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 **4**

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 . . . 5
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 **34**

10 Conclusion **34**

Appendix A: Check if 'sibsp' and 'parch' should be continuous or categorical **35**

References **49**

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 . . . 32
8 VIF Results for Base Logistic Regression Model . . . 32
9 VIF Results for Filtered Logistic Regression Model . . . 33
10 Comparison of stats between base and filtered log models: . . . 34
11 Comparison of our base model, best performing developed model, and challenger model . . . 35

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

We have tested five different models of which four were logistics regression based and one was a poisson model. The best model with regards to predictive accuracy was a “filtered” logistic regression (filtered because we removed insignificant predictors) with an accuracy of 84.18%.

The final model shows that the most significant predictors of survival are: age, sex, and deck_F.
Below we show the final calculated function:

```
survived ~ 5.74 - 0.05 age - 0.025 sibsp + 0.80 pclass_1 + 1.82 pclass_2 - 3.27 sex_M + 0.83 embarked
```

3 Introduction

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))
```

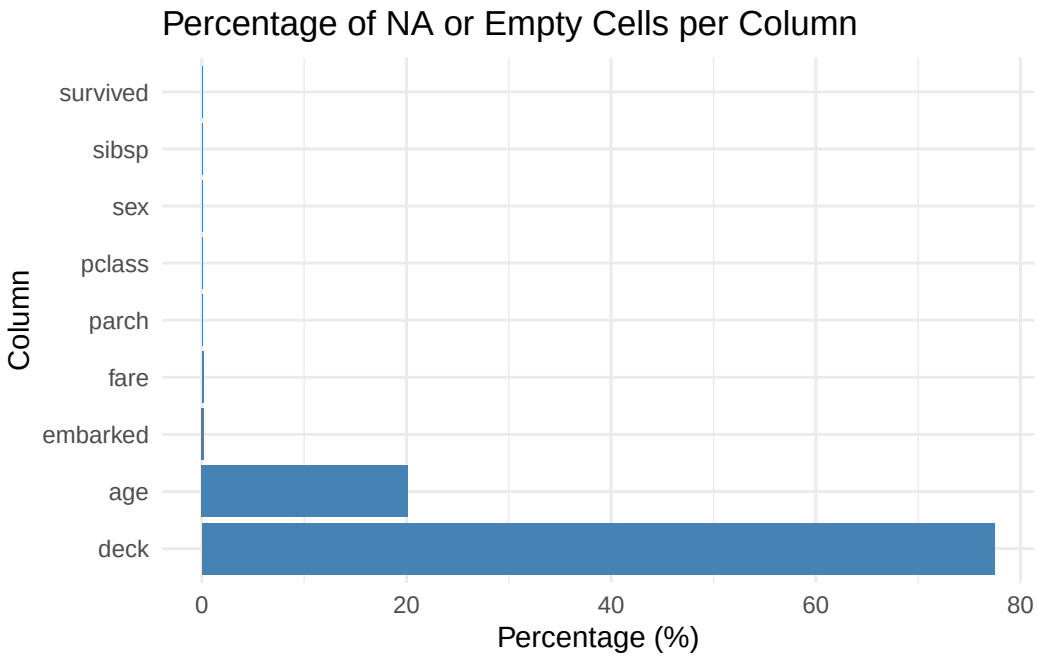


Figure 1: Percentage of Missing Values

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
set.seed (1023)
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
```

```
# 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
set.seed (1023)
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
```

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

plot_missing_barchart(data.clean)
```

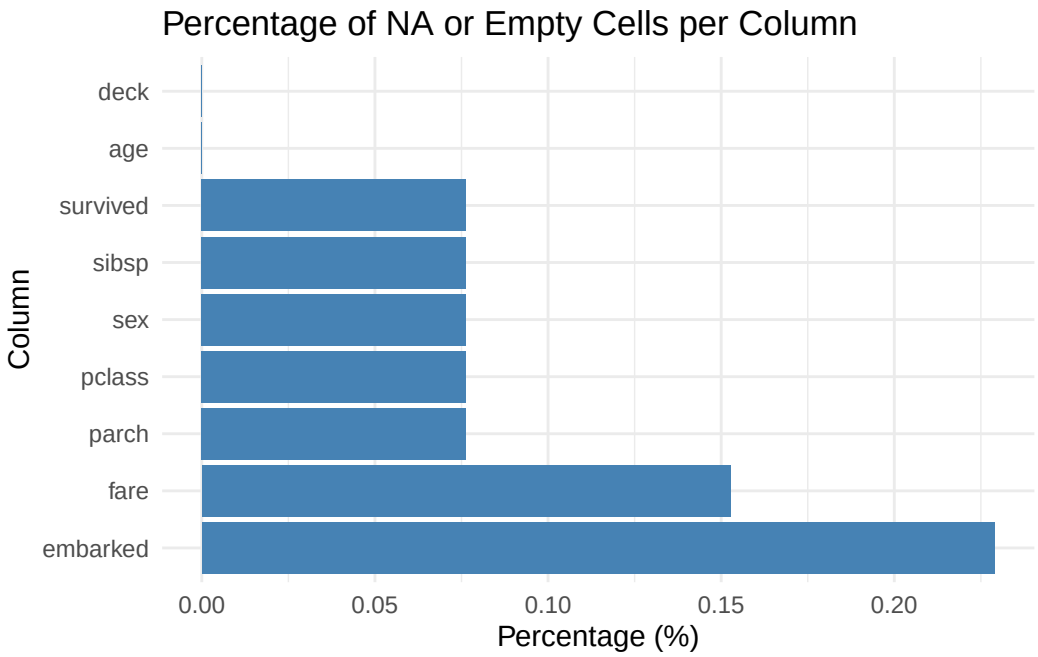


Figure 2: Percentage of Missing Values after Imputation

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")
```

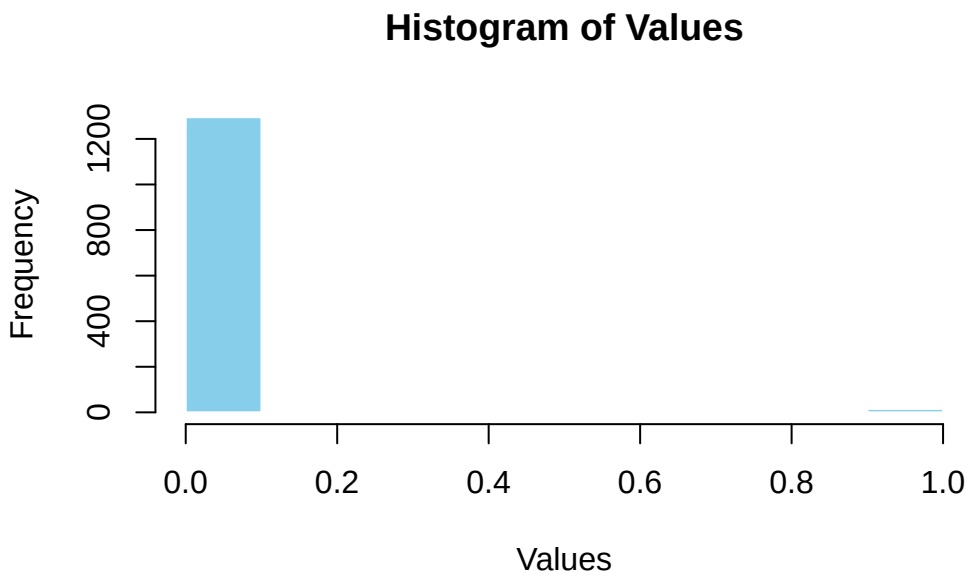


Figure 3: Histogram of Deck_G

```
# 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)
```

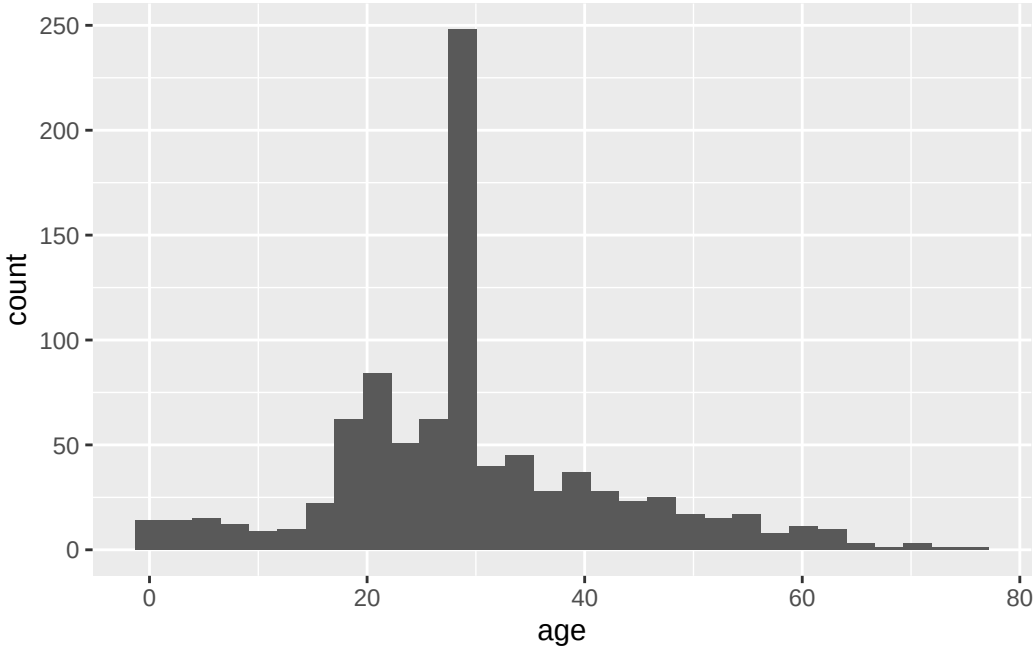


Figure 4: Histogram of survival vs age

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

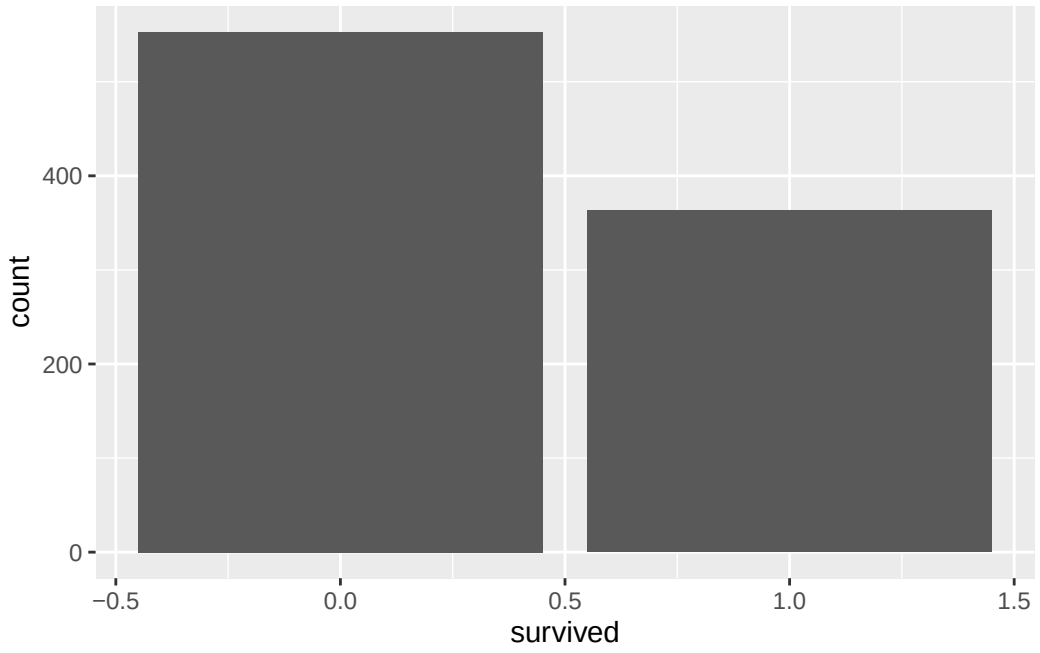


Figure 5: Barchart of survival

```
# 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')
```

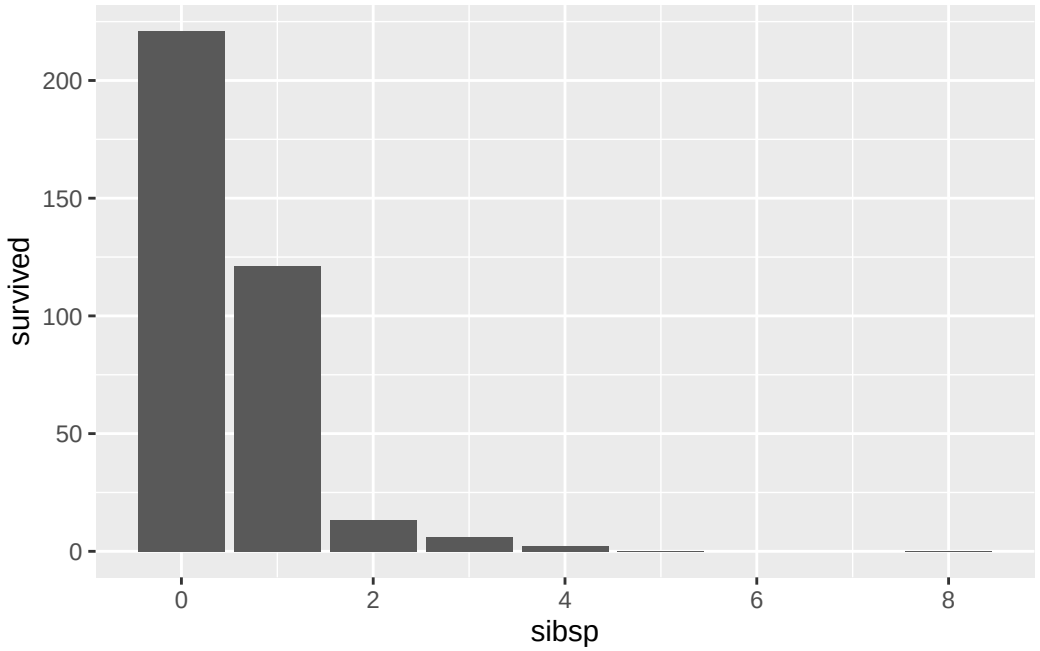


Figure 6: Barchart of survival vs Num. of siblings

```
cor(train)
```

```
survived        age       sibsp       parch        fare
```

survived	1.00000000	-0.04599216	-0.02754365	0.095720424	0.24962042
age	-0.04599216	1.00000000	-0.15495239	-0.122885529	0.16447722
sibsp	-0.02754365	-0.15495239	1.00000000	0.355216328	0.16529618
parch	0.09572042	-0.12288553	0.35521633	1.000000000	0.20987827
fare	0.24962042	0.16447722	0.16529618	0.209878267	1.00000000
pclass_1	0.28432063	0.34847627	-0.02230662	-0.016451014	0.59116304
pclass_2	0.06118909	0.01898825	-0.06908924	-0.017696720	-0.12563735
sex_M	-0.53377358	0.05385130	-0.12706675	-0.243501842	-0.20177640
embarked_C	0.14757622	0.05701305	-0.06121524	-0.001165488	0.27208779
embarked_Q	-0.02542950	-0.03373092	-0.06022107	-0.093253422	-0.12699238
deck_A	0.01921839	0.11573201	-0.06315491	-0.054051764	0.06359762
deck_B	0.15658451	0.12638063	-0.01583908	0.076255398	0.45645334
deck_C	0.17861407	0.15235605	0.02822826	-0.049048850	0.30882422
deck_D	0.08437944	0.10150497	-0.02726397	-0.029029703	0.01777467
deck_E	0.33473867	0.11691545	-0.07405912	-0.016332073	-0.01633414
deck_F	-0.47745405	-0.29217392	0.06499841	0.015200439	-0.42777519

	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	0.28790374	-0.08080438	0.05888172	0.167298858	-0.05745333
deck_B	0.42601897	-0.11327680	-0.09133967	0.182056642	-0.08108169
deck_C	0.48876162	-0.07908768	-0.09801461	0.169699800	-0.08023736
deck_D	0.15200710	0.04093807	-0.03323992	0.248144405	-0.07224766
deck_E	0.05822613	-0.02663532	-0.08900107	-0.078527161	-0.08770239
deck_F	-0.72064119	0.14341225	0.15235799	-0.319688792	0.21273851

	deck_A	deck_B	deck_C	deck_D	deck_E
survived	0.01921839	0.15658451	0.17861407	0.08437944	0.33473867
age	0.11573201	0.12638063	0.15235605	0.10150497	0.11691545
sibsp	-0.06315491	-0.01583908	0.02822826	-0.02726397	-0.07405912
parch	-0.05405176	0.07625540	-0.04904885	-0.02902970	-0.01633207
fare	0.06359762	0.45645334	0.30882422	0.01777467	-0.01633414
pclass_1	0.28790374	0.42601897	0.48876162	0.15200710	0.05822613
pclass_2	-0.08080438	-0.11327680	-0.07908768	0.04093807	-0.02663532
sex_M	0.05888172	-0.09133967	-0.09801461	-0.03323992	-0.08900107
embarked_C	0.16729886	0.18205664	0.16969980	0.24814440	-0.07852716
embarked_Q	-0.05745333	-0.08108169	-0.08023736	-0.07224766	-0.08770239
deck_A	1.00000000	-0.04614047	-0.06005247	-0.04955727	-0.06365303
deck_B	-0.04614047	1.00000000	-0.08474977	-0.06993828	-0.08983109
deck_C	-0.06005247	-0.08474977	1.00000000	-0.09102568	-0.11691645
deck_D	-0.04955727	-0.06993828	-0.09102568	1.00000000	-0.09648328
deck_E	-0.06365303	-0.08983109	-0.11691645	-0.09648328	1.00000000
deck_F	-0.22835159	-0.32226388	-0.41943104	-0.34612824	-0.44457878

	deck_F
survived	-0.47745405
age	-0.29217392

```
sibsp       0.06499841
parch       0.01520044
fare       -0.42777519
pclass_1   -0.72064119
pclass_2    0.14341225
sex_M       0.15235799
embarked_C -0.31968879
embarked_Q  0.21273851
deck_A     -0.22835159
deck_B     -0.32226388
deck_C     -0.41943104
deck_D     -0.34612824
deck_E     -0.44457878
deck_F      1.00000000
```

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

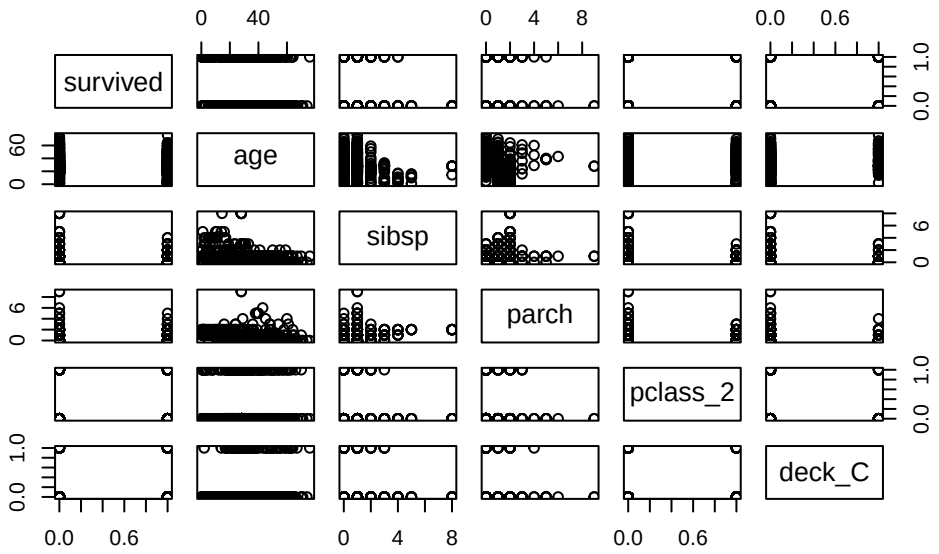


Figure 7: Scatter plots of all variables in train data

```r
# 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.

```r
#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
```

```
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
```

```
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
```

```
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
```

```
#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")
```

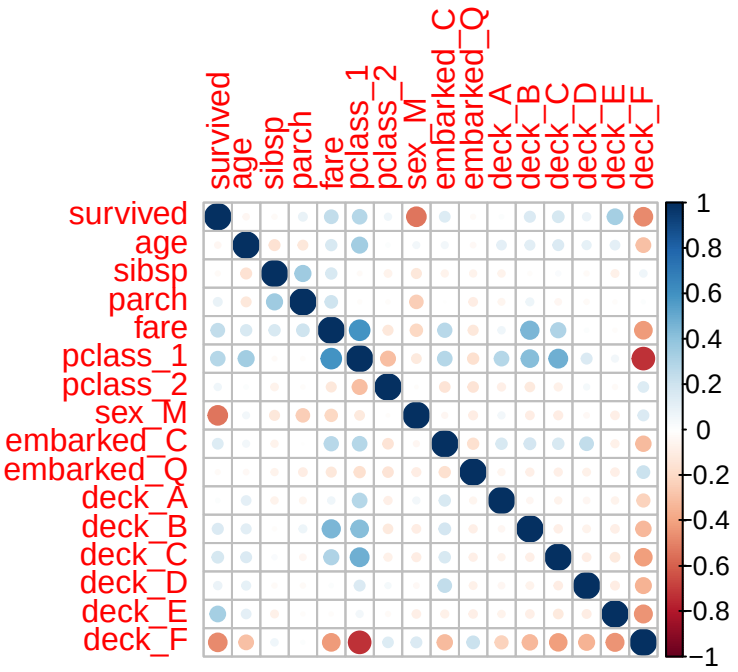


Figure 8: Correlation Matrix

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.

```
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
```

```
# Log model on train data:
set.seed (1023)
lmod <- glm(survived ~ ., family = binomial, data = train)
# summary(lmod)
vif(lmod)
```

```
      age      sibsp      parch       fare   pclass_1   pclass_2      sex_M
 1.487819   1.241500   1.283338   1.720620   4.661296   1.574259   1.619104
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
 1.433942   1.368562   5.122014   6.161202   9.446197   7.824253   7.280808
    deck_F
18.174852
```

```
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
)
```

```r
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	0.8504367	0.8115543
base.model.test	0.8392857	0.7758007

```r
summary(lmod)
```

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

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.386e+00  8.913e-01   6.042 1.52e-09 ***
age         -5.396e-02  9.051e-03  -5.962 2.49e-09 ***
sibsp       -2.538e-01  1.205e-01  -2.106  0.03520 *
parch        1.371e-02  1.185e-01   0.116  0.90790
fare        -6.976e-05  2.289e-03  -0.030  0.97568
pclass_1     7.953e-01  4.670e-01   1.703  0.08856 .
pclass_2     1.829e+00  3.011e-01   6.073 1.25e-09 ***
sex_M       -3.274e+00  2.566e-01 -12.756  < 2e-16 ***
embarked_C   8.299e-01  2.877e-01   2.884  0.00393 **
embarked_Q   7.944e-01  3.602e-01   2.205  0.02744 *
deck_A      -2.166e+00  1.011e+00  -2.143  0.03212 *
deck_B      -1.510e+00  1.006e+00  -1.501  0.13338
deck_C      -1.646e+00  9.553e-01  -1.723  0.08484 .
deck_D      -2.616e+00  9.310e-01  -2.810  0.00495 **
deck_E       4.274e-01  8.975e-01   0.476  0.63393
deck_F      -4.342e+00  8.572e-01  -5.065 4.08e-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:  634.78  on 900  degrees of freedom
AIC: 666.78

Number of Fisher Scoring iterations: 6
```

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

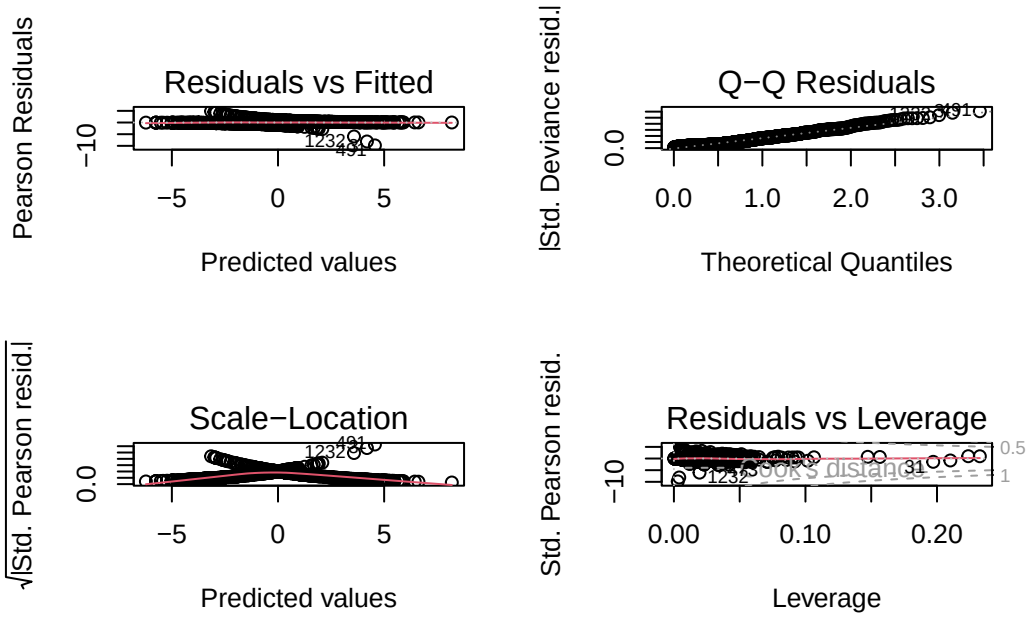


Figure 9: 4x4 standard plots for log. model

```
# 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 (lmod)

2) Model with insig. pred. removed (filtered.model)

3) Stepwise model (step.model)

4) Model with high vif pred. removed (vif.model)

When comparing the accuracy and F1 score on all models the filtered model was the highest performer and we decided to use that as the champion model until now. We suspect that multicollinearity among the decks has an impact on the model performance which is why removing these collinear predictors and filtering for insignificant variables improved the model slightly against the based model (+0.3%).

```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]

    # 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)
}
```

```r
##############################################################################
#                      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 0.9710719
```

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

```
Removing: parch with p-value 0.894389
```

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

```
Removing: deck_E with p-value 0.5871368
```

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

```
[1] "all variable are signifcant"
```

```
################################################################################
#                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
```

```
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 (+)         757.110     776.390     -92005909.772    0.44556    0.44435
 3      age (+)            723.096     747.196     -99521155.302    0.46694    0.46518
 4      pclass_2 (+)       697.456     726.376    -105700242.392    0.48278    0.48051
 5      deck_E (+)         674.236     707.976    -111671761.113    0.49683    0.49406
 6      sibsp (+)          668.302     706.862    -113609066.331    0.50117    0.49788
 7      deck_D (+)         664.319     707.699    -115088607.776    0.50442    0.50059
 8      embarked_Q (+)     663.468     711.668    -115792422.465    0.50596    0.50160
 9      embarked_Q (-)     664.319     707.699    -115088607.776    0.50442    0.50059
-------------------------------------------------------------------------------------

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

                         Model Summary
-------------------------------------------------------------
R                       0.710       RMSE                0.344
R-Squared               0.504       MSE                 0.119
Adj. R-Squared          0.501       Coef. Var          87.272
Pred R-Squared          0.495       AIC                664.319
MAE                     0.262       SBC                707.699
-------------------------------------------------------------
 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    110.541          7         15.792    132.026    0.0000
Residual      108.606        908          0.120
Total         219.147        915
```

```
-----------------------------------------------------------------

                                Parameter Estimates
-------------------------------------------------------------------------------------------
      model      Beta    Std. Error    Std. Beta       t        Sig      lower     upper
-------------------------------------------------------------------------------------------
(Intercept)     1.131         0.043                 26.050    0.000     1.046     1.216
      sex_M    -0.464         0.024       -0.457    -19.025    0.000    -0.512    -0.416
     deck_F    -0.447         0.031       -0.445    -14.444    0.000    -0.508    -0.387
        age    -0.006         0.001       -0.171     -6.871    0.000    -0.008    -0.005
   pclass_2     0.147         0.028        0.124      5.193    0.000     0.091     0.202
     deck_E     0.169         0.043        0.108      3.942    0.000     0.085     0.253
      sibsp    -0.031         0.011       -0.068     -2.848    0.005    -0.052    -0.010
     deck_D    -0.123         0.050       -0.064     -2.439    0.015    -0.222    -0.024
-------------------------------------------------------------------------------------------
```

```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)  4.039203   0.452351   8.929  < 2e-16 ***
sex_M       -3.223591   0.242196 -13.310  < 2e-16 ***
deck_F      -2.635131   0.348620  -7.559 4.07e-14 ***
age         -0.053691   0.008709  -6.165 7.05e-10 ***
deck_E       1.898805   0.387471   4.901 9.56e-07 ***
pclass_2     1.407566   0.270017   5.213 1.86e-07 ***
pclass_1     0.453560   0.388459   1.168  0.24297
sibsp       -0.305368   0.112220  -2.721  0.00651 **
fare         0.001986   0.002090   0.951  0.34181
---
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:  654.59  on 907  degrees of freedom
AIC: 672.59

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)  4.089766   0.450105   9.086  < 2e-16 ***
sex_M       -3.246187   0.241244 -13.456  < 2e-16 ***
deck_F      -2.659167   0.348576  -7.629 2.37e-14 ***
age         -0.053845   0.008699  -6.190 6.02e-10 ***
deck_E       1.874294   0.386905   4.844 1.27e-06 ***
pclass_2     1.432280   0.269447   5.316 1.06e-07 ***
pclass_1     0.582248   0.364952   1.595  0.11062
sibsp       -0.286517   0.109911  -2.607  0.00914 **
---
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:  655.53  on 908  degrees of freedom
AIC: 671.53

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
  1.487819   1.241500   1.283338   1.720620   4.661296   1.574259   1.619104
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
  1.433942   1.368562   5.122014   6.161202   9.446197   7.824253   7.280808
```

```
   deck_F
 18.174852
```

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

```
       age      sibsp      parch       fare   pclass_1   pclass_2      sex_M
  1.477874   1.230404   1.274009   1.720786   4.741756   1.481905   1.468964
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
  1.420600   1.325908   1.986762   2.357338   3.004607   1.736194   1.557128
```

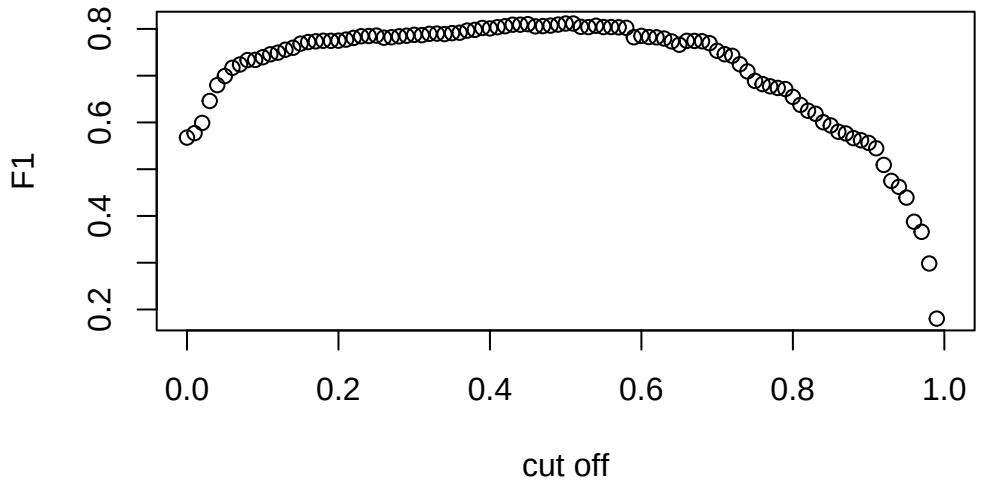
```
# Removing deck_F from the model eliminates the multicollinearity completely.
```

```
################################################################################
#                          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
}
```

```
# 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.
```

```
plot(test_cutoff[,1],test_cutoff[,8],xlab="cut off",ylab="F1")
```




```
optimal.cutoff = test_cutoff[which.max(test_cutoff[,8]),]
optimal.cutoff[1]
```

```
  p
0.5
```

```
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	0.8504367	0.8115543
base.model.test	0.8392857	0.7758007
stepwise.model.train	0.8231441	0.7692308
stepwise.model.test	0.8367347	0.7611940
filtered.model.train	0.8504367	0.8115543
filtered.model.test	0.8418367	0.7801418
vif.model.train	0.8351528	0.7911480
vif.model.test	0.8392857	0.7789474

7 Challenger Models

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 37.6% when the passenger is second class. ((exp(0.31918) - 1) * 100)

The odds of survival decreases by 68.14% when the passenger is male. ((exp(-1.144) - 1) * 100)

The odds of survival decreases by 64.76% when the passenger is from deck G. ((exp(-1.043) - 1) * 100)

Our Poisson Regression has an accuracy of 74.74% with optimal cutoff(.17) based on max F1 score (0.71954).

```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)  7.787e-01  3.449e-01   2.258 0.023946 *
age         -1.609e-02  4.355e-03  -3.695 0.000220 ***
sibsp       -6.954e-02  6.979e-02  -0.996 0.319019
parch       -5.055e-04  6.756e-02  -0.007 0.994030
fare         9.289e-05  1.025e-03   0.091 0.927778
pclass_1     3.590e-01  2.047e-01   1.753 0.079536 .
pclass_2     5.102e-01  1.532e-01   3.329 0.000871 ***
sex_M       -1.095e+00  1.201e-01  -9.112  < 2e-16 ***
embarked_C   1.072e-01  1.349e-01   0.795 0.426660
embarked_Q   4.537e-01  2.162e-01   2.098 0.035872 *
deck_A      -6.157e-01  4.732e-01  -1.301 0.193282
deck_B      -6.494e-01  4.130e-01  -1.572 0.115868
deck_C      -6.891e-01  3.943e-01  -1.748 0.080527 .
deck_D      -7.195e-01  3.962e-01  -1.816 0.069347 .
deck_E      -2.300e-01  3.556e-01  -0.647 0.517786
deck_F      -1.527e+00  3.419e-01  -4.467 7.94e-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: 404.15  on 900  degrees of freedom
AIC: 1162.2
```

```
Number of Fisher Scoring iterations: 5
```

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

```
[1] 0
```

```
# 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	430.4971	NA	NA	NA
900	404.1538	12	26.34334	0.0095947

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

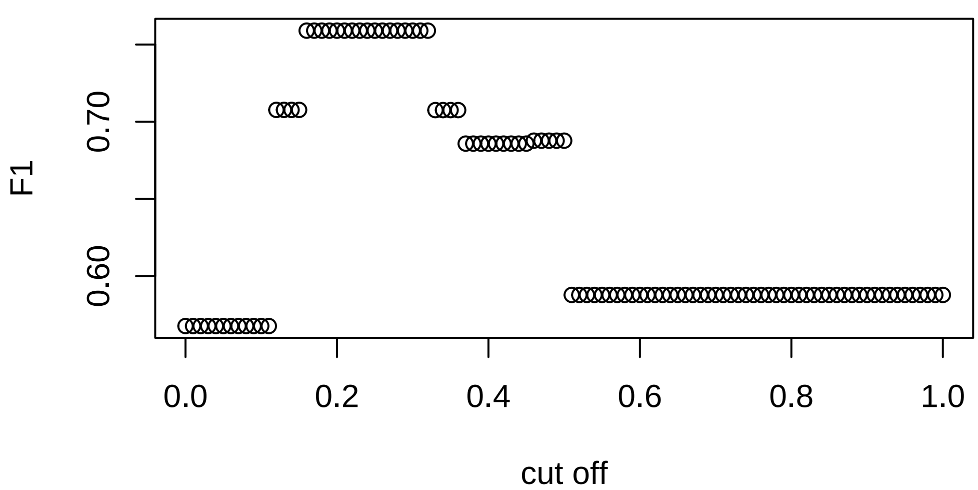
	Df	Deviance	AIC	LRT	Pr(>Chi)
	NA	430.4971	1164.497	NA	NA
pclass_2	1	436.9472	1168.947	6.450073	0.0110948
sex_M	1	540.1126	1272.113	109.615481	0.0000000
deck_F	1	520.5287	1252.529	90.031585	0.0000000

```
### 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.
```

```r
# 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")
```




```r
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
```

```r
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")

# 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	0.7663755	0.7590090
poisson.model.test	0.7474490	0.7195467

8 Model Limitation and Assumptions

The champion model selected is the filtered logistic regression model due to its superior test accuracy (84.1%) and F1 score (0.78014). 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. The base model is used as the benchmark due to its similar performance and broader feature set.

```r
summary(binary.model.filtered)
```

```
Call:
glm(formula = survived ~ age + sibsp + pclass_1 + pclass_2 +
    sex_M + embarked_C + embarked_Q + deck_A + deck_B + deck_C +
    deck_D + deck_F, family = binomial, data = train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.745434   0.507728  11.316  < 2e-16 ***
age         -0.053296   0.008919  -5.976 2.29e-09 ***
sibsp       -0.254771   0.116064  -2.195  0.02816 *
pclass_1     0.799195   0.449070   1.780  0.07513 .
pclass_2     1.827594   0.297584   6.141 8.18e-10 ***
sex_M       -3.278525   0.251332 -13.045  < 2e-16 ***
embarked_C   0.830036   0.282806   2.935  0.00334 **
embarked_Q   0.785344   0.354744   2.214  0.02684 *
deck_A      -2.549579   0.616634  -4.135 3.55e-05 ***
deck_B      -1.896873   0.585234  -3.241  0.00119 **
deck_C      -2.032446   0.516558  -3.935 8.33e-05 ***
deck_D      -2.995951   0.494319  -6.061 1.35e-09 ***
deck_F      -4.706551   0.404816 -11.626  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1230.1  on 915  degrees of freedom
Residual deviance:  635.0  on 903  degrees of freedom
AIC: 661

Number of Fisher Scoring iterations: 6
```

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 over-itting 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.

```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	8.234753	0.0041096	0.9727993	0
Filtered.model	8.213280	0.0041585	0.9726405	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	0.8045607	1.242914
sibsp	0.8152516	1.226615
parch	0.7785122	1.284501
fare	0.5132230	1.948471
pclass_1	0.2345039	4.264322
pclass_2	0.7848041	1.274203
sex_M	0.8534444	1.171723
embarked_C	0.7524221	1.329041
embarked_Q	0.8549107	1.169713
deck_A	0.2520772	3.967038
deck_B	0.1505798	6.640995
deck_C	0.1062098	9.415324
deck_D	0.1509330	6.625458
deck_E	0.1122820	8.906149

Variables	Tolerance	VIF
deck_F	0.0522602	19.135037

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

Table 9: VIF Results for Filtered Logistic Regression Model

Variables	Tolerance	VIF
age	0.8258163	1.210923
sibsp	0.9322261	1.072701
pclass_1	0.2639511	3.788581
pclass_2	0.7953422	1.257321
sex_M	0.9103965	1.098423
embarked_C	0.7737213	1.292455
embarked_Q	0.8651816	1.155827
deck_A	0.6538958	1.529295
deck_B	0.5054336	1.978499
deck_C	0.4299243	2.325991
deck_D	0.6025945	1.659491
deck_F	0.3615519	2.765855

```
# 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 10: Comparison of stats between base and filtered log models:

	RMSE	Rsquared	MAE
log.train.base	0.3867342	0.4727588	0.1495633
log.test.base	0.4008919	0.4240145	0.1607143
filtered.log.train.base	0.3867342	0.4727588	0.1495633
filtered.test.base	0.3976975	0.4322221	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 Accuracy and F1 values in order to determine that the model is working correctly. The first level we are monitoring is Accuracy or F1 values falling below 70%. Once one of those values falls below 70% we would examine the test data to make sure the data is valid. Assuming the data is valid, we will keep a closer eye on the model performance moving forward. Once both Accuracy and F1 values fall below 70%, or one of the values falls below 60%, we will retrain the model on more recent data.

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 data relevant to ship travel today. Our expectation is that age is one data point that will look drastically different for ship travel today than it did for the Titanic simply due to the massive increase in life expectancy (verywellhealth, 2024). Due to this change, we will make sure to closely monitor age in new data and will adjust the model if we see the median age change by 20%.

Another concern of ours is that there will be concept drift. It is possible that over time, the relationships between variables in the model will change, resulting in the model losing effectiveness. In order to account for this, we will continue to monitor correlations between the model variables and will adjust the model if we notice any large (greater than 20%) changes in the correlations.

Overall, we will monitor our model to make sure it remains robust. Our assumptions are that a robust model will maintain both Accuracy and F1 values of over 70%, while the correlations in the model remain fairly constant and new data doesn't appear to vary drastically from our training data. We will look for the model to keep Accuracy and F1 values above 70%, while also making sure that median age of new data doesn't change by more than 20% and that correlations between variables in the model don't change by more than 20%. If any of these events were to happen, we would look to retrain our model on more recent data in order to maintain model robustness.

10 Conclusion

As we can see from the comparing all three models. The "filtered.model" performed the best against the test data set. This is due to the removal insignificant variables until our model remained with only sigificant predictors. Multicollinarity seemed to be 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 p-values filtering is the best model to predict if a passenger were to survive, with an accuracy of 84.28% and improved over the base model by 0.25%.

```
data.frame(rbind(base.model.test.summary,
                 filtered.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	0.8392857	0.7758007
filtered.model.test	0.8418367	0.7801418
poisson.model.test	0.7474490	0.7195467

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)
```

```r
# 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)
```

```r
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,]
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
-1.28431 -0.18993 -0.02983  0.19714  0.97447

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.1771844  0.1208120   9.744  < 2e-16 ***
age         -0.0062915  0.0010464  -6.013 2.66e-09 ***
sibsp       -0.0387862  0.0222879  -1.740  0.08216 .
parch       -0.0300588  0.0271365  -1.108  0.26830
fare         0.0003968  0.0002994   1.325  0.18550
pclass_1     0.0811930  0.0556170   1.460  0.14468
pclass_2     0.1305011  0.0315138   4.141 3.79e-05 ***
sex_M       -0.4220045  0.0253002 -16.680  < 2e-16 ***
embarked_C   0.0477656  0.0322484   1.481  0.13891
embarked_Q   0.1103169  0.0427122   2.583  0.00996 **
deck_A      -0.3194255  0.1468546  -2.175  0.02988 *
deck_B      -0.2768199  0.1381986  -2.003  0.04547 *
deck_C      -0.2682689  0.1346614  -1.992  0.04666 *
deck_D      -0.3270119  0.1314397  -2.488  0.01303 *
deck_E       0.0136466  0.1251546   0.109  0.91320
```

```
deck_F      -0.5835976  0.1189681  -4.905 1.11e-06 ***
sibsp_1      0.0752979  0.0356515   2.112  0.03496 *
sibsp_2      0.1073260  0.0801711   1.339  0.18101
sibsp_3     -0.1027125  0.1199623  -0.856  0.39211
sibsp_4     -0.2242068  0.1301364  -1.723  0.08526 .
sibsp_5     -0.2237646  0.2026863  -1.104  0.26989
parch_1      0.1456157  0.0456425   3.190  0.00147 **
parch_2      0.1363899  0.0718664   1.898  0.05804 .
parch_3      0.2094703  0.1744398   1.201  0.23014
parch_4     -0.0890979  0.2045944  -0.435  0.66332
parch_5      0.0600587  0.2394630   0.251  0.80202
parch_6      0.0037453  0.2906192   0.013  0.98972
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3413 on 889 degrees of freedom
Multiple R-squared:  0.5237,    Adjusted R-squared:  0.5098
F-statistic: 37.59 on 26 and 889 DF,  p-value: < 2.2e-16
```

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

```
      age      sibsp      parch       fare   pclass_1   pclass_2      sex_M
 1.461878   3.469370   4.492844   1.920452   4.690456   1.313421   1.156946
embarked_C embarked_Q     deck_A     deck_B     deck_C     deck_D     deck_E
 1.420534   1.169222   5.198873   8.907075  13.377776   9.336210  11.767550
    deck_F    sibsp_1    sibsp_2    sibsp_3    sibsp_4    sibsp_5    parch_1
26.496638   1.868491   1.445820   1.463025   1.863122   1.404482   1.917338
   parch_2    parch_3    parch_4    parch_5    parch_6
 3.089996   1.298950   1.431051   1.471912   1.446896
```

```
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)  5.266e+00  1.399e+00   3.765 0.000167 ***
age         -5.266e-02  9.841e-03  -5.351 8.76e-08 ***
sibsp       -1.942e+00  1.285e+02  -0.015 0.987938
parch       -1.671e+00  1.615e+02  -0.010 0.991745
fare         7.583e-04  2.406e-03   0.315 0.752647
pclass_1     8.262e-01  4.862e-01   1.699 0.089253 .
pclass_2     1.344e+00  3.064e-01   4.385 1.16e-05 ***
sex_M       -3.044e+00  2.476e-01 -12.296  < 2e-16 ***
embarked_C   6.262e-01  2.821e-01   2.220 0.026451 *
embarked_Q   1.147e+00  3.682e-01   3.116 0.001834 **
deck_A      -2.578e+00  1.522e+00  -1.694 0.090192 .
deck_B      -2.209e+00  1.496e+00  -1.477 0.139625
```

```
deck_C      -2.266e+00  1.468e+00  -1.544 0.122706
deck_D      -2.772e+00  1.439e+00  -1.926 0.054095 .
deck_E       3.489e-01  1.422e+00   0.245 0.806183
deck_F      -4.675e+00  1.374e+00  -3.403 0.000665 ***
sibsp_1      2.202e+00  1.285e+02   0.017 0.986323
sibsp_2      4.142e+00  2.570e+02   0.016 0.987141
sibsp_3      4.166e+00  3.854e+02   0.011 0.991377
sibsp_4      4.649e+00  5.139e+02   0.009 0.992783
sibsp_5     -6.681e+00  1.233e+03  -0.005 0.995677
parch_1      2.640e+00  1.615e+02   0.016 0.986959
parch_2      4.102e+00  3.230e+02   0.013 0.989869
parch_3      5.957e+00  4.845e+02   0.012 0.990190
parch_4      6.123e+00  6.460e+02   0.009 0.992438
parch_5      8.503e+00  8.076e+02   0.011 0.991599
parch_6     -4.236e+00  1.760e+03  -0.002 0.998080
---
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:  609.92  on 889  degrees of freedom
AIC: 663.92

Number of Fisher Scoring iterations: 15
```

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

```
         age        sibsp        parch         fare     pclass_1     pclass_2
1.644869e+00 7.753684e+05 1.618468e+06 1.763642e+00 5.096422e+00 1.529833e+00
       sex_M   embarked_C   embarked_Q       deck_A       deck_B       deck_C
1.439671e+00 1.505300e+00 1.370293e+00 1.035046e+01 1.551825e+01 2.526478e+01
      deck_D       deck_E       deck_F      sibsp_1      sibsp_2      sibsp_3
1.949011e+01 1.511758e+01 4.497328e+01 3.293441e+05 1.876474e+05 2.068774e+05
     sibsp_4      sibsp_5      parch_1      parch_2      parch_3      parch_4
1.995922e+05 1.372545e+00 3.184507e+05 6.909815e+05 2.223700e+05 2.249046e+05
     parch_5      parch_6
4.152894e+05 1.434758e+00
```

```r
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	0.3881430	0.4633288	0.1506550
linear.test.ap1	0.4040610	0.4221825	0.1632653
log.train.ap1	0.3951121	0.4460886	0.1561135
log.test.ap1	0.3912304	0.4439052	0.1530612

```
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
         0 499  76
         1  62 279

               Accuracy : 0.8493
                 95% CI : (0.8245, 0.8719)
    No Information Rate : 0.6124
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6803

 Mcnemar's Test P-Value : 0.2685

              Precision : 0.8182
                 Recall : 0.7859
                     F1 : 0.8017
             Prevalence : 0.3876
         Detection Rate : 0.3046
   Detection Prevalence : 0.3723
      Balanced Accuracy : 0.8377

       '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
         0 215  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
```

```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
         0 523 105
         1  38 250

               Accuracy : 0.8439
                 95% CI : (0.8187, 0.8668)
    No Information Rate : 0.6124
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6593

 Mcnemar's Test P-Value : 3.406e-08

              Precision : 0.8681
                 Recall : 0.7042
                     F1 : 0.7776
             Prevalence : 0.3876
         Detection Rate : 0.2729
   Detection Prevalence : 0.3144
```

```
      Balanced Accuracy : 0.8182

       '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
         0 227  40
         1  20 105

               Accuracy : 0.8469
                 95% CI : (0.8074, 0.8811)
    No Information Rate : 0.6301
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.662

 Mcnemar's Test P-Value : 0.01417

              Precision : 0.8400
                 Recall : 0.7241
                     F1 : 0.7778
             Prevalence : 0.3699
         Detection Rate : 0.2679
   Detection Prevalence : 0.3189
      Balanced Accuracy : 0.8216

       '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----
```

```r
#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)

# 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
```

```r
# 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
```

```r
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/

verywellhealth. (2024). *How has average life expectancy changed from the 1800s to today?* https://www.verywellhealth.com/longevity-throughout-history-2224054