  -2.114  0.03449 *
deck_B      -1.2596191  0.9199589  -1.369  0.17093
deck_C      -1.6453843  0.8501358  -1.935  0.05294 .
deck_D      -2.3295283  0.8315684  -2.801  0.00509 **
deck_E       0.4168948  0.8021132   0.520  0.60324
deck_F      -4.3018644  0.7733641  -5.563 2.66e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1230.15  on 915  degrees of freedom
```

```
Residual deviance:  635.57  on 900  degrees of freedom
AIC: 667.57

Number of Fisher Scoring iterations: 6
```

```r
par(mfrow=c(2,2))
plot(lmod)
```
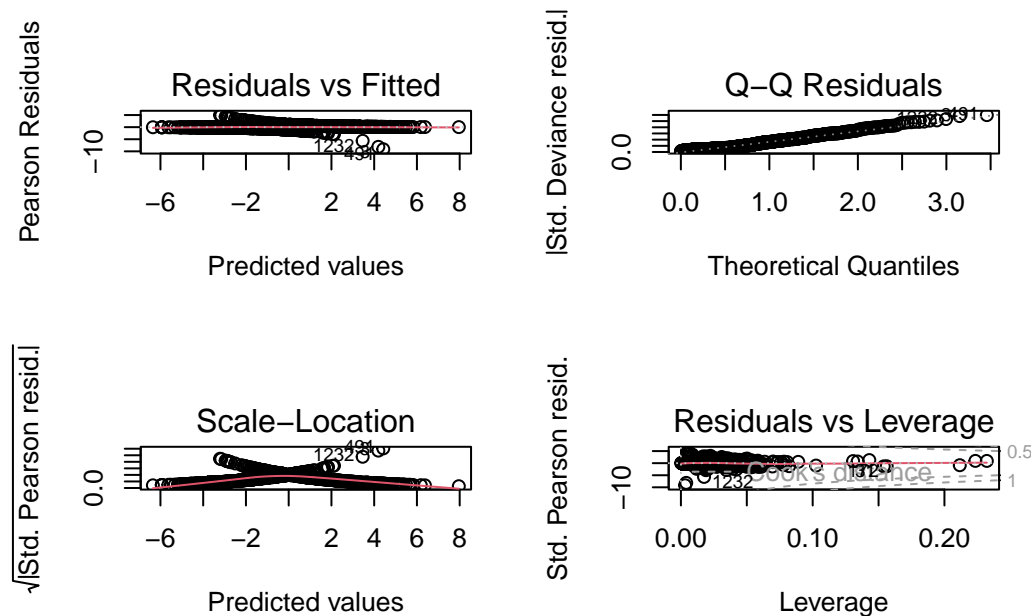

Figure 9: 4x4 standard plots for log. model

```
# Plots show that a linear model is not appropriate for this data.
```

6 Model Performance Testing

We compare four different models: 1) Base log. model 2) Model with insig. pred. removed 3) Stepwise model 4) Model with high vif pred. removed

When comparing the accuracy and F1 score on all models the base model was still the highest performer and we decided to use that as the champion model until now.

```r
##############################################################################
#            Function to remove Insig. Predictors one by one                 #
##############################################################################

backward_eliminate = function(model, alpha = 0.05) {
  repeat {
    d1 = drop1(model, test = "F")

    # Get p-values excluding intercept row
    pvals = d1$`Pr(>F)`[-1]
```

```r
    # Stop if all predictors are significant or only intercept left
    if( all(is.na(pvals)) || max(pvals, na.rm = TRUE) <= alpha ){
      print("all variable are signifcant")
      break
    }

    # Remove the term with max p-value
    term_to_remove = rownames(d1)[-1][which.max(pvals)]
    cat("Removing:", term_to_remove, "with p-value", max(pvals, na.rm = TRUE), "\n")
    model = update(model, paste(". ~ . -", term_to_remove))
  }
  return(model)
}


################################################################################
#                    Function to remove Cooks Outliers                        #
################################################################################
# model_formula: A formula object, e.g. PSA.level ~ .
# data: A data frame containing the variables in the model.
# threshold: A numeric value indicating the Cook's D threshold (default 0.5).
# print: If TRUE (default) will print the rows beign removed.
# returns: A list with the final model and the filtered dataset.
#
# Example usage:
# result = remove_cooks_outliers(PSA.level ~ ., mydata)
# summary(result$model)
# str(result$filtered_data)
remove_cooks_outliers = function(model_formula, data, threshold = 0.5,
                                 print = TRUE)
  {

  all_high_cd_rows = data.frame()  # to store all removed rows

  repeat {
    model = glm(model_formula, family = binomial, data = data)
    cooksD = cooks.distance(model)
    high_cd_indices = which(cooksD > threshold)

    if (length(high_cd_indices) == 0) { # If there are no more outliers
      break
    }

    if (print == TRUE){
      cat("Removing rows with Cook's D >", threshold, ":\n", high_cd_indices, "\n")
    }
    # Save these outliers before removing them
    high_cd_rows = data[high_cd_indices, ]
    all_high_cd_rows = rbind(all_high_cd_rows, high_cd_rows)

    # Update data by removing high CD rows for next iteration.
    data = data[-high_cd_indices, ]
  }
```

```r
  final_model = glm(model_formula, family = binomial, data = data)
  return(list(model = final_model, filtered_data = data, high_cd_data = all_high_cd_rows))
}
```

```r
##############################################################################
#                    Model with insig. variables removed                    #
##############################################################################
binary.model.filtered = backward_eliminate(lmod)
```

Warning in drop1.glm(model, test = "F"): F test assumes 'quasibinomial' family

Removing: fare with p-value 0.9215538

Warning in drop1.glm(model, test = "F"): F test assumes 'quasibinomial' family

Removing: parch with p-value 0.7803802

Warning in drop1.glm(model, test = "F"): F test assumes 'quasibinomial' family

Removing: deck_E with p-value 0.5506678

Warning in drop1.glm(model, test = "F"): F test assumes 'quasibinomial' family

[1] "all variable are signifcant"

```r
##############################################################################
#               step wise model with insig. variables removed               #
##############################################################################
library(olsrr)
```

Warning: package 'olsrr' was built under R version 4.4.3

Attaching package: 'olsrr'

The following object is masked from 'package:MASS':

    cement

The following object is masked from 'package:datasets':

    rivers

```r
ols_step_both_p(lmod,p_enter=0.1,p_remove=0.05,details=FALSE)
```

```
                         Stepwise Summary
-----------------------------------------------------------------------------
Step    Variable          AIC          SBC          SBIC          R2       Adj. R2
-----------------------------------------------------------------------------
0       Base Model     1293.366     1303.006    -28281799.310   0.00000   0.00000
1       sex_M (+)       988.183     1002.643    -55309468.385   0.28491   0.28413
2       deck_F (+)      757.429      776.709    -91941771.510   0.44537   0.44415
3       age (+)         725.480      749.580    -99004330.024   0.46555   0.46379
4       pclass_2 (+)    696.998      725.918   -105806102.781   0.48304   0.48077
5       deck_E (+)      673.931      707.671   -111746149.031   0.49700   0.49423
6       sibsp (+)       667.234      705.794   -113874510.309   0.50175   0.49846
7       deck_D (+)      663.423      706.804   -115313992.417   0.50490   0.50108
8       embarked_Q (+)  661.553      709.753   -116277909.273   0.50699   0.50264
9       embarked_Q (-)  663.423      706.804   -115313992.417   0.50490   0.50108
-----------------------------------------------------------------------------


Final Model Output
------------------


                     Model Summary
-----------------------------------------------------------------
R                     0.711      RMSE              0.344
R-Squared             0.505      MSE               0.118
Adj. R-Squared        0.501      Coef. Var        87.229
Pred R-Squared        0.495      AIC             663.423
MAE                   0.261      SBC             706.804
-----------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error
 AIC: Akaike Information Criteria
 SBC: Schwarz Bayesian Criteria


                         ANOVA
-----------------------------------------------------------------------
             Sum of
             Squares       DF    Mean Square      F          Sig.
-----------------------------------------------------------------------
Regression   110.647        7        15.807    132.282     0.0000
Residual     108.500      908         0.119
Total        219.147      915
-----------------------------------------------------------------------


                      Parameter Estimates
----------------------------------------------------------------------------------
       model      Beta    Std. Error    Std. Beta      t        Sig      lower     upper
----------------------------------------------------------------------------------
(Intercept)      1.128      0.044                     25.918    0.000    1.043     1.213
     sex_M      -0.465      0.024        -0.458      -19.077    0.000   -0.513    -0.418
    deck_F      -0.450      0.031        -0.450      -14.496    0.000   -0.511    -0.389
       age      -0.006      0.001        -0.170       -6.842    0.000   -0.008    -0.005
   pclass_2      0.156      0.028         0.132        5.491    0.000    0.100     0.211
```

```
   deck_E      0.165           0.043          0.105       3.842     0.000      0.081      0.249
    sibsp     -0.032           0.011         -0.071      -2.957     0.003     -0.054     -0.011
   deck_D     -0.116           0.048         -0.064      -2.404     0.016     -0.211     -0.021
-------------------------------------------------------------------------------------
```

```
Call:
glm(formula = survived ~ sex_M + deck_F + age + deck_E + pclass_2 +
    pclass_1 + sibsp + fare, family = binomial, data = train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.027276   0.443292   9.085  < 2e-16 ***
sex_M       -3.228025   0.242958 -13.286  < 2e-16 ***
deck_F      -2.649341   0.335751  -7.891 3.00e-15 ***
age         -0.054754   0.008774  -6.241 4.36e-10 ***
deck_E       1.823622   0.376762   4.840 1.30e-06 ***
pclass_2     1.472120   0.270394   5.444 5.20e-08 ***
pclass_1     0.531123   0.376285   1.411  0.15810
sibsp       -0.324078   0.115699  -2.801  0.00509 **
fare         0.001937   0.002089   0.927  0.35379
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1230.15  on 915  degrees of freedom
Residual deviance:  652.91  on 907  degrees of freedom
AIC: 670.91

Number of Fisher Scoring iterations: 6
```

```
Call:
glm(formula = survived ~ sex_M + deck_F + age + deck_E + pclass_2 +
    pclass_1 + sibsp, family = binomial, data = train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.075133   0.441121   9.238  < 2e-16 ***
sex_M       -3.250356   0.241996 -13.431  < 2e-16 ***
```

```
deck_F        -2.671528    0.335679  -7.959 1.74e-15 ***
age           -0.054914    0.008763  -6.266 3.70e-10 ***
deck_E         1.802546    0.376305   4.790 1.67e-06 ***
pclass_2       1.497212    0.269700   5.551 2.83e-08 ***
pclass_1       0.658065    0.351293   1.873  0.06103 .
sibsp         -0.305711    0.113440  -2.695  0.00704 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1230.15  on 915  degrees of freedom
Residual deviance:  653.81  on 908  degrees of freedom
AIC: 669.81


Number of Fisher Scoring iterations: 6
```

```
################################################################################
#                Model with influential points removed                       #
################################################################################
# Applying function to remove influential points via Cooks Distance:
filtering.result = remove_cooks_outliers(survived ~ ., data = train)
cat(nrow(filtering.result$high_cd_data),'rows were identified as outliers')
```

```
0 rows were identified as outliers
```

```
# plot(logmod)

# Since there were now rows identified as outliers, then this model will be the
# same as the initial binary model and need no be considered in the final
# model compare.


################################################################################
#                Model with high VIF preds removed                            #
################################################################################
library(car)
vif(lmod)
```

```
       age       sibsp       parch        fare   pclass_1   pclass_2      sex_M
   1.493399    1.258340    1.281859    1.727034   4.267734   1.560100   1.604117
embarked_C embarked_Q      deck_A      deck_B      deck_C     deck_D     deck_E
   1.513511    1.397623    4.532575    4.848833   7.604802   6.662169   6.091048
     deck_F
  14.919819
```

```
vif.model = glm(survived ~ . -deck_F, family = binomial, data = train)
vif(vif.model)
```

```
       age       sibsp       parch        fare   pclass_1   pclass_2      sex_M
   1.481259    1.236486    1.275733    1.729266   4.369376   1.472982   1.452735
embarked_C embarked_Q      deck_A      deck_B      deck_C     deck_D     deck_E
   1.511473    1.348123    1.988884    2.269651   2.863440   1.737448   1.558559
```

```r
# Removing deck_F from the model eliminates the multicollinearity completely.


###############################################################################
#                        Function to calc. cutoff                           #
###############################################################################
cutoff.prg<-function(pred,act){
# pred<-predicted_probabilities
# act<-true_labels
p<-seq(0,1,0.01)
n<-length(p)
out<-matrix(0,nrow=n,ncol=12)
for(i in 1:n){
predictions <- ifelse(pred >p[i], 1, 0)
confusion_matrix <- confusionMatrix(as.factor(predictions),as.factor(act),mode="prec_recall", positiv
out[i,]<-cbind(p=p[i],t(confusion_matrix[[4]]))
}
dimnames(out)[[2]]<-c("p","Sensitivity","Specificity","Pos Pred Value","Neg Pred Value","Precision",'
out
}
```

```r
# Finding the optimal cutoff
observations = train$survived
prob <- predict(lmod, train, type="response")

test_cutoff<-cutoff.prg(prob,observations)
```

```
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
```

```r
plot(test_cutoff[,1],test_cutoff[,8],xlab="cut off",ylab="F1")
```
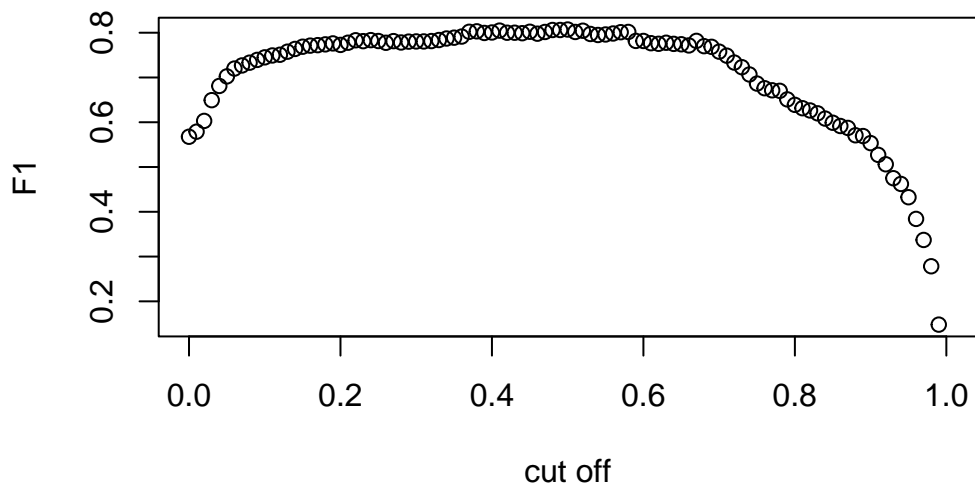
```r
optimal.cutoff = test_cutoff[which.max(test_cutoff[,8]),]
optimal.cutoff[1]
```

```
  p
0.5
```

```r
observations.train = train$survived
observations.test = test$survived

# Confusion matrix on base log model on train data
y_hat_prob = predict(lmod, train, type="response")
predictions.binary.model.train <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)
confusion.matrix.binary.model.train <- confusionMatrix(as.factor(predictions.binary.model.train),as.f
base.model.accuracy = confusion.matrix.binary.model.train$overall['Accuracy']
base.model.f1 = confusion.matrix.binary.model.train$byClass['F1']
base.model.train.summary = data.frame(
  Accuracy = base.model.accuracy,
  F1 = base.model.f1
)
row.names(base.model.train.summary) <- 'base.model.train'

# Confusion matrix on base log model on test data
y_hat_prob = predict(lmod, test, type="response")

predictions.binary.model.test <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)

confusion.matrix.binary.model.test <- confusionMatrix(as.factor(predictions.binary.model.test),as.fac

base.model.accuracy.test = confusion.matrix.binary.model.test$overall['Accuracy']
base.model.f1.test = confusion.matrix.binary.model.test$byClass['F1']

base.model.test.summary = data.frame(
```

```r
  Accuracy = base.model.accuracy.test,
  F1 = base.model.f1.test
)
row.names(base.model.test.summary) <- 'base.model.test'

# Confusion matrix on stepwise log model on train data
y_hat_prob = predict(binary.model.stepwise, train, type="response")
predictions.binary.model.step.train <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)
confusion.matrix.binary.model.step.train <- confusionMatrix(as.factor(predictions.binary.model.step.t
stepwise.model.accuracy = confusion.matrix.binary.model.step.train$overall['Accuracy']
stepwise.model.f1 = confusion.matrix.binary.model.step.train$byClass['F1']
stepwise.model.train.summary = data.frame(
  Accuracy = stepwise.model.accuracy,
  F1 = stepwise.model.f1
)
row.names(stepwise.model.train.summary) <- 'stepwise.model.train'

# Confusion matrix on stepwise log model on test data
y_hat_prob = predict(binary.model.stepwise, test, type="response")
predictions.binary.model.step.test <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)
confusion.matrix.binary.model.step.test <- confusionMatrix(as.factor(predictions.binary.model.step.te
stepwise.model.accuracy = confusion.matrix.binary.model.step.test$overall['Accuracy']
stepwise.model.f1 = confusion.matrix.binary.model.step.test$byClass['F1']
stepwise.model.test.summary = data.frame(
  Accuracy = stepwise.model.accuracy,
  F1 = stepwise.model.f1
)
row.names(stepwise.model.test.summary) <- 'stepwise.model.test'

# Confusion matrix on log model w/ insig. pred. removed
y_hat_prob = predict(binary.model.filtered, train, type="response")
predictions.binary.model.filtered.train <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)
confusion.matrix.binary.model.filtered.train <- confusionMatrix(as.factor(predictions.binary.model.fi
filtered.model.accuracy = confusion.matrix.binary.model.filtered.train$overall['Accuracy']
filtered.model.f1 = confusion.matrix.binary.model.filtered.train$byClass['F1']
filtered.model.train.summary = data.frame(
  Accuracy = filtered.model.accuracy,
  F1 = filtered.model.f1
)
row.names(filtered.model.train.summary) <- 'filtered.model.train'

# Confusion matrix on log model w/ insig. pred. removed
y_hat_prob = predict(binary.model.filtered, test, type="response")

predictions.binary.model.filtered.test <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)

confusion.matrix.binary.model.filtered.test <- confusionMatrix(as.factor(predictions.binary.model.fil

filtered.model.accuracy = confusion.matrix.binary.model.filtered.test$overall['Accuracy']
filtered.model.f1 = confusion.matrix.binary.model.filtered.test$byClass['F1']

filtered.model.test.summary = data.frame(
```

```
  Accuracy = filtered.model.accuracy,
  F1 = filtered.model.f1
)
row.names(filtered.model.test.summary) <- 'filtered.model.test'

# Confusion matrix on log model w/ high vif pred. removed
y_hat_prob = predict(vif.model, train, type="response")
predictions.binary.model.vif.train <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)
confusion.matrix.binary.model.vif.train <- confusionMatrix(as.factor(predictions.binary.model.vif.tra
vif.model.accuracy = confusion.matrix.binary.model.vif.train$overall['Accuracy']
vif.model.f1 = confusion.matrix.binary.model.vif.train$byClass['F1']
vif.model.train.summary = data.frame(
  Accuracy = vif.model.accuracy,
  F1 = vif.model.f1
)
row.names(vif.model.train.summary) <- 'vif.model.train'

# Confusion matrix on log model w/ high vif pred. removed
y_hat_prob = predict(vif.model, test, type="response")
predictions.binary.model.vif.test <- ifelse(y_hat_prob > optimal.cutoff[1], 1, 0)
confusion.matrix.binary.model.vif.test <- confusionMatrix(as.factor(predictions.binary.model.vif.test
vif.model.accuracy = confusion.matrix.binary.model.vif.test$overall['Accuracy']
vif.model.f1 = confusion.matrix.binary.model.vif.test$byClass['F1']
vif.model.test.summary = data.frame(
  Accuracy = vif.model.accuracy,
  F1 = vif.model.f1
)
row.names(vif.model.test.summary) <- 'vif.model.test'
```

```
data.frame(rbind(base.model.train.summary, base.model.test.summary,
                 stepwise.model.train.summary, stepwise.model.test.summary,
                 filtered.model.train.summary, filtered.model.test.summary,
                 vif.model.train.summary,vif.model.test.summary))
```

Table 3: Comparison of all log models performance

	Accuracy	F1
base.model.train	0.8471616	0.8071625
base.model.test	0.8367347	0.7730496
stepwise.model.train	0.8220524	0.7687943
stepwise.model.test	0.8418367	0.7669173
filtered.model.train	0.8471616	0.8066298
filtered.model.test	0.8367347	0.7730496
vif.model.train	0.8329694	0.7877947
vif.model.test	0.8367347	0.7746479

7 Challenger Models

Build an alternative model based on one of the following approaches to predict survival as applicable:logistic regression, decision tree, NN, or SVM, Poisson regression or negative binomial. Check the applicable model

assumptions. Apply in-sample and out-of-sample testing, back testing and review the comparative goodness of fit of the candidate models. Describe step by step your procedure to get to the best model and why you believe it is fit for purpose.

```r
library(MASS)

# Train base Poisson model:
poissonReg <- glm(survived ~ .,family=poisson, train)
summary(poissonReg)
```

```
Call:
glm(formula = survived ~ ., family = poisson, data = train)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  7.519e-01  3.472e-01   2.166 0.030341 *
age         -1.668e-02  4.337e-03  -3.847 0.000119 ***
sibsp       -7.014e-02  6.994e-02  -1.003 0.315889
parch        1.132e-02  6.704e-02   0.169 0.865869
fare         9.688e-05  1.025e-03   0.095 0.924696
pclass_1     3.485e-01  1.987e-01   1.754 0.079421 .
pclass_2     5.468e-01  1.515e-01   3.609 0.000307 ***
sex_M       -1.088e+00  1.199e-01  -9.069  < 2e-16 ***
embarked_C   7.474e-02  1.362e-01   0.549 0.583046
embarked_Q   5.098e-01  2.183e-01   2.335 0.019548 *
deck_A      -5.113e-01  4.521e-01  -1.131 0.258009
deck_B      -5.677e-01  4.064e-01  -1.397 0.162402
deck_C      -6.456e-01  3.912e-01  -1.650 0.098877 .
deck_D      -6.580e-01  3.820e-01  -1.722 0.084985 .
deck_E      -1.870e-01  3.526e-01  -0.530 0.595781
deck_F      -1.540e+00  3.413e-01  -4.513 6.38e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 672.00  on 915  degrees of freedom
Residual deviance: 400.84  on 900  degrees of freedom
AIC: 1158.8

Number of Fisher Scoring iterations: 5
```

```r
# compare the fitted model to the null model
1-pchisq(672.34-390.72, length(coef(poissonReg)))
```

```
[1] 0
```

```r
anova(poissonReg,test="Chi")
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL	NA	NA	915	671.9954	NA
age	1	1.1812909	914	670.8141	0.2770933
sibsp	1	0.7166530	913	670.0975	0.3972440
parch	1	5.6694845	912	664.4280	0.0172625
fare	1	25.6933009	911	638.7347	0.0000004
pclass_1	1	27.9481171	910	610.7866	0.0000001
pclass_2	1	17.1639017	909	593.6227	0.0000343
sex_M	1	118.6444922	908	474.9782	0.0000000
embarked_C	1	1.6665466	907	473.3116	0.1967217
embarked_Q	1	0.0858234	906	473.2258	0.7695556
deck_A	1	0.0631207	905	473.1627	0.8016298
deck_B	1	0.0077426	904	473.1549	0.9298829
deck_C	1	0.0326392	903	473.1223	0.8566319
deck_D	1	0.5268888	902	472.5954	0.4679178
deck_E	1	57.6272332	901	414.9682	0.0000000
deck_F	1	14.1318730	900	400.8363	0.0001704

```
drop1(poissonReg,test="Chi")
```

	Df	Deviance	AIC	LRT	Pr(>Chi)
	NA	400.8363	1158.836	NA	NA
age	1	416.0444	1172.044	15.2080992	0.0000963
sibsp	1	401.8958	1157.896	1.0594618	0.3033380
parch	1	400.8646	1156.865	0.0282941	0.8664193
fare	1	400.8452	1156.845	0.0088926	0.9248705
pclass_1	1	403.9035	1159.903	3.0671630	0.0798888
pclass_2	1	413.5888	1169.589	12.7524939	0.0003555
sex_M	1	489.9790	1245.979	89.1427013	0.0000000
embarked_C	1	401.1362	1157.136	0.2998976	0.5839466
embarked_Q	1	405.8500	1161.850	5.0137077	0.0251474
deck_A	1	402.0796	1158.080	1.2432486	0.2648459
deck_B	1	402.6560	1158.656	1.8197039	0.1773485
deck_C	1	403.3130	1159.313	2.4766883	0.1155453
deck_D	1	403.5115	1159.511	2.6751598	0.1019251
deck_E	1	401.1067	1157.107	0.2703827	0.6030752
deck_F	1	414.9682	1170.968	14.1318730	0.0001704

```
# Testing against test data
predicted_probs <- predict(poissonReg, newdata = test, type = "response")

# Get the results of different cutoff values
testResult<-cutoff.prg(predicted_probs,test$survived)
```

```
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
```

match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.
Warning in confusionMatrix.default(as.factor(predictions), as.factor(act), :
Levels are not in the same order for reference and data. Refactoring data to
match.

```r
# We get get the optimal p cut off value based on the maximum F1 score
plot(testResult[,1],testResult[,8],xlab="cut off",ylab="F1")
```
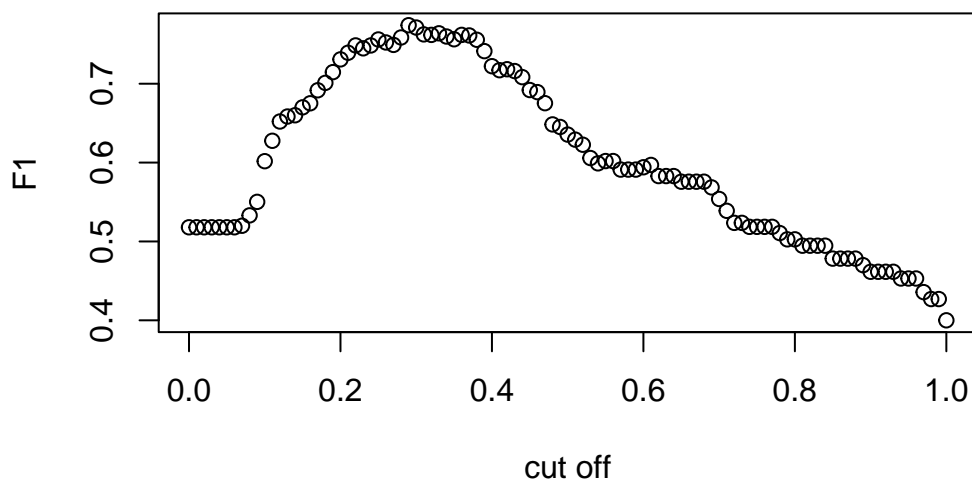


```r
testResult[which.max(testResult[,8]),]
```

p	Sensitivity	Specificity
0.2900000	0.8759124	0.7921569
Pos Pred Value	Neg Pred Value	Precision
0.6936416	0.9223744	0.6936416
Recall	F1	Prevalence
0.8759124	0.7741935	0.3494898
Detection Rate	Detection Prevalence	Balanced Accuracy
0.3061224	0.4413265	0.8340346

8 Model Limitation and Assumptions

The champion model selected is the filtered logistic regression model due to its superior train accuracy (85.04%) and F1 score (0.8099). The filtered logistic regression model was developed through backward elimination procedure, applied to the full base model. This process iteratively removed predictors with high p-values until only statistically significant predictors remained, based on threshold (p <0.05). The base model is used as the benchmark due to its similar performance and broader feature set.

"Survived = 5.56 - 0.055age - 0.23sibsp + 1pclass_1 + 1.9pclass_2 - 3.2sex_M + 0.84embarked_C + 0.86embarked_Q - 2.5deck_A - 1.9deck_B - 2.1deck_C - 3.1deck_D - 4.6deck_F"

To further evaluate and benchmark the logistic regression models, we computed RMSE, R^2, and MAE using the predicted probabilities against the actual binary outcomes. While traditional R^2 is not appropriate for logistic models, these metrics provide insight into how well the predicted probabilities align with observed outcomes. The Filtered model appears to have higher R^2 (0.4425>0.4342), lower RMSE(0.3945 < 0.3977) and MAE (0.1556 <0.1582). Additionally, model validation was done through train/test split with consistent performance across sets. The performance on test data and train data show minimal difference, indicating the model is robust, no overfitting problem.

To evaluate the stability and fit of the base and filtered logistic regression models, we conducted residual-based tests on their linear analogs. While logistic regression does not formally require normally distributed or homoscedastic residuals, these tests offer insights into model specification quality and residual behavior.

Breusch-Pagan Test: Both models exhibit statistically significant heteroskedasticity ($p < 0.01$)

Shapiro-Wilk Test: Both models show highly significant deviations from normality ($p < 0.01$)

Both models show similar residual behavior, with slight heteroskedasticity and non-normality. These findings do not violate logistic regression assumptions but suggest the model could be further improved through additional transformations or interaction terms.

Multicollinearity was significantly lower in the filtered model (max VIF: 3.69 vs. 16.63), supporting better generalization and interpretability. The VIF is less than 5, confirmed no severe multicollinearity.

```r
library(car)

# Normality Test
# Breusch Pagan Test on Logistic Model
BPtest_basemodel <- ols_test_breusch_pagan(lmod)
BPtest_filtered.model <- ols_test_breusch_pagan(binary.model.filtered)

# Shapiro-Wilk test
SWtest_basemodel <- shapiro.test(residuals(lmod))
SWtest_filteredmodel <- shapiro.test(residuals(binary.model.filtered))


# Combine into unified rows
diagnostics_summary <- data.frame(
  Model = c("Base.model", "Filtered.model"),
  BP_Statistic = c(BPtest_basemodel$bp, BPtest_filtered.model$bp),
  BP_pvalue = c(BPtest_basemodel$p, BPtest_filtered.model$p),
  Shapiro_W = c(SWtest_basemodel$statistic, SWtest_filteredmodel$statistic),
  Shapiro_pvalue = c(SWtest_basemodel$p.value, SWtest_filteredmodel$p.value)
)
```

```
diagnostics_summary
```

Table 6: Summary of Diagnostic tests on Filtered vs Base models

Model	BP_Statistic	BP_pvalue	Shapiro_W	Shapiro_pvalue
Base.model	10.11660	0.0014694	0.9752266	0
Filtered.model	10.31519	0.0013194	0.9751365	0

```
# Run VIF on the logistic model
ols_vif_tol(lmod)
```

Table 7: VIF Results for Base Logistic Regression Model

Variables	Tolerance	VIF
age	0.8112422	1.232677
sibsp	0.8133135	1.229538
parch	0.7793306	1.283152
fare	0.5101132	1.960349
pclass_1	0.2468157	4.051607
pclass_2	0.7943336	1.258917
sex_M	0.8512441	1.174751
embarked_C	0.7176662	1.393405
embarked_Q	0.8520166	1.173686
deck_A	0.2572099	3.887875
deck_B	0.1656314	6.037501
deck_C	0.1183078	8.452526
deck_D	0.1511669	6.615204
deck_E	0.1232885	8.111060
deck_F	0.0562819	17.767709

```
ols_vif_tol(binary.model.filtered)
```

Table 8: VIF Results for Filtered Logistic Regression Model

Variables	Tolerance	VIF
age	0.8251918	1.211839
sibsp	0.9323496	1.072559
pclass_1	0.2772087	3.607391
pclass_2	0.8044472	1.243090
sex_M	0.9085314	1.100677
embarked_C	0.7371420	1.356591
embarked_Q	0.8625868	1.159304
deck_A	0.6525446	1.532462
deck_B	0.5110353	1.956812
deck_C	0.4334379	2.307135
deck_D	0.5807436	1.721930
deck_F	0.3568101	2.802611

```
# Base Model - Train
observations.train <- train$survived
y_hat_base_train <- predict(lmod, train, type = "response")
pred_base_train <- ifelse(y_hat_base_train > optimal.cutoff[1], 1, 0)
ModelTrain_base <- data.frame(obs = observations.train, pred = pred_base_train)
log.train.base <- defaultSummary(ModelTrain_base)

# Base Model - Test
observations.test <- test$survived
y_hat_base_test <- predict(lmod, test, type = "response")
pred_base_test <- ifelse(y_hat_base_test > optimal.cutoff[1], 1, 0)
ModelTest_base <- data.frame(obs = observations.test, pred = pred_base_test)
log.test.base <- defaultSummary(ModelTest_base)

# Base Model - Train
observations.train <- train$survived
y_hat_base_train <- predict(binary.model.filtered, train, type = "response")
pred_base_train <- ifelse(y_hat_base_train > optimal.cutoff[1], 1, 0)
ModelTrain_base <- data.frame(obs = observations.train, pred = pred_base_train)
filtered.log.train.base <- defaultSummary(ModelTrain_base)

# Base Model - Test
observations.test <- test$survived
y_hat_base_test <- predict(binary.model.filtered, test, type = "response")
pred_base_test <- ifelse(y_hat_base_test > optimal.cutoff[1], 1, 0)
ModelTest_base <- data.frame(obs = observations.test, pred = pred_base_test)
filtered.test.base <- defaultSummary(ModelTest_base)
```

```
data.frame(rbind(log.train.base,log.test.base,filtered.log.train.base,filtered.test.base))
```

Table 9: Comparison of stats between base and filtered log models:

	RMSE	Rsquared	MAE
log.train.base	0.3909456	0.4631895	0.1528384
log.test.base	0.4040610	0.4177733	0.1632653
filtered.log.train.base	0.3909456	0.4627873	0.1528384
filtered.test.base	0.4040610	0.4177733	0.1632653

9 Ongoing Model Monitoring Plan

In order to maintain the effectiveness of the model, we would need to continue to test it on new data. Since the Titanic was a rare event, we do not have a lot of new data to test on the model, but we can still be prepared in case new data were to become available. The first step in monitoring the model is to determine specific thresholds that we expect the model to stay above. We would want the model to maintain certain R^2, RMSE, and MAE values in order to determine that the model is working correctly. One of the biggest concerns with our model is data drift. Since the Titanic sank over 100 years ago, the data that we are using from the model may not align with today relevant to ship travel today.

10 Conclusion

Summarize your results here. What is the best model for the data and why?

Appendix A: Check if 'sibsp' and 'parch' should be continuous or categorical

We don't see significant improvement between modeling these predictors as continuous or categorical, therefore we decided to leave them as continuous.

```r
library(car)
data.clean.ap1 = odata[, !(names(odata) %in% c("name", "ticket", "boat","body","home.dest"))]


################################################################################
#                           Data Augmentation                                  #
################################################################################
#Extract deck letter from cabin
data.clean.ap1$deck <- substr(data.clean.ap1$cabin, 1,1)

# Remove cabin col:
data.clean.ap1$cabin <- NULL


################################################################################
#                            Imputing data                                     #
################################################################################

# ---- Age----
#Replace NAs in age column with Median value
median_age <- median(data.clean.ap1$age, na.rm = TRUE)
data.clean.ap1 <- data.clean.ap1 %>%
  mutate(age = ifelse(is.na(age), median_age, age))

# ---- deck----
# For deck, since its a category, we decided to use KNN  to impute the column:

# Install if not already installed
# install.packages("VIM")
library(VIM)

# Replace "" with NA in the 'deck' column
data.clean.ap1$deck[data.clean.ap1$deck == ""] <- NA

# Convert 'cabin' to factor
data.clean.ap1$deck <- as.factor(data.clean.ap1$deck)

# Apply kNN imputation just to Cabin column
data.clean.ap1 <- kNN(data.clean.ap1, variable = "deck", k = 5)

# Check that NAs were imputed
# sum(is.na(data.clean$deck))        # Original
# sum(is.na(data.clean.imputed$deck)) # After
```

```
# Remove indicator col:
data.clean.ap1$deck_imp <- NULL


################################################################################
#                           Dummify Cat. cols                                  #
################################################################################

# Dummifying pclass:
data.clean.ap1$pclass_1 = ifelse(data.clean.ap1$pclass == 1, 1, 0)
data.clean.ap1$pclass_2 = ifelse(data.clean.ap1$pclass == 2, 1, 0)

# Dummifying sex:
data.clean.ap1$sex_M = ifelse(data.clean.ap1$sex == 'male', 1, 0)

# Dummifying embarked:
data.clean.ap1$embarked_C = ifelse(data.clean.ap1$embarked == 'C', 1, 0)
data.clean.ap1$embarked_Q = ifelse(data.clean.ap1$embarked == 'Q', 1, 0)

# Dummifying deck:
data.clean.ap1$deck_A = ifelse(data.clean.ap1$deck == 'A', 1, 0)
data.clean.ap1$deck_B = ifelse(data.clean.ap1$deck == 'B', 1, 0)
data.clean.ap1$deck_C = ifelse(data.clean.ap1$deck == 'C', 1, 0)
data.clean.ap1$deck_D = ifelse(data.clean.ap1$deck == 'D', 1, 0)
data.clean.ap1$deck_E = ifelse(data.clean.ap1$deck == 'E', 1, 0)
data.clean.ap1$deck_F = ifelse(data.clean.ap1$deck == 'F', 1, 0)
#data.clean.ap1$deck_G = ifelse(data.clean.ap1$deck == 'G', 1, 0) # removed due to causing issues

# Dummifying sibsp:
data.clean.ap1$sibsp_1 = ifelse(data.clean.ap1$sibsp == 1, 1, 0)
data.clean.ap1$sibsp_2 = ifelse(data.clean.ap1$sibsp == 2, 1, 0)
data.clean.ap1$sibsp_3 = ifelse(data.clean.ap1$sibsp == 3, 1, 0)
data.clean.ap1$sibsp_4 = ifelse(data.clean.ap1$sibsp == 4, 1, 0)
data.clean.ap1$sibsp_5 = ifelse(data.clean.ap1$sibsp == 5, 1, 0)
#data.clean.ap1$sibsp_8 = ifelse(data.clean.ap1$sibsp == 8, 1, 0) # removed due to causing issues

# Dummifying parch:
data.clean.ap1$parch_1 = ifelse(data.clean.ap1$parch == 1, 1, 0)
data.clean.ap1$parch_2 = ifelse(data.clean.ap1$parch == 2, 1, 0)
data.clean.ap1$parch_3 = ifelse(data.clean.ap1$parch == 3, 1, 0)
data.clean.ap1$parch_4 = ifelse(data.clean.ap1$parch == 4, 1, 0)
data.clean.ap1$parch_5 = ifelse(data.clean.ap1$parch == 5, 1, 0)
data.clean.ap1$parch_6 = ifelse(data.clean.ap1$parch == 6, 1, 0)
#data.clean.ap1$parch_9 = ifelse(data.clean.ap1$parch == 9, 1, 0) # removed due to causing issues

# Removing Dummified cols:
data.clean.ap1 = subset(data.clean.ap1, select  = -c(pclass, sex, embarked, deck))#, sibsp, parch))

data.clean.ap1 = na.omit(data.clean.ap1)

cat(nrow(odata) - nrow(data.clean.ap1),'rows were removed from original dataset')
```

```
2 rows were removed from original dataset
```

```
set.seed(567)
train_indices_ap1 = sample(1 : nrow(data.clean.ap1), size = 0.7005*nrow(data.clean.ap1), replace = FA
train.ap1 = data.clean.ap1[train_indices_ap1,]
test.ap1 = data.clean.ap1[-train_indices_ap1,]
cat("We are using:", nrow(train.ap1)/nrow(data.clean.ap1) * 100, '% of the data for training')
```

We are using: 70.03058 % of the data for training

```
mulvar_model.ap1 <- lm(survived ~ ., data = train.ap1)
summary(mulvar_model.ap1)
```

Call:
lm(formula = survived ~ ., data = train.ap1)

Residuals:
     Min   1Q Median   3Q   Max
-1.2866 -0.1941 -0.0224 0.1911 0.9739

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.1796068 0.1203338   9.803 < 2e-16 ***
age   -0.0064249 0.0010420 -6.166 1.06e-09 ***
sibsp   -0.0401162 0.0221954 -1.807 0.07104 .
parch   -0.0274112 0.0270379 -1.014 0.31095
fare   0.0004356 0.0002985   1.459 0.14481
pclass_1   0.1097829 0.0542228   2.025 0.04320 *
pclass_2   0.1381701 0.0312063   4.428 1.07e-05 ***
sex_M   -0.4129619 0.0252816 -16.335 < 2e-16 ***
embarked_C   0.0461410 0.0323127   1.428 0.15366
embarked_Q   0.1210184 0.0426024   2.841 0.00461 **
deck_A   -0.3494376 0.1454048 -2.403 0.01646 *
deck_B   -0.3123626 0.1372942 -2.275 0.02314 *
deck_C   -0.3040370 0.1331009 -2.284 0.02259 *
deck_D   -0.3432119 0.1290127 -2.660 0.00795 **
deck_E   0.0041655 0.1243264   0.034 0.97328
deck_F   -0.5934388 0.1184958 -5.008 6.63e-07 ***
sibsp_1   0.0567316 0.0355896   1.594 0.11128
sibsp_2   0.1212082 0.0797596   1.520 0.12895
sibsp_3   -0.1501858 0.1195550 -1.256 0.20937
sibsp_4   -0.2298081 0.1296372 -1.773 0.07662 .
sibsp_5   -0.2293432 0.2019073 -1.136 0.25631
parch_1   0.1426458 0.0454040   3.142 0.00174 **
parch_2   0.1456666 0.0715147   2.037 0.04196 *
parch_3   0.2704118 0.1739380   1.555 0.12039
parch_4   -0.0799175 0.2038416 -0.392 0.69511
parch_5   0.0679073 0.2385574   0.285 0.77597
parch_6   0.0143690 0.2895211   0.050 0.96043

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.34 on 889 degrees of freedom
Multiple R-squared:  0.5273,     Adjusted R-squared:  0.5135
F-statistic: 38.14 on 26 and 889 DF,  p-value: < 2.2e-16
```

```
vif(mulvar_model.ap1)
```

```
         age        sibsp        parch         fare     pclass_1     pclass_2       sex_M
    1.460949     3.467000     4.494487     1.922834     4.492433     1.297793     1.164100
 embarked_C embarked_Q       deck_A       deck_B       deck_C       deck_D       deck_E
    1.437150     1.172137     5.135825     8.715727    13.411354     9.672668    12.015361
      deck_F       sibsp_1      sibsp_2      sibsp_3      sibsp_4      sibsp_5      parch_1
   26.716246     1.876291     1.441990     1.464254     1.863037     1.404398     1.911901
     parch_2       parch_3      parch_4      parch_5      parch_6
    3.083293     1.301393     1.431435     1.472004     1.446997
```

```
lmod.ap1 <- glm(as.factor(survived) ~ ., family = binomial, data = train.ap1)
summary(lmod.ap1)
```

```
Call:
glm(formula = as.factor(survived) ~ ., family = binomial, data = train.ap1)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   5.253e+00  1.384e+00   3.797 0.000147 ***
age          -5.481e-02  9.965e-03  -5.500 3.79e-08 ***
sibsp        -1.946e+00  1.291e+02  -0.015 0.987976
parch        -1.642e+00  1.623e+02  -0.010 0.991928
fare          1.057e-03  2.428e-03   0.435 0.663393
pclass_1      1.156e+00  4.819e-01   2.398 0.016480 *
pclass_2      1.496e+00  3.095e-01   4.834 1.34e-06 ***
sex_M        -2.982e+00  2.464e-01 -12.103  < 2e-16 ***
embarked_C    6.410e-01  2.868e-01   2.235 0.025424 *
embarked_Q    1.316e+00  3.735e-01   3.523 0.000427 ***
deck_A       -2.866e+00  1.503e+00  -1.907 0.056492 .
deck_B       -2.498e+00  1.479e+00  -1.689 0.091220 .
deck_C       -2.624e+00  1.445e+00  -1.817 0.069262 .
deck_D       -2.971e+00  1.415e+00  -2.100 0.035711 *
deck_E        3.196e-01  1.404e+00   0.228 0.819965
deck_F       -4.753e+00  1.358e+00  -3.499 0.000467 ***
sibsp_1       2.065e+00  1.291e+02   0.016 0.987240
sibsp_2       4.356e+00  2.583e+02   0.017 0.986546
sibsp_3       3.607e+00  3.874e+02   0.009 0.992572
sibsp_4       4.557e+00  5.166e+02   0.009 0.992962
sibsp_5      -6.723e+00  1.240e+03  -0.005 0.995674
parch_1       2.597e+00  1.623e+02   0.016 0.987237
parch_2       4.192e+00  3.247e+02   0.013 0.989698
parch_3       6.236e+00  4.870e+02   0.013 0.989785
parch_4       6.227e+00  6.494e+02   0.010 0.992349
parch_5       8.544e+00  8.117e+02   0.011 0.991602
parch_6      -4.112e+00  1.770e+03  -0.002 0.998146
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1223.12  on 915  degrees of freedom
Residual deviance:  602.86  on 889  degrees of freedom
AIC: 656.86

Number of Fisher Scoring iterations: 15
```

```
vif(lmod.ap1)
```

```
          age         sibsp         parch          fare      pclass_1      pclass_2
1.653195e+00 7.591027e+05 1.634339e+06 1.757536e+00 4.949336e+00 1.555093e+00
        sex_M     embarked_C    embarked_Q        deck_A        deck_B        deck_C
1.404155e+00 1.549826e+00 1.415978e+00 1.012962e+01 1.475410e+01 2.517074e+01
       deck_D        deck_E        deck_F       sibsp_1       sibsp_2       sibsp_3
1.980371e+01 1.464836e+01 4.346983e+01 3.294475e+05 1.808931e+05 1.937224e+05
      sibsp_4       sibsp_5       parch_1       parch_2       parch_3       parch_4
1.974919e+05 1.372219e+00 3.139987e+05 6.986245e+05 2.469813e+05 2.280957e+05
      parch_5       parch_6
4.056249e+05 1.434520e+00
```

```
y_hat_mulvar_train.ap1<-predict(mulvar_model.ap1, data = train.ap1)
predictions_train.ap1 <- ifelse(y_hat_mulvar_train.ap1 > 0.5, 1, 0)
ModelTrain_mulvar.ap1<-data.frame(obs = train.ap1$survived, pred=predictions_train.ap1)
linear.train.ap1 <- defaultSummary(ModelTrain_mulvar.ap1)

y_hat_mulvar_test.ap1<-predict(mulvar_model.ap1, newdata = test.ap1)
predictions_test.ap1 <- ifelse(y_hat_mulvar_test.ap1 > 0.5, 1, 0)
ModelTest_mulvar.ap1<-data.frame(obs = test.ap1$survived, pred=predictions_test.ap1)
linear.test.ap1 <- defaultSummary(ModelTest_mulvar.ap1)


y_hat_log_train.ap1<-predict(lmod.ap1, data = train.ap1)
predictions_log_train.ap1 <- ifelse(y_hat_log_train.ap1 > 0.5, 1, 0)
ModelTrain_lmod.ap1<-data.frame(obs = train.ap1$survived, pred=predictions_log_train.ap1)
log.train.ap1 <- defaultSummary(ModelTrain_lmod.ap1)

y_hat_log_test.ap1<-predict(lmod.ap1, newdata = test.ap1)
predictions_log_test.ap1 <- ifelse(y_hat_log_test.ap1 > 0.5, 1, 0)
ModelTest_lmod.ap1<-data.frame(obs = test.ap1$survived, pred=predictions_log_test.ap1)
log.test.ap1 <- defaultSummary(ModelTest_lmod.ap1)

data.frame(rbind(linear.train.ap1,linear.test.ap1,log.train.ap1,log.test.ap1))
```

	RMSE	Rsquared	MAE
linear.train.ap1	0.3796114	0.4824536	0.1441048
linear.test.ap1	0.3976975	0.4365256	0.1581633

	RMSE	Rsquared	MAE
log.train.ap1	0.3964912	0.4429840	0.1572052
log.test.ap1	0.4008919	0.4214242	0.1607143

```
confusion_matrix_mulvar_train.ap1 <- confusionMatrix(as.factor(predictions_train.ap1), as.factor(trai
confusion_matrix_mulvar_train.ap1
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 502  73
         1  59 282

               Accuracy : 0.8559
                 95% CI : (0.8315, 0.878)
    No Information Rate : 0.6124
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6942

 Mcnemar's Test P-Value : 0.2578

              Precision : 0.8270
                 Recall : 0.7944
                     F1 : 0.8103
             Prevalence : 0.3876
         Detection Rate : 0.3079
   Detection Prevalence : 0.3723
      Balanced Accuracy : 0.8446

       'Positive' Class : 1
```

```
confusion_matrix_mulvar_test.ap1 <- confusionMatrix(as.factor(predictions_test.ap1), as.factor(test.a
confusion_matrix_mulvar_test.ap1
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 216  31
         1  31 114

               Accuracy : 0.8418
                 95% CI : (0.8019, 0.8765)
    No Information Rate : 0.6301
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6607
```

```
        Mcnemar's Test P-Value : 1

                   Precision : 0.7862
                      Recall : 0.7862
                          F1 : 0.7862
                  Prevalence : 0.3699
              Detection Rate : 0.2908
        Detection Prevalence : 0.3699
           Balanced Accuracy : 0.8304

            'Positive' Class : 1
```

```
confusion_matrix_log_train.ap1 <- confusionMatrix(as.factor(predictions_log_train.ap1), as.factor(tra
confusion_matrix_log_train.ap1
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 523 106
         1  38 249

               Accuracy : 0.8428
                 95% CI : (0.8176, 0.8658)
    No Information Rate : 0.6124
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6568

 Mcnemar's Test P-Value : 2.36e-08

              Precision : 0.8676
                 Recall : 0.7014
                     F1 : 0.7757
             Prevalence : 0.3876
         Detection Rate : 0.2718
   Detection Prevalence : 0.3133
      Balanced Accuracy : 0.8168

       'Positive' Class : 1
```

```
confusion_matrix_log_test.ap1 <- confusionMatrix(as.factor(predictions_log_test.ap1), as.factor(test.
confusion_matrix_log_test.ap1
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
```

```
          0 226   42
          1   21 103

               Accuracy : 0.8393
                 95% CI : (0.7991, 0.8742)
    No Information Rate : 0.6301
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6446

 Mcnemar's Test P-Value : 0.01174

              Precision : 0.8306
                 Recall : 0.7103
                     F1 : 0.7658
             Prevalence : 0.3699
         Detection Rate : 0.2628
   Detection Prevalence : 0.3163
      Balanced Accuracy : 0.8127

       'Positive' Class : 1
```

```r
library(car)
data.clean.ap2 = odata[, !(names(odata) %in% c("name", "ticket", "boat","body","home.dest"))]

################################################################################
#                          Data Augmentation                                   #
################################################################################
#Extract deck letter from cabin
data.clean.ap2$deck <- substr(data.clean.ap2$cabin, 1,1)

# Remove cabin col:
data.clean.ap2$cabin <- NULL

################################################################################
#                           Imputing data                                      #
################################################################################

# ---- Age----
#Replace NAs in age column with Median value
median_age <- median(data.clean.ap2$age, na.rm = TRUE)
data.clean.ap2 <- data.clean.ap2 %>%
  mutate(age = ifelse(is.na(age), median_age, age))

# ---- deck----
# For deck, since its a category, we decided to use KNN  to impute the column:

# Install if not already installed
# install.packages("VIM")
library(VIM)

# Replace "" with NA in the 'deck' column
```

```r
data.clean.ap2$deck[data.clean.ap2$deck == ""] <- NA

# Convert 'cabin' to factor
data.clean.ap2$deck <- as.factor(data.clean.ap2$deck)

# Apply kNN imputation just to Cabin column
data.clean.ap2 <- kNN(data.clean.ap2, variable = "deck", k = 5)

# Check that NAs were imputed
# sum(is.na(data.clean$deck))          # Original
# sum(is.na(data.clean.imputed$deck)) # After

# Remove indicator col:
data.clean.ap2$deck_imp <- NULL


################################################################################
#                          Dummify Cat. cols                                  #
################################################################################

# Dummifying pclass:
data.clean.ap2$pclass_1 = ifelse(data.clean.ap2$pclass == 1, 1, 0)
data.clean.ap2$pclass_2 = ifelse(data.clean.ap2$pclass == 2, 1, 0)

# Dummifying sex:
data.clean.ap2$sex_M = ifelse(data.clean.ap2$sex == 'male', 1, 0)

# Dummifying embarked:
data.clean.ap2$embarked_C = ifelse(data.clean.ap2$embarked == 'C', 1, 0)
data.clean.ap2$embarked_Q = ifelse(data.clean.ap2$embarked == 'Q', 1, 0)

# Dummifying deck:
data.clean.ap2$deck_A = ifelse(data.clean.ap2$deck == 'A', 1, 0)
data.clean.ap2$deck_B = ifelse(data.clean.ap2$deck == 'B', 1, 0)
data.clean.ap2$deck_C = ifelse(data.clean.ap2$deck == 'C', 1, 0)
data.clean.ap2$deck_D = ifelse(data.clean.ap2$deck == 'D', 1, 0)
data.clean.ap2$deck_E = ifelse(data.clean.ap2$deck == 'E', 1, 0)
data.clean.ap2$deck_F = ifelse(data.clean.ap2$deck == 'F', 1, 0)
#data.clean.ap2$deck_G = ifelse(data.clean.ap2$deck == 'G', 1, 0) # removed due to causing issues

# Dummifying sibsp to 2 categories:
data.clean.ap2$sibsp_y = ifelse(data.clean.ap2$sibsp > 0, 1, 0)

# Dummifying parch to 2 categories:
data.clean.ap2$parch_y = ifelse(data.clean.ap2$parch > 0, 1, 0)

# Removing Dummified cols:
data.clean.ap2 = subset(data.clean.ap2, select  = -c(pclass, sex, embarked, deck))#, sibsp, parch))

data.clean.ap2 = na.omit(data.clean.ap2)

cat(nrow(odata) - nrow(data.clean.ap2),'rows were removed from original dataset')
```

2 rows were removed from original dataset

```r
set.seed(567)
train_indices_ap2 = sample(1 : nrow(data.clean.ap2), size = 0.7005*nrow(data.clean.ap2), replace = FA
train.ap2 = data.clean.ap2[train_indices_ap2,]
test.ap2 = data.clean.ap2[-train_indices_ap2,]
cat("We are using:", nrow(train.ap2)/nrow(data.clean.ap2) * 100, '% of the data for training')
```

We are using: 70.03058 % of the data for training

```r
mulvar_model.ap2 <- lm(survived ~ ., data = train.ap2)
summary(mulvar_model.ap2)
```

Call:
lm(formula = survived ~ ., data = train.ap2)

Residuals:
```
     Min       1Q    Median       3Q      Max
-1.27243  -0.19451  -0.02769  0.19202  0.96436
```

Coefficients:
```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.1720948  0.1212992   9.663  < 2e-16 ***
age         -0.0061866  0.0010236  -6.044 2.20e-09 ***
sibsp       -0.0867184  0.0185029  -4.687 3.21e-06 ***
parch       -0.0487147  0.0214741  -2.269 0.023534 *
fare         0.0003848  0.0003005   1.281 0.200641
pclass_1     0.1055453  0.0557446   1.893 0.058629 .
pclass_2     0.1376903  0.0314471   4.378 1.34e-05 ***
sex_M       -0.4270550  0.0253448 -16.850  < 2e-16 ***
embarked_C   0.0585880  0.0319987   1.831 0.067439 .
embarked_Q   0.1061446  0.0425937   2.492 0.012881 *
deck_A      -0.3315651  0.1464353  -2.264 0.023797 *
deck_B      -0.2852592  0.1389479  -2.053 0.040362 *
deck_C      -0.2951467  0.1350558  -2.185 0.029120 *
deck_D      -0.3605291  0.1318846  -2.734 0.006386 **
deck_E       0.0078459  0.1256544   0.062 0.950226
deck_F      -0.5745493  0.1193029  -4.816 1.72e-06 ***
sibsp_y      0.1224343  0.0370630   3.303 0.000993 ***
parch_y      0.1573960  0.0479445   3.283 0.001067 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 0.3436 on 898 degrees of freedom
Multiple R-squared: 0.5124,	Adjusted R-squared: 0.5031
F-statistic: 55.5 on 17 and 898 DF, p-value: < 2.2e-16

```r
vif(mulvar_model.ap2)
```

```
        age        sibsp       parch        fare    pclass_1    pclass_2      sex_M
   1.380228     2.359204    2.776007    1.908310    4.649205    1.290432    1.145551
embarked_C embarked_Q      deck_A      deck_B      deck_C      deck_D      deck_E
   1.379990     1.147247    5.439805    8.740948   13.154666    9.148653   11.598396
     deck_F      sibsp_y     parch_y
  26.237760     2.297316    3.213776
```

```
lmod.ap2 <- glm(as.factor(survived) ~ ., family = binomial, data = train.ap2)
summary(lmod.ap2)
```

```
Call:
glm(formula = as.factor(survived) ~ ., family = binomial, data = train.ap2)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.1275081  1.3016899   3.939 8.18e-05 ***
age         -0.0558992  0.0094958  -5.887 3.94e-09 ***
sibsp       -1.0266480  0.2640080  -3.889 0.000101 ***
parch       -0.2791962  0.1939136  -1.440 0.149925
fare         0.0007343  0.0024042   0.305 0.760042
pclass_1     1.1421684  0.4862413   2.349 0.018825 *
pclass_2     1.4015232  0.2992294   4.684 2.82e-06 ***
sex_M       -3.0275901  0.2424805 -12.486  < 2e-16 ***
embarked_C   0.6732550  0.2814737   2.392 0.016762 *
embarked_Q   1.2284326  0.3679272   3.339 0.000841 ***
deck_A      -2.5767916  1.4219365  -1.812 0.069960 .
deck_B      -2.2194317  1.4042525  -1.581 0.113991
deck_C      -2.3552547  1.3682224  -1.721 0.085179 .
deck_D      -2.9570245  1.3451465  -2.198 0.027928 *
deck_E       0.4470949  1.3202938   0.339 0.734886
deck_F      -4.4528239  1.2754924  -3.491 0.000481 ***
sibsp_y      1.3882952  0.3947966   3.516 0.000437 ***
parch_y      1.2304958  0.4199141   2.930 0.003386 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1223.12  on 915  degrees of freedom
Residual deviance:  620.68  on 898  degrees of freedom
AIC: 656.68

Number of Fisher Scoring iterations: 6
```

```
vif(lmod.ap2)
```

```
        age        sibsp       parch        fare    pclass_1    pclass_2      sex_M
   1.619429     3.641267    2.989704    1.753639    5.122869    1.534060    1.410059
embarked_C embarked_Q      deck_A      deck_B      deck_C      deck_D      deck_E
   1.487757     1.363260    9.852461   13.562280   21.612617   16.823106   13.659175
```

```
     deck_F     sibsp_y     parch_y
  39.454519    3.482784    3.287406
```

```
y_hat_mulvar_train.ap2<-predict(mulvar_model.ap2, data = train.ap2)
predictions_train.ap2 <- ifelse(y_hat_mulvar_train.ap2 > 0.5, 1, 0)
ModelTrain_mulvar.ap2<-data.frame(obs = train.ap2$survived, pred=predictions_train.ap2)
linear.train.ap2 <- defaultSummary(ModelTrain_mulvar.ap2)

y_hat_mulvar_test.ap2<-predict(mulvar_model.ap2, newdata = test.ap2)
predictions_test.ap2 <- ifelse(y_hat_mulvar_test.ap2 > 0.5, 1, 0)
ModelTest_mulvar.ap2<-data.frame(obs = test.ap2$survived, pred=predictions_test.ap2)
linear.test.ap2 <- defaultSummary(ModelTest_mulvar.ap2)


y_hat_log_train.ap2<-predict(lmod.ap2, data = train.ap2)
predictions_log_train.ap2 <- ifelse(y_hat_log_train.ap2 > 0.5, 1, 0)
ModelTrain_lmod.ap2<-data.frame(obs = train.ap2$survived, pred=y_hat_log_train.ap2)
log.train.ap2 <- defaultSummary(ModelTrain_mulvar.ap2)

y_hat_log_test.ap2<-predict(lmod.ap2, newdata = test.ap2)
predictions_log_test.ap2 <- ifelse(y_hat_log_test.ap2 > 0.5, 1, 0)
ModelTest_lmod.ap2<-data.frame(obs = test.ap2$survived, pred=predictions_log_test.ap2)
log.test.ap2 <- defaultSummary(ModelTest_lmod.ap2)


data.frame(rbind(linear.train.ap2,linear.test.ap2,log.train.ap2,log.test.ap2))
```

	RMSE	Rsquared	MAE
linear.train.ap2	0.3909456	0.4570397	0.1528384
linear.test.ap2	0.4040610	0.4221825	0.1632653
log.train.ap2	0.3909456	0.4570397	0.1528384
log.test.ap2	0.4008919	0.4212961	0.1607143

```
confusion_matrix_mulvar_train.ap2 <- confusionMatrix(as.factor(predictions_train.ap2), as.factor(tra
confusion_matrix_mulvar_train.ap2
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0    1
         0 498   77
         1  63  278

               Accuracy : 0.8472
                 95% CI : (0.8222, 0.8699)
    No Information Rate : 0.6124
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6757
```

```
         Mcnemar's Test P-Value : 0.2719

                    Precision : 0.8152
                       Recall : 0.7831
                           F1 : 0.7989
                   Prevalence : 0.3876
              Detection Rate : 0.3035
       Detection Prevalence : 0.3723
           Balanced Accuracy : 0.8354

               'Positive' Class : 1
```

```
confusion_matrix_mulvar_test.ap2 <- confusionMatrix(as.factor(predictions_test.ap2), as.factor(test.a
confusion_matrix_mulvar_test.ap2
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0    1
         0 215   32
         1  32  113

                    Accuracy : 0.8367
                      95% CI : (0.7964, 0.8719)
         No Information Rate : 0.6301
         P-Value [Acc > NIR] : <2e-16

                       Kappa : 0.6498

     Mcnemar's Test P-Value : 1

                    Precision : 0.7793
                       Recall : 0.7793
                           F1 : 0.7793
                   Prevalence : 0.3699
              Detection Rate : 0.2883
       Detection Prevalence : 0.3699
           Balanced Accuracy : 0.8249

               'Positive' Class : 1
```

```
confusion_matrix_log_train.ap2 <- confusionMatrix(as.factor(predictions_log_train.ap2), as.factor(tra
confusion_matrix_log_train.ap2
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0    1
         0 523  111
```

```
           1  38 244
```

```
              Accuracy : 0.8373
                95% CI : (0.8118, 0.8607)
   No Information Rate : 0.6124
   P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.6439

 Mcnemar's Test P-Value : 3.669e-09

             Precision : 0.8652
                Recall : 0.6873
                    F1 : 0.7661
            Prevalence : 0.3876
        Detection Rate : 0.2664
  Detection Prevalence : 0.3079
     Balanced Accuracy : 0.8098

       'Positive' Class : 1
```

```
confusion_matrix_log_test.ap2 <- confusionMatrix(as.factor(predictions_log_test.ap2), as.factor(test.
confusion_matrix_log_test.ap2
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0    1
         0 227   43
         1  20  102

              Accuracy : 0.8393
                95% CI : (0.7991, 0.8742)
   No Information Rate : 0.6301
   P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.6436

 Mcnemar's Test P-Value : 0.005576

             Precision : 0.8361
                Recall : 0.7034
                    F1 : 0.7640
            Prevalence : 0.3699
        Detection Rate : 0.2602
  Detection Prevalence : 0.3112
     Balanced Accuracy : 0.8112

       'Positive' Class : 1
```

References

National Oceanic and Atmospheric Administration (NOAA). (2023). *RMS titanic – history and significance.* https://www.noaa.gov/office-of-general-counsel/gc-international-section/rms-titanic-history-and-significance

Statology. (2025). *How to measure correlation between categorical variables.* https://www.statology.org/correlation-between-categorical-variables/