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

09 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

1 Instructions: 2

2 Executive Summary 3

3 Introduction 3

4 Description of the data and quality 4
- 4.1 Loading the data . . . 4
- 4.2 Removing un-needed columns . . . 4
- 4.3 Data Augmentation . . . 4
- 4.4 Initial Check for Missing values . . . 5
- 4.5 Imputing data . . . 5
- 4.6 Dummifying Columns: . . . 7
- 4.7 Remove NA rows and deck_G . . . 8
- 4.8 Divide into Test / Train . . . 8
- 4.9 EDA . . . 9

5 Model Development Process 14

6 Model Performance Testing 17

7 Challenger Models 26

8 Model Limitation and Assumptions 30

9 Ongoing Model Monitoring Plan 33

10 Conclusion 33

Appendix A: Check if 'sibsp' and 'parch' should be continuous or categorical 34

References 48

List of Figures

1 Percentage of Missing Values . . . 5
2 Percentage of Missing Values after Imputation . . . 7
3 Histogram of Deck_G . . . 8
4 Histogram of survival vs age . . . 9
5 Barchart of survival . . . 10
6 Barchart of survival vs Num. of siblings . . . 10
7 Scatter plots of all variables in train data . . . 12
8 Correlation Matrix . . . 14
9 4x4 standard plots for log. model . . . 17

List of Tables

2 Summary Stats. of base log. model . . . 16
3 Comparison of all log models performance . . . 26
7 Summary of Diagnostic tests on Filtered vs Base models . . . 31
8 VIF Results for Base Logistic Regression Model . . . 31
9 VIF Results for Filtered Logistic Regression Model . . . 32
10 Comparison of stats between base and filtered log models: . . . 33
11 Comparison of our base model, best performing developed model, and challenger model . . . 33

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 (See 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))
```

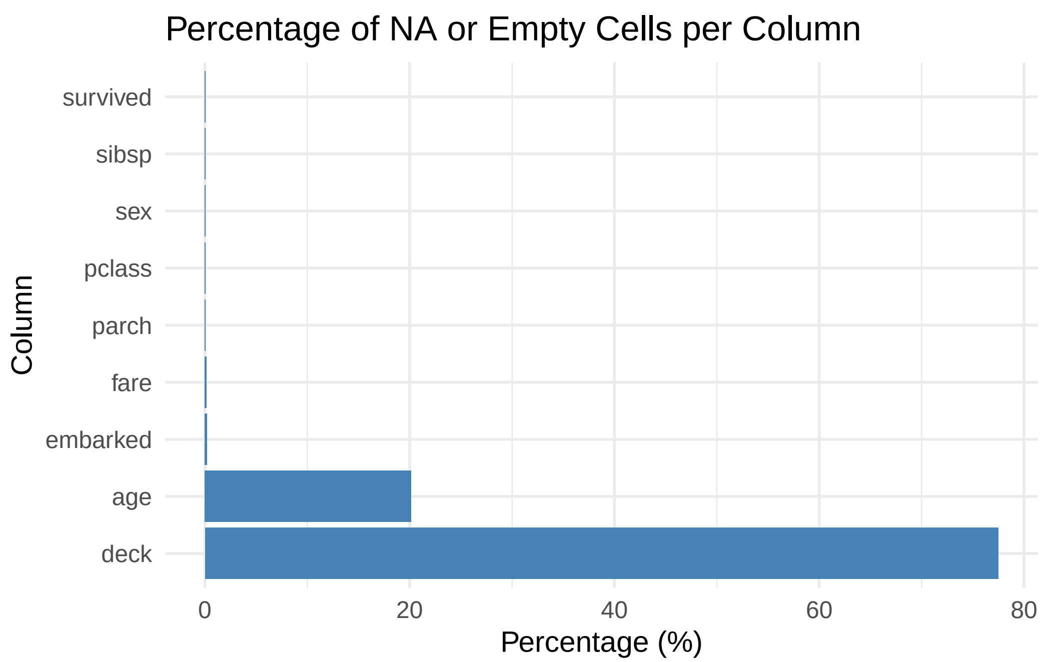


Figure 1: Percentage of Missing Values

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)
```

```
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)
```

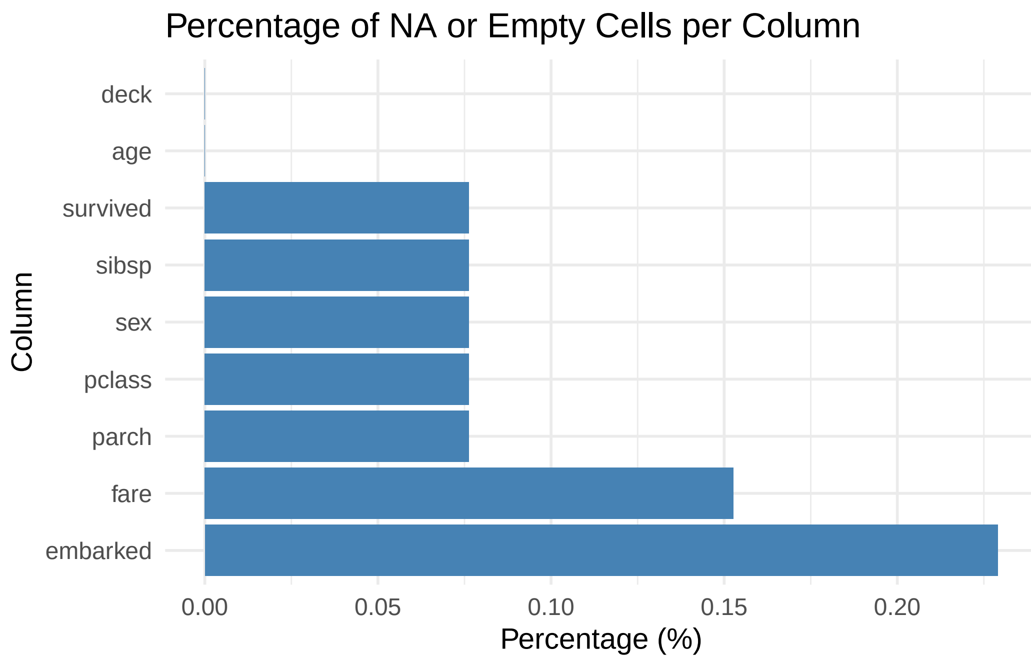


Figure 2: Percentage of Missing Values after Imputation

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 (See 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")
```

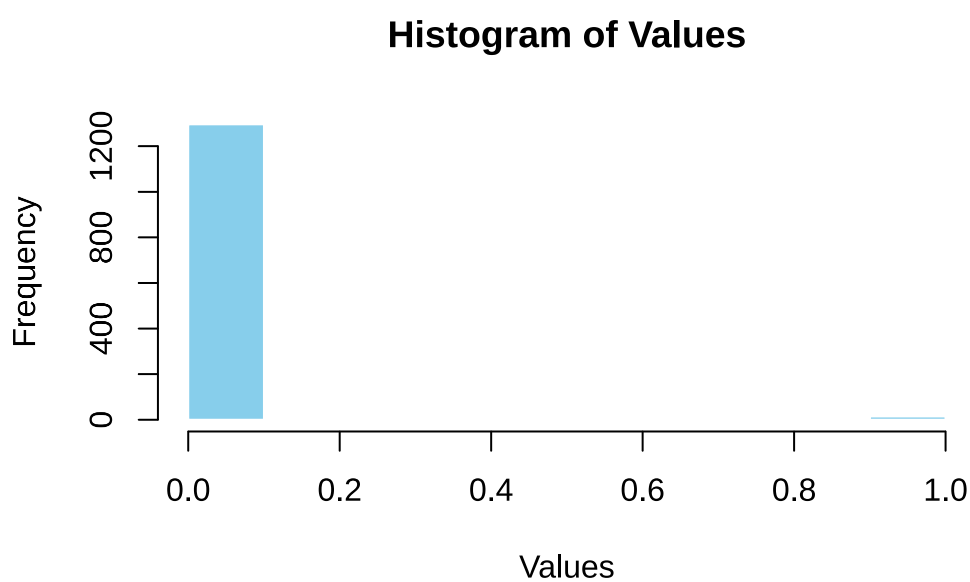


Figure 3: Histogram of Deck_G

```
# 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)
```

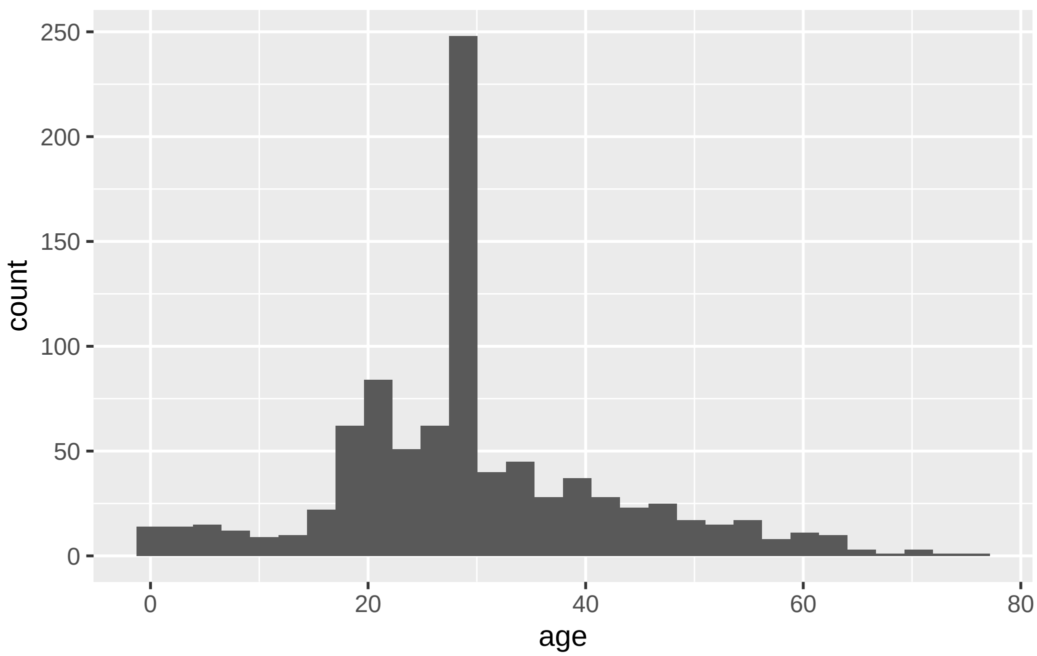


Figure 4: Histogram of survival vs age

```
# Bar chart showing that more people died than survived
ggplot(train, aes(survived)) +
  geom_bar()
```

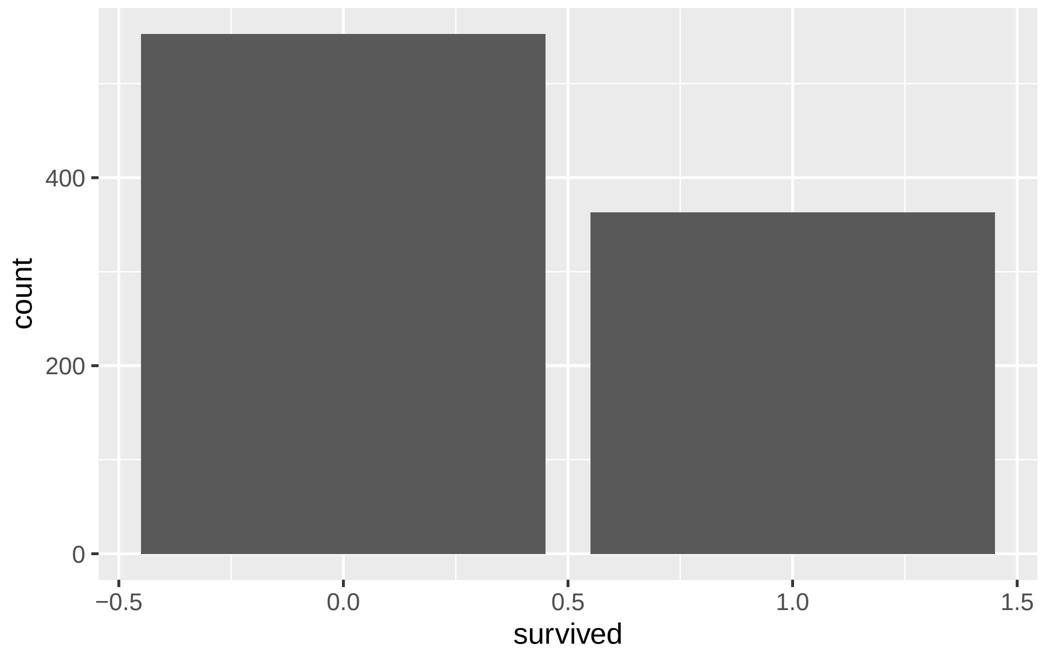


Figure 5: Barchart of survival

```r
# 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')
```

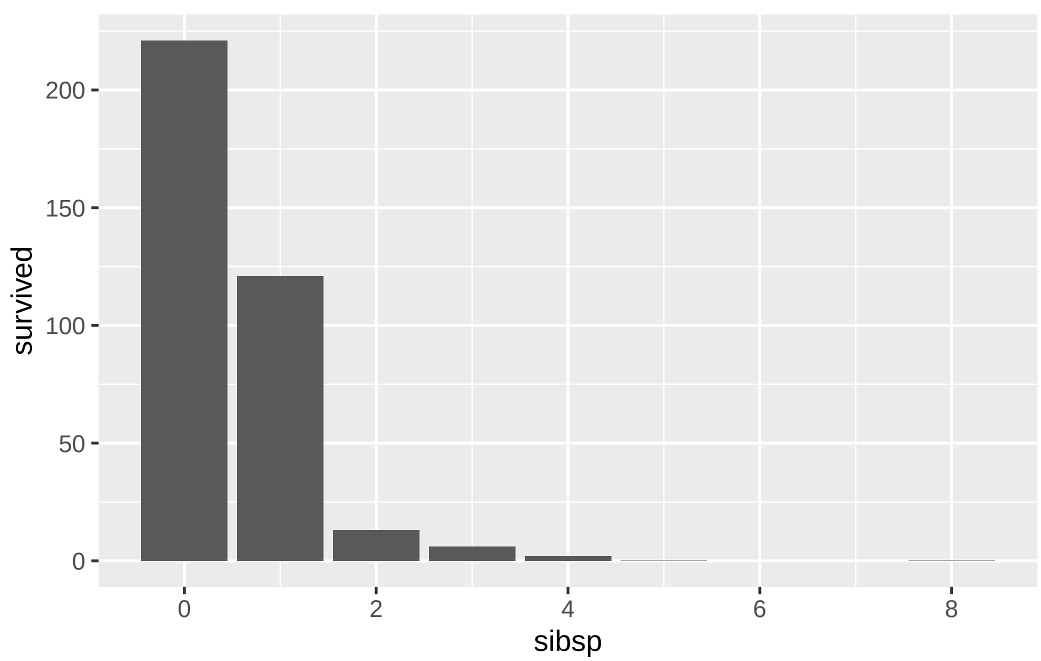


Figure 6: Barchart of survival vs Num. of siblings

```r
cor(train)
```

```
              survived          age        sibsp        parch         fare
```

```
survived    1.00000000 -0.04599216 -0.027543645  0.095720424  0.24962042
age        -0.04599216  1.00000000 -0.154952394 -0.122885529  0.16447722
sibsp      -0.02754365 -0.15495239  1.000000000  0.355216328  0.16529618
parch       0.09572042 -0.12288553  0.355216328  1.000000000  0.20987827
fare        0.24962042  0.16447722  0.165296185  0.209878267  1.00000000
pclass_1    0.28432063  0.34847627 -0.022306615 -0.016451014  0.59116304
pclass_2    0.06118909  0.01898825 -0.069089243 -0.017696720 -0.12563735
sex_M      -0.53377358  0.05385130 -0.127066747 -0.243501842 -0.20177640
embarked_C  0.14757622  0.05701305 -0.061215242 -0.001165488  0.27208779
embarked_Q -0.02542950 -0.03373092 -0.060221066 -0.093253422 -0.12699238
deck_A      0.03351255  0.11849174 -0.066920639 -0.057789108  0.05775137
deck_B      0.16163293  0.11668084 -0.018056185  0.073894445  0.45996364
deck_C      0.16746961  0.15595995  0.026704925 -0.051690185  0.30933961
deck_D      0.09857507  0.04928304 -0.005406569 -0.019083135  0.01011501
deck_E      0.32212013  0.15182289 -0.084456592 -0.036003896 -0.01993646
deck_F     -0.47931430 -0.28802595  0.064953732  0.028147672 -0.41755468
              pclass_1     pclass_2       sex_M   embarked_C  embarked_Q
survived    0.28432063  0.061189091 -0.53377358  0.147576220 -0.02542950
age         0.34847627  0.018988253  0.05385130  0.057013047 -0.03373092
sibsp      -0.02230662 -0.069089243 -0.12706675 -0.061215242 -0.06022107
parch      -0.01645101 -0.017696720 -0.24350184 -0.001165488 -0.09325342
fare        0.59116304 -0.125637353 -0.20177640  0.272087792 -0.12699238
pclass_1    1.00000000 -0.302550060 -0.11706927  0.285611052 -0.16373859
pclass_2   -0.30255006  1.000000000 -0.01057862 -0.143425262 -0.14105035
sex_M      -0.11706927 -0.010578622  1.00000000 -0.040151739 -0.09800681
embarked_C  0.28561105 -0.143425262 -0.04015174  1.000000000 -0.16362864
embarked_Q -0.16373859 -0.141050354 -0.09800681 -0.163628636  1.00000000
deck_A      0.28591131 -0.084644694  0.05358941  0.185442366 -0.05946853
deck_B      0.43224693 -0.122902623 -0.09558575  0.185963991 -0.08030780
deck_C      0.48474445 -0.077533075 -0.09364506  0.172425379 -0.07946766
deck_D      0.13502549  0.038571996 -0.05947357  0.269688905 -0.09069030
deck_E      0.05159522 -0.006826092 -0.06575988 -0.108208463 -0.08986794
deck_F     -0.70753043  0.136114513  0.15216823 -0.325659369  0.22529873
                deck_A      deck_B      deck_C       deck_D       deck_E
survived    0.03351255  0.16163293  0.16746961  0.098575075  0.322120129
age         0.11849174  0.11668084  0.15595995  0.049283037  0.151822887
sibsp      -0.06692064 -0.01805619  0.02670493 -0.005406569 -0.084456592
parch      -0.05778911  0.07389444 -0.05169019 -0.019083135 -0.036003896
fare        0.05775137  0.45996364  0.30933961  0.010115010 -0.019936462
pclass_1    0.28591131  0.43224693  0.48474445  0.135025488  0.051595219
pclass_2   -0.08464469 -0.12290262 -0.07753308  0.038571996 -0.006826092
sex_M       0.05358941 -0.09558575 -0.09364506 -0.059473565 -0.065759880
embarked_C  0.18544237  0.18596399  0.17242538  0.269688905 -0.108208463
embarked_Q -0.05946853 -0.08030780 -0.07946766 -0.090690296 -0.089867941
deck_A      1.00000000 -0.04730303 -0.06177894 -0.053418545 -0.066980420
deck_B     -0.04730303  1.00000000 -0.08342784 -0.072137753 -0.090452052
deck_C     -0.06177894 -0.08342784  1.00000000 -0.094213702 -0.118132634
deck_D     -0.05341854 -0.07213775 -0.09421370  1.000000000 -0.102146031
deck_E     -0.06698042 -0.09045205 -0.11813263 -0.102146031  1.000000000
deck_F     -0.23206097 -0.31338100 -0.40928339 -0.353896063 -0.443743031
                deck_F
survived   -0.47931430
age        -0.28802595
```

```
sibsp       0.06495373
parch       0.02814767
fare       -0.41755468
pclass_1   -0.70753043
pclass_2    0.13611451
sex_M       0.15216823
embarked_C -0.32565937
embarked_Q  0.22529873
deck_A     -0.23206097
deck_B     -0.31338100
deck_C     -0.40928339
deck_D     -0.35389606
deck_E     -0.44374303
deck_F      1.00000000
```

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

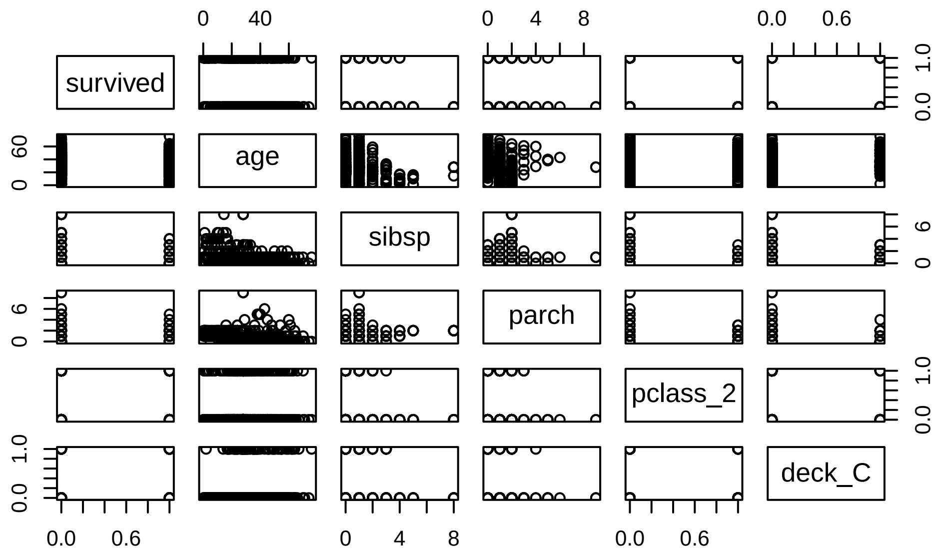


Figure 7: Scatter plots of all variables in train data

```
# 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
```

```
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.
```

```
Attaching package: 'rcompanion'


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

    phi
```

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

```
Cramer V
  0.5338
```

```r
library(corrplot)
```

```
corrplot 0.95 loaded
```

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

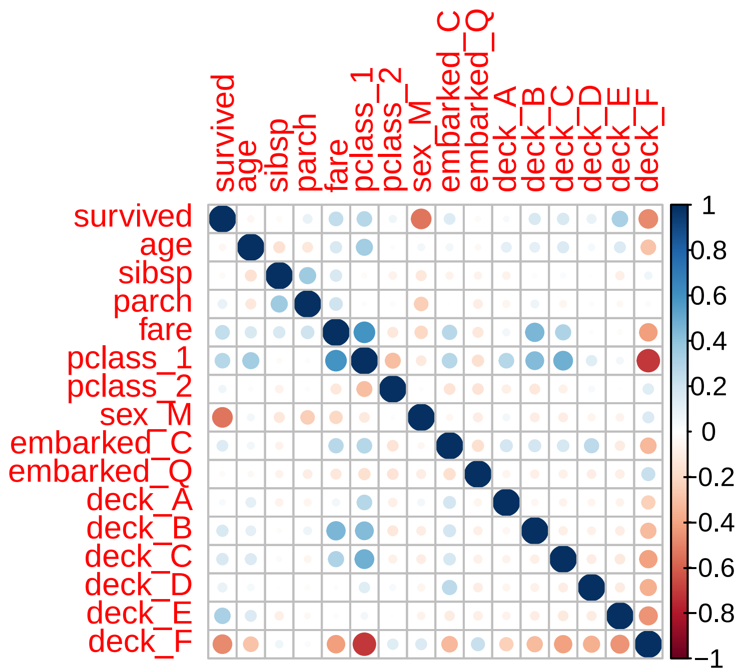


Figure 8: Correlation Matrix

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)
```

```
Loading required package: carData
```

```
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
  1.525578   1.241394   1.293316   1.718545   4.501868   1.540346   1.608259
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
  1.484871   1.397384   5.486232   5.989183   9.392249   8.038956   8.173562
    deck_F
 18.354762
```

```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'



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	0.8482533	0.8093278
base.model.test	0.8443878	0.7829181

```r
summary(lmod)
```

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

Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept)  5.412e+00  8.945e-01   6.050 1.45e-09 ***
age         -5.725e-02  9.037e-03  -6.335 2.37e-10 ***
sibsp       -2.434e-01  1.199e-01  -2.030  0.04232 *  
parch        2.348e-02  1.181e-01   0.199  0.84239    
fare         4.969e-05  2.301e-03   0.022  0.98277    
pclass_1     1.012e+00  4.589e-01   2.206  0.02741 *  
pclass_2     1.765e+00  2.973e-01   5.939 2.87e-09 ***
sex_M       -3.265e+00  2.557e-01 -12.768  < 2e-16 ***
embarked_C   7.810e-01  2.916e-01   2.678  0.00740 ** 
embarked_Q   8.999e-01  3.639e-01   2.473  0.01340 *  
deck_A      -2.136e+00  1.004e+00  -2.128  0.03338 *  
deck_B      -1.587e+00  1.013e+00  -1.567  0.11708    
deck_C      -1.874e+00  9.535e-01  -1.965  0.04940 *  
deck_D      -2.644e+00  9.259e-01  -2.856  0.00429 ** 
deck_E       2.530e-01  8.911e-01   0.284  0.77646    
deck_F      -4.351e+00  8.584e-01  -5.069 4.01e-07 ***
---
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:  634.33  on 900  degrees of freedom
AIC: 666.33

Number of Fisher Scoring iterations: 6
```

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

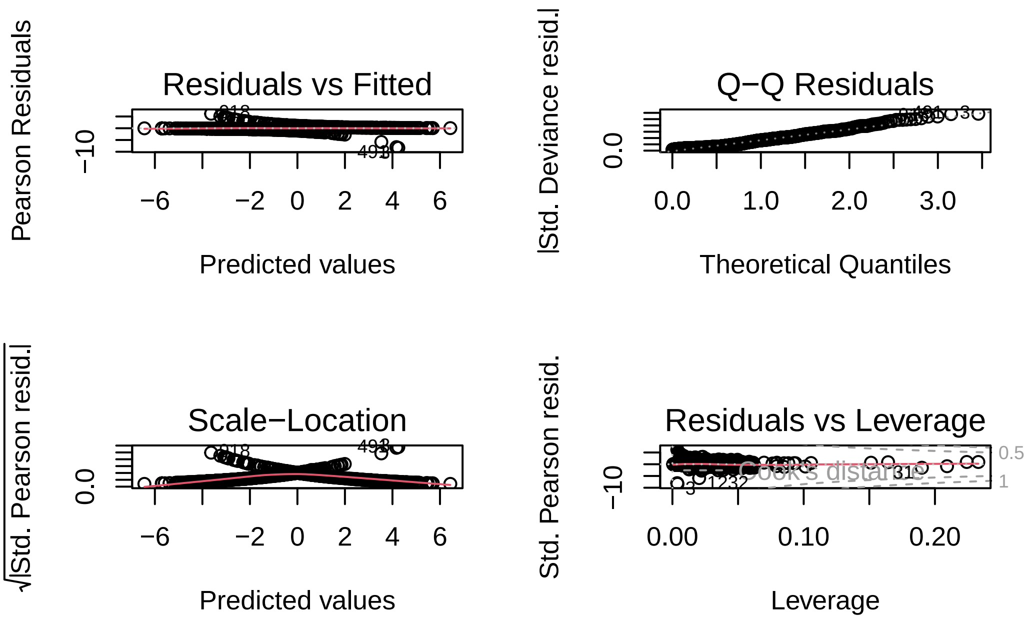


Figure 9: 4x4 standard plots for log. model

```
# 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.

```
################################################################################
#               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]

    # 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, ]
  }

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

```
################################################################################
#                     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 0.979474

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

Removing: parch with p-value 0.8042413

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

Removing: deck_E with p-value 0.7522959

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)
```

```

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 (+)          754.487     773.767     -92534510.256     0.44715     0.44594
 3      age (+)             721.049     745.149     -99967131.537     0.46813     0.46638
 4      deck_E (+)          694.782     723.702    -106319323.666     0.48429     0.48203
 5      pclass_2 (+)        672.910     706.651    -111995587.797     0.49756     0.49480
 6      sibsp (+)           667.029     705.589    -113925553.460     0.50186     0.49857
 7      deck_D (+)          661.184     704.564    -115879574.839     0.50611     0.50230
 8      embarked_Q (+)      659.681     707.881    -116754564.164     0.50799     0.50365
```

```
 9      embarked_Q (-)     661.184     704.564    -115879574.839    0.50611    0.50230
 10     embarked_C (+)     660.026     708.226    -116666703.385    0.50781    0.50347
 11     embarked_C (-)     661.184     704.564    -115879574.839    0.50611    0.50230
-------------------------------------------------------------------------------------

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

                          Model Summary
---------------------------------------------------------------
R                       0.711       RMSE                 0.344
R-Squared               0.506       MSE                  0.118
Adj. R-Squared          0.502       Coef. Var           87.122
Pred R-Squared          0.497       AIC                661.184
MAE                     0.260       SBC                704.564
---------------------------------------------------------------
 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    110.912          7         15.845    132.923    0.0000
Residual      108.235        908          0.119
Total         219.147        915
---------------------------------------------------------------------

                                  Parameter Estimates
-----------------------------------------------------------------------------------------
      model      Beta    Std. Error    Std. Beta         t        Sig      lower     upper
-----------------------------------------------------------------------------------------
(Intercept)     1.141         0.043                 26.357      0.000      1.056     1.226
      sex_M    -0.468         0.024       -0.460   -19.226      0.000     -0.516    -0.420
     deck_F    -0.452         0.031       -0.451   -14.567      0.000     -0.512    -0.391
        age    -0.007         0.001       -0.176    -7.114      0.000     -0.008    -0.005
     deck_E     0.163         0.042        0.106     3.852      0.000      0.080     0.247
   pclass_2     0.142         0.028        0.120     5.024      0.000      0.086     0.197
      sibsp    -0.031         0.011       -0.067    -2.811      0.005     -0.052    -0.009
     deck_D    -0.137         0.049       -0.074    -2.795      0.005     -0.233    -0.041
-----------------------------------------------------------------------------------------
```

```r
# Fit model with stepwise parms only:
binary.model.stepwise = glm(survived ~ sex_M + deck_F  + age + deck_E  +
                              pclass_2 + pclass_1 + sibsp + fare,
                            family = binomial, data = train)
summary(binary.model.stepwise)
```

```
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)  3.981403   0.438152   9.087  < 2e-16 ***
sex_M       -3.233885   0.242916 -13.313  < 2e-16 ***
deck_F      -2.538492   0.338044  -7.509 5.94e-14 ***
age         -0.056278   0.008733  -6.444 1.16e-10 ***
deck_E       1.832691   0.372320   4.922 8.55e-07 ***
pclass_2     1.353178   0.268462   5.040 4.64e-07 ***
pclass_1     0.625092   0.382656   1.634  0.10235
sibsp       -0.304005   0.112469  -2.703  0.00687 **
fare         0.002005   0.002091   0.959  0.33768
---
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:  654.47  on 907  degrees of freedom
AIC: 672.47

Number of Fisher Scoring iterations: 6
```

```r
# Removing fare from stepwise model since its insig.:
binary.model.stepwise = update(binary.model.stepwise, paste(". ~ . -", 'fare'))
summary(binary.model.stepwise)
```

```
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)  4.027774   0.436268   9.232  < 2e-16 ***
sex_M       -3.256877   0.241984 -13.459  < 2e-16 ***
deck_F      -2.559277   0.338208  -7.567 3.81e-14 ***
age         -0.056387   0.008721  -6.465 1.01e-10 ***
deck_E       1.813422   0.372128   4.873 1.10e-06 ***
pclass_2     1.379349   0.267722   5.152 2.58e-07 ***
pclass_1     0.757176   0.357985   2.115  0.03442 *
sibsp       -0.285271   0.110246  -2.588  0.00967 **
---
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:  655.42  on 908  degrees of freedom
```

```
AIC: 671.42

Number of Fisher Scoring iterations: 6
```

```r
################################################################################
#                 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
```

```r
# 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
  1.525578   1.241394   1.293316   1.718545   4.501868   1.540346   1.608259
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
  1.484871   1.397384   5.486232   5.989183   9.392249   8.038956   8.173562
    deck_F
 18.354762
```

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

```
       age      sibsp      parch       fare   pclass_1   pclass_2      sex_M
  1.514301   1.230447   1.283727   1.719904   4.547578   1.460059   1.463199
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
  1.478255   1.349790   2.030492   2.311424   2.853072   1.742811   1.584700
```

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

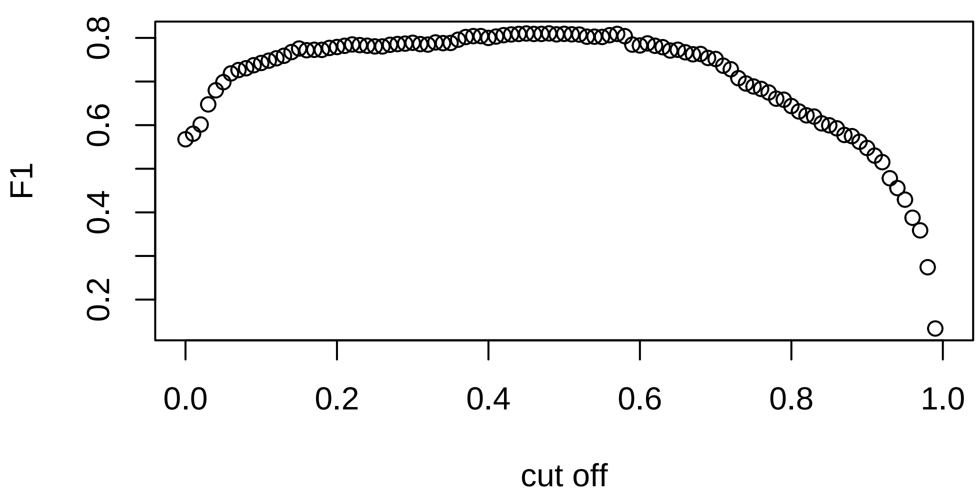
```r
################################################################################
#                          Function to calc. cutoff                            #
################################################################################
cutoff.prg<-function(pred,act){
# pred<-predicted_probabilities
# act<-true_labels
p<-seq(0,1,0.01)
n<-length(p)
```

```r
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
0.48
```

```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(
  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	0.8482533	0.8103683
base.model.test	0.8418367	0.7816901
stepwise.model.train	0.8165939	0.7711172
stepwise.model.test	0.8341837	0.7686833
filtered.model.train	0.8460699	0.8081633
filtered.model.test	0.8418367	0.7816901
vif.model.train	0.8329694	0.7906977
vif.model.test	0.8367347	0.7762238

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.

We decided to build a Poisson model for our challenger model. After building our original poisson model, we first check to see if the variables are important.

Ho: Variables are not important

Ha: Variable are important

Since we have a p-value of 0, we reject the null hypothesis. Variables are important.

Our poisson model has three significant variables, pclass_2, sex_M, and deck_F with an alpha of 0.05.

According to the poisson regression:

The odds of survival increases by 38.16% when the passenger is second class. ((exp(0.32326) - 1) * 100)

The odds of survival decreases by 68.37% when the passenger is male. ((exp(-1.15125) - 1) * 100)

The odds of survival decreases by 63.19% when the passenger is from deck G. ((exp(-0.99963) - 1) * 100)

Our Poisson Regression has an accuracy of 75.51% with optimal cutoff(.17) based on max F1 score (0.7272).

```r
library(MASS)

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

# compare the fitted model to the null model and calculate if the variables are important
summary(poissonReg_full)
```

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

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  7.653e-01  3.444e-01   2.222 0.026264 *
age         -1.776e-02  4.415e-03  -4.023 5.75e-05 ***
sibsp       -5.461e-02  6.961e-02  -0.784 0.432754
parch        2.236e-02  6.726e-02   0.332 0.739527
fare         1.401e-06  1.034e-03   0.001 0.998919
pclass_1     3.840e-01  2.002e-01   1.919 0.055024 .
pclass_2     5.574e-01  1.533e-01   3.637 0.000276 ***
sex_M       -1.096e+00  1.192e-01  -9.196  < 2e-16 ***
embarked_C   1.461e-01  1.388e-01   1.053 0.292401
embarked_Q   5.358e-01  2.189e-01   2.447 0.014392 *
deck_A      -5.240e-01  4.595e-01  -1.140 0.254158
deck_B      -6.356e-01  4.121e-01  -1.542 0.123013
deck_C      -6.960e-01  3.946e-01  -1.764 0.077789 .
deck_D      -7.640e-01  3.886e-01  -1.966 0.049310 *
deck_E      -1.605e-01  3.569e-01  -0.450 0.652902
deck_F      -1.554e+00  3.424e-01  -4.537 5.71e-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: 397.76  on 900  degrees of freedom
AIC: 1155.8

Number of Fisher Scoring iterations: 5
```

```r
1-pchisq(672.00-409.95, length(coef(poissonReg_full)) - 1)
```

```
[1] 0
```

```r
# We then reduce the model and see if the removed variables are significant
poissonReg_reduced <- glm(survived ~ pclass_2+sex_M+deck_F,family=poisson, train)

# Verify that all variables are significant and no more can be dropped
anova(poissonReg_reduced, poissonReg_full, test="Chi")
```

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
912	428.7449	NA	NA	NA
900	397.7597	12	30.98517	0.0019803

```r
drop1(poissonReg_reduced,test="Chi")
```

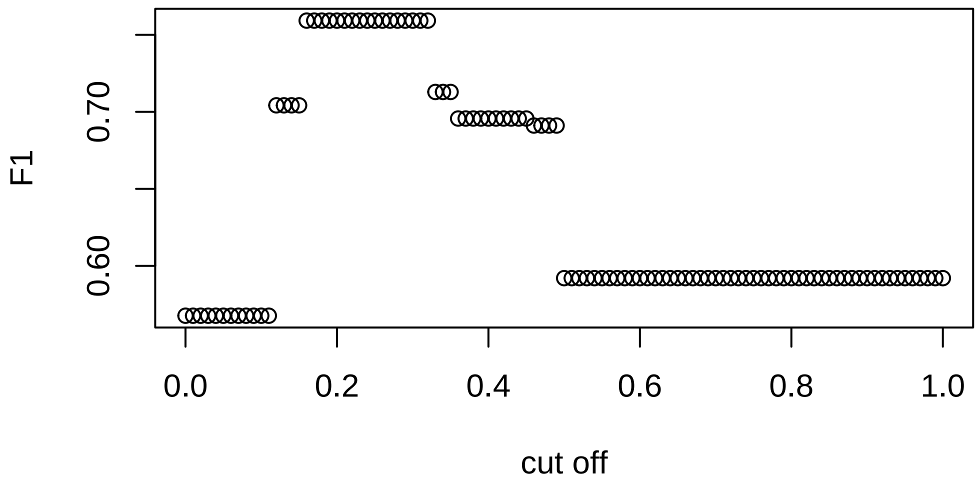
	Df	Deviance	AIC	LRT	Pr(>Chi)
	NA	428.7449	1162.745	NA	NA
pclass_2	1	435.2529	1167.253	6.508028	0.0107389
sex_M	1	538.8175	1270.818	110.072623	0.0000000
deck_F	1	520.5287	1252.529	91.783810	0.0000000

```r
### Train Data

# Testing against train data
predicted_probs <- predict(poissonReg_reduced, newdata = train, type = "response")

# Get the results of different cutoff values
trainResult<-cutoff.prg(predicted_probs,train$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.
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.
```

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




```
poisson.model.cutoff = trainResult[which.max(trainResult[,8]),]

# Get the F1 and Accuracy for train data using the optimal p value
observations.train <- train$survived
predictions.poisson.model.train <- ifelse(predicted_probs > poisson.model.cutoff[1], 1, 0)

confusion.matrix.binary.model.poisson.train <- confusionMatrix(as.factor(predictions.poisson.model.tr

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

poisson.model.train.summary = data.frame(
  Accuracy = poisson.model.accuracy.train,
  F1 = poisson.model.f1.train
)

row.names(poisson.model.train.summary) <- 'poisson.model.train'

### Test Data

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

```r
# Get the F1 and Accuracy for test data using the optimal p value
observations.test <- test$survived
predictions.poisson.model.test <- ifelse(predicted_probs > poisson.model.cutoff[1], 1, 0)

confusion.matrix.binary.model.poisson.test <- confusionMatrix(as.factor(predictions.poisson.model.tes

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

poisson.model.test.summary = data.frame(
  Accuracy = poisson.model.accuracy.test,
  F1 = poisson.model.f1.test
)

row.names(poisson.model.test.summary) <- 'poisson.model.test'

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

	Accuracy	F1
poisson.model.train	0.7652838	0.7592385
poisson.model.test	0.7551020	0.7257143

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)
)
```

```r
diagnostics_summary
```

Table 7: Summary of Diagnostic tests on Filtered vs Base models

Model	BP_Statistic	BP_pvalue	Shapiro_W	Shapiro_pvalue
Base.model	8.818434	0.0029820	0.9741629	0
Filtered.model	8.766123	0.0030688	0.9740530	0

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

Table 8: VIF Results for Base Logistic Regression Model

Variables	Tolerance	VIF
age	0.8014421	1.247751
sibsp	0.8139299	1.228607
parch	0.7767675	1.287386
fare	0.5106963	1.958111
pclass_1	0.2390773	4.182747
pclass_2	0.7882882	1.268572
sex_M	0.8536797	1.171399
embarked_C	0.7283295	1.373005
embarked_Q	0.8516863	1.174141
deck_A	0.2417476	4.136545
deck_B	0.1524257	6.560574
deck_C	0.1073871	9.312101

Variables	Tolerance	VIF
deck_D	0.1437156	6.958186
deck_E	0.1096865	9.116889
deck_F	0.0518910	19.271182

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

Table 9: VIF Results for Filtered Logistic Regression Model

Variables	Tolerance	VIF
age	0.8235167	1.214304
sibsp	0.9311335	1.073960
pclass_1	0.2691220	3.715787
pclass_2	0.7993736	1.250980
sex_M	0.9116354	1.096930
embarked_C	0.7494085	1.334386
embarked_Q	0.8625361	1.159372
deck_A	0.6445365	1.551503
deck_B	0.4997149	2.001141
deck_C	0.4310400	2.319970
deck_D	0.5876091	1.701812
deck_F	0.3665167	2.728389

```
# 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)
```

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

Table 10: Comparison of stats between base and filtered log models:

	RMSE	Rsquared	MAE
log.train.base	0.3895468	0.4678500	0.1517467
log.test.base	0.3976975	0.4341949	0.1581633
filtered.log.train.base	0.3923393	0.4621351	0.1539301
filtered.test.base	0.3976975	0.4341949	0.1581633

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

As we can see from the comparing all three models. The "vif.model" performed the base against the test data set. This is due to the removal high vif variables until our model reached a total VIF under 10 for each variable. Multicollinarity was a big factor that impacted our models accuracy and F1. Once we removed indications multicollinarity, we could see that our models performance increased over the base model.

However, with the "poisson.model", we see a decrease in accuracy compared the base model. This decrease is expected due to the nature of the poisson regression. The regression is most useful when the response variable is a count variable rather than a binary response. Thus a decrease in accuracy and F1 is expected against the base model.

As a conclusion, the model reduced via VIF is the best model to predict if a passenger were to survive, with an accuracy of 85.20% and improved over the base model by 0.25%.

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

Table 11: Comparison of our base model, best performing developed model, and challenger model

	Accuracy	F1
base.model.test	0.8418367	0.7816901
vif.model.test	0.8367347	0.7762238
poisson.model.test	0.7551020	0.7257143

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.

```
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                                 #
```

```r
################################################################################

# 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
```

```r
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,]
```

```r
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
```

```r
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
```

```r
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 
```

```r
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
log.train.ap1	0.3964912	0.4429840	0.1572052
log.test.ap1	0.4008919	0.4214242	0.1607143

```r
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
```

```r
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
```

```r
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
```

```r
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
```

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

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

```r
# 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
```

```r
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
```

```r
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
```

```r
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

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

```r
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/