Titanic - Ananlyzing Survival

Christoph Beck

2025-09-17

Executive summary

This analysis showcases the end-to-end pipeline of taking peoples characteristics and infering their likelihood of being involved in a binary outcome. It is directly tranferable to problems like customer churn prediction, sales predictions, and general inference into human decision making processes.

Introduction

The problem is simple: We get two data sets of titanic passengers, train and test. The training data-set contains information about the passengers death or survival, while our task is to estimate that exact outcome for passengers in the test-set.

We get a dataframe which further contains the following variables: PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked.

ParCh stands for "number of parents or children", SibSp stands for "number of siblings or spouses, disregarding fiancés".

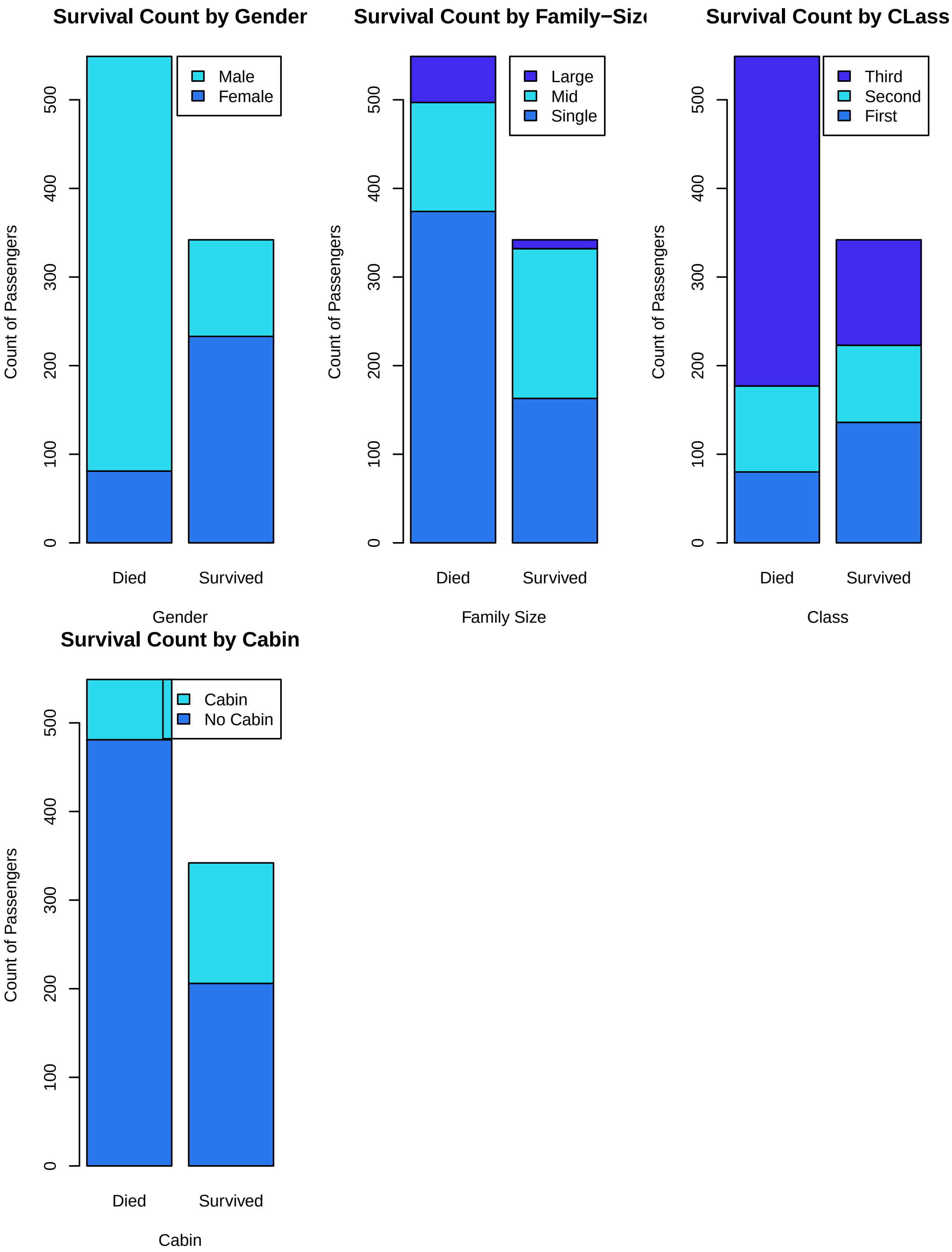
"Embarked" refers to the ports of Southampton, Charlestown, and Queens.

Things to note are, that the amount of passengers with missing ages and cabin data is substantial. This means that measures need to be taken for effective use of them.

EDA

First things first, we'll take a closer look at our data:

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.1.0
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```



Gender clearly had a severe impact. Family-size was similar, with mid-size families (2-4 people) being the “best” class. Passenger class was beneficial if you were in first class.

In order to look at the impact of age on survival, we'll need to address the issue of missing age values. The easiest way to do so is to filter people by their titles. Via the tidyverse plugin we can do exactly this:

```r
library(tidyverse)
name = titanic$Name
title = str_extract(name, "\\b[A-Za-z]+\\.")

# dataframing ####

base.data = data.frame(survived, age, gender, family.cat, pclass, title, has.cabin)

# age imputation ####

young.woman = as.numeric( base.data$title == "Lady." |
                            base.data$title == "Miss." |
                            base.data$title == "Mlle." |
                            base.data$title == "Ms." )

older.woman = as.numeric( base.data$title == "Mme."|
                            base.data$title == "Mrs.")
young.man =   as.numeric(base.data$title == "Master.")

old.man =     as.numeric(base.data$title == "Mr.")

specialist =  as.numeric(base.data$title == "Capt."|
                           base.data$title == "Col."|
                           base.data$title == "Dr."|
                           base.data$title == "Major."|
                           base.data$title == "Rev.")

noble =       as.numeric(base.data$title == "Countess."|
                           base.data$title == "Count."| # same logic as 'Dona.'
                           base.data$title == "Don."|
                           base.data$title == "Jonkheer."|
                           base.data$title == "Sir."|
                           base.data$title == "Dame.")

other = as.numeric (!(noble | specialist |
                        old.man | young.man |
                        older.woman | young.woman))

title.class = data.frame(titanic$Age, young.woman,
                         older.woman, young.man,
                         old.man, specialist,
                         noble, other )

median.by.group = aggregate(age ~ young.woman +
                              older.woman + young.man +
                              old.man + specialist +
                              noble + other,
                            data = title.class,
                            FUN = median,
                            na.rm = T)

group.names = c("Young Woman", "Older Woman",
                "Young Man", "Old Man", "Specialist",
                "Noble", "Other")
median.age = c(median.by.group$age, sum(other))
print(median.age)
```

```
## [1] 21.5 35.0  3.5 30.0 50.0 39.0  0.0
```

```
median.age = data.frame(group.names , median.age)
print(median.age)
```

```
##   group.names median.age
## 1 Young Woman       21.5
## 2 Older Woman       35.0
## 3   Young Man        3.5
## 4     Old Man       30.0
## 5  Specialist       50.0
## 6       Noble       39.0
## 7       Other        0.0
```

```
# proper medians have been distilled and labeled. Moving on to NA replacement

median.age.by.group = median.age$median.age

base.data$age[is.na(base.data$age) & title.class$young.woman == 1] = median.age.by.group[1]
base.data$age[is.na(base.data$age) & title.class$older.woman == 1] = median.age.by.group[2]
base.data$age[is.na(base.data$age) & title.class$young.man == 1] = median.age.by.group[3]
base.data$age[is.na(base.data$age) & title.class$old.man == 1] = median.age.by.group[4]
base.data$age[is.na(base.data$age) & title.class$specialist == 1] = median.age.by.group[5]
base.data$age[is.na(base.data$age)
              & title.class$noble == 1] = median.age.by.group[6]
base.data$age[is.na(base.data$age) &
                title.class$other == 1] = median.age.by.group[7]
```

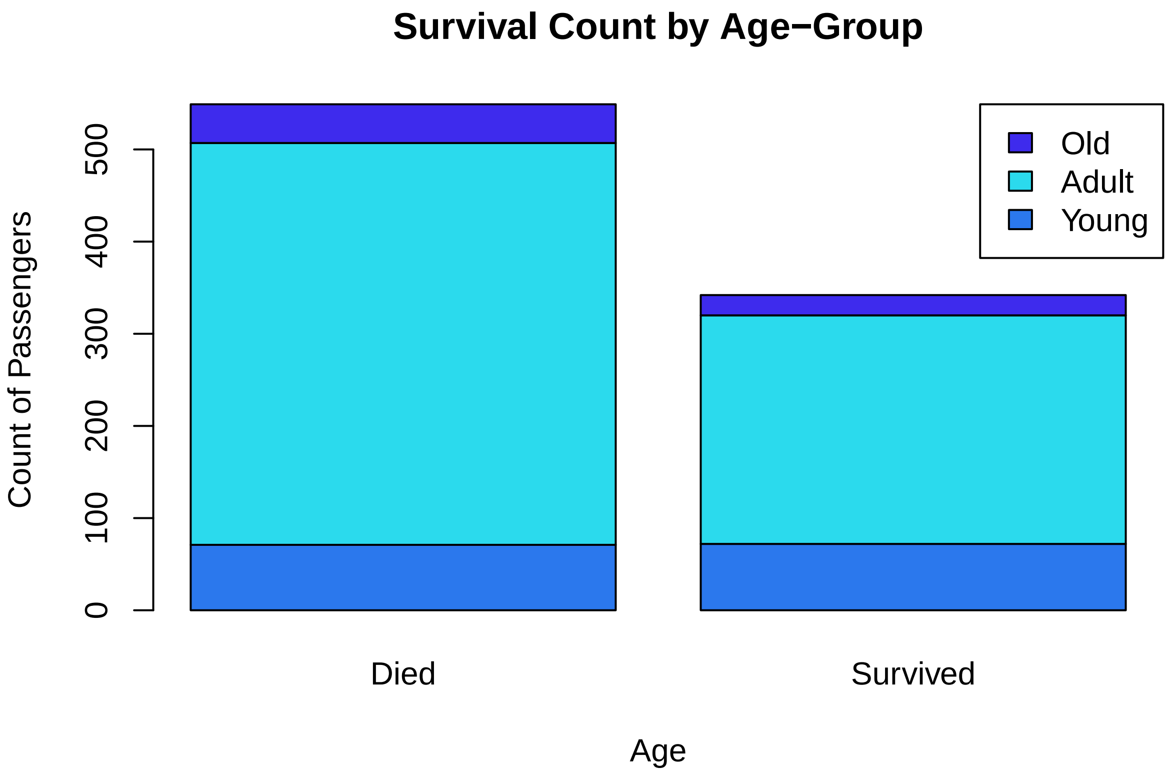
Now we'll take a look at different age groups:

```
age.class = cut(base.data$age, breaks = c(0, 18, 50, 100))

age_survival_table <- table(age.class, titanic$Survived)
colnames(age_survival_table) <- c("Died", "Survived")
rownames(age_survival_table) <- c("Young", "Adult", "Old")
print(age_survival_table)
```

```
##
## age.class Died Survived
##     Young   71       72
##     Adult  436      248
##     Old     42       22
```

```
barplot(age_survival_table,
        main = "Survival Count by Age-Group",
        xlab = "Age",
        ylab = "Count of Passengers",
        col = c("#2B78ED", "#2BDAED", "#3E2BED"),
        legend.text = c("Young", "Adult", "Old"),
        args.legend = list(x = "topright"))
```



We could expand the model by looking at cabins, ports, geographic data, etc., but we're going to stick to these variables for our model.

We'll also actually get a number for the strength of the correlation of the corresponding classes with survival:

```
cat.test = function(x, y) {
  # contingency table
  table.xy = table( x, y)

  # chisquare test
  chi.test = chisq.test(table.xy)

  # Cramér's V
  n = sum(table.xy)
  k = min(dim(table.xy))
  V = sqrt (chi.test$statistic / (n * (k - 1)) )

  # Return the results
  list(
    table = table.xy,
    chi.squared = chi.test,
    cramers.v = V,
    std.residuals = chi.test$stdres
  )
}
```

```r
family.v = cat.test(survived, family.cat)$cramers.v
pclass.v = cat.test(survived, pclass)$cramers.v
gender.v = cat.test(survived, gender)$cramers.v
age.v = cat.test(survived, age.class)$cramers.v
```

we get effect sizes of 28.9% for family-size, 34% for pclass, 54.1 for gender, and 10.8% for age.

Creating an interaction based model

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
survival.est = glm(formula = survived ~ age + gender + family.cat +
socio.class + has.cabin + age:gender +
age:family.cat + age:socio.class,
family = "binomial", data = base.data)
probs = predict(object = survival.est, type = "response")

roc.curve = roc(base.data$survived, probs)
```

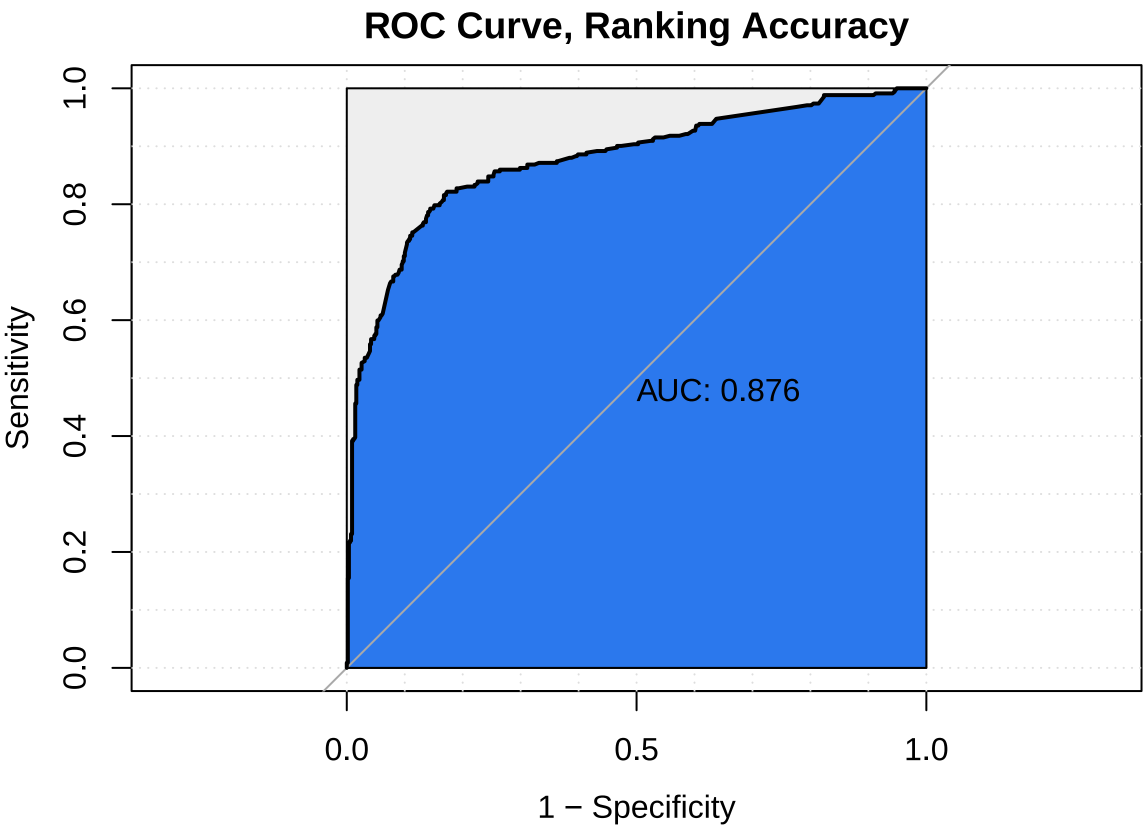
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
survival.roc = roc(base.data$survived, probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(survival.roc,
     main = "ROC Curve, Ranking Accuracy",
     print.auc = T,
     auc.polygon = T,
     auc.polygon.col = "#2B78ED",
     max.auc.polygon = T,
     grid = T,
     legacy.axes = T)
```



The idea is to perform a logistic regression which uses our engineered features, but also considers interactions in-between them.

Our current model can get a maximum AUC of 0.876.
This means it can correctly differentiate between survival and non-survival cases ~88% of the time.

We can now use this model to start making predictions about passenger survival likelihood, but we're missing a last critical step.

Youdens J statistic

in order to maximize the difference between true positive (accurately guessing survival) and false positive (inaccurately guessing survival) rate, we're calculating the point on the ROC-curve, which is farthest away from the line of randomness (the diagonal).

```
best.cutoff <- coords(survival.roc, "best", ret = "threshold", best.method = "youden")
```

In theory, by crossing a threshold of 36.3% in estimated likelihood, it's most likely for the passenger to have survived.

We could now transform new passenger data in analogous ways, and make binary forecasts by combining likelihoods and cutoffs.

Generating Test Predictions

Having built and validated our model on the training data, we now apply it to the unseen test dataset to generate survival predictions for submission.

```r
# Titanic survivor analysis

rm(list = ls())
library(readr)
titanic <- read_csv("~/Downloads/titanic/test.csv")
load("~/Documents/R_Projects/titanic_median.RData") # loading median estimates,
load("~/Documents/titanic_model.RData") # and model.
load("~/Documents/titanic_cutoff.RData")


# assigning variables ####

age = titanic$Age
id = titanic$PassengerId
parch = titanic$Parch
sibsp = titanic$SibSp
sex = titanic$Sex
fare = titanic$Fare
pclass = titanic$Pclass


# data cleaning ####

gender = (sex == "female") * 1 # converting to binary variable, female = 1
gender = as.factor(gender)
socio.class = as.factor(pclass) # factorization
has.cabin = as.factor(as.numeric(! is.na(titanic$Cabin)))

# analyzing family size's impact, creating a size variable

family.size = parch + sibsp + 1 # +1 for self
family.cat = cut(family.size,
                 breaks = c(0, 1, 4, Inf),
                 labels = c("alone", "small (2-4)", "large (5+)"))

library(tidyverse) # string extraction with tidyverse

name = titanic$Name
title = str_extract(name, "\\b[A-Za-z]+\\.")

# dataframing ####

base.data = data.frame(age, gender, family.cat, pclass, title, has.cabin)
# age imputation ####

young.woman = as.numeric( base.data$title == "Lady." |
                            base.data$title == "Miss." |
                            base.data$title == "Mlle." |
                            base.data$title == "Ms." )

older.woman = as.numeric( base.data$title == "Mme."|
                            base.data$title == "Mrs.")
young.man =   as.numeric(base.data$title == "Master.")
```

```r
old.man =     as.numeric(base.data$title == "Mr.")

specialist =  as.numeric(base.data$title == "Capt."|
                           base.data$title == "Col."|
                           base.data$title == "Dr."|
                           base.data$title == "Major."|
                           base.data$title == "Rev.")

noble =       as.numeric(base.data$title == "Countess."|
                           base.data$title == "Count."| # same logic as 'Dona.'
                           base.data$title == "Don."|
                           base.data$title == "Jonkheer."|
                           base.data$title == "Sir."|
                           base.data$title == "Dame.")

other = as.numeric (!(noble | specialist | old.man | young.man | older.woman | young.woman))

title.class = data.frame(titanic$Age, young.woman, older.woman, young.man, old.man, specialist, noble,

# proper medians have been distilled and labeled. Moving on to NA replacement


base.data$age[is.na(base.data$age) &
                title.class$young.woman == 1] = median.age.by.group[1]
base.data$age[is.na(base.data$age) &
                title.class$older.woman == 1] = median.age.by.group[2]
base.data$age[is.na(base.data$age) &
                title.class$young.man == 1] = median.age.by.group[3]
base.data$age[is.na(base.data$age) &
                title.class$old.man == 1] = median.age.by.group[4]
base.data$age[is.na(base.data$age) &
                title.class$specialist == 1] = median.age.by.group[5]
base.data$age[is.na(base.data$age) &
                title.class$noble == 1] = median.age.by.group[6]
base.data$age[is.na(base.data$age) &
                title.class$other == 1] = median.age.by.group[7]

# new ages successfully imputed.

# running the model on the test data:
probs = predict(survival.est, newdata = base.data, type = "response")

best.cutoff = as.numeric(best.cutoff)
predicted.class = ifelse(probs >= best.cutoff, 1, 0)
PassengerId = id
Survived = predicted.class

submission = data.frame(PassengerId, Survived)
write_csv(submission, file = "~/Documents/submission.csv")


# Output a summary of the predictions
cat("Submission file 'submission.csv' has been created.\n")
```

```
## Submission file 'submission.csv' has been created.
```

```r
cat("Total passengers in test set:", nrow(titanic), "\n")
```

```
## Total passengers in test set: 418
```

```r
cat("Number predicted to survive:", sum(submission$Survived), "\n")
```

```
## Number predicted to survive: 193
```

```r
cat("Number predicted to perish:", sum(submission$Survived == 0), "\n")
```

```
## Number predicted to perish: 225
```

```r
cat("Predicted survival rate:", round(mean(submission$Survived) * 100, 2), "%\n")
```

```
## Predicted survival rate: 46.17 %
```

The preprocessing and prediction pipeline has been successfully applied to the test data. The survival predictions for all passengers have been saved to **'submission.csv'**, which is formatted for submission to the Kaggle competition. The model predicts a survival rate of approximately 46.17% on the unseen test data.

Later testing revealed that adjusting for class imbalance and changing the cutoff from 32% to 60% significantly improved predictive accuracy. This is essentially us saying "We're pessimistic about survival".

This seems to work in our favour and definitely needs more future exploration on my side.

Evaluation

This project demonstrates an effective, end-to-end pipeline for solving binary classification problems. The methodology—spanning thoughtful data exploration, robust feature engineering, and predictive modeling—is directly transferable to high-impact business applications.

The model successfully identifies key drivers of survival and achieves a strong accuracy of ~77.8% on unseen data, significantly outperforming naive benchmarks, for example "everyone dies", which score about 62%. This proves its utility for making data-driven predictions.

Future work could explore ensemble methods like Random Forests, Neural Networks, or Gradient Boosting to capture more complex patterns, potentially boosting performance further. Ultimately, this framework provides a powerful foundation for tackling critical business problems such as customer churn prediction, lead scoring, and fraud detection, enabling smarter decision-making and improved operational efficiency.