

Titanic Survival Analysis: An R-Based Approach

This document details the process of creating a predictive model for the Kaggle competition on Titanic survival, using the R programming language.

1. Environment Setup

To begin, it is necessary to install and load the R libraries that will be used throughout the project.

1.1. Package Installation

The main libraries for this analysis are **tidyverse** for data manipulation and visualization; **randomForest** for modeling; and **caret** to facilitate machine learning tasks.

They can be installed with the following command in the R console:

```
install.packages("tidyverse")  
install.packages("randomForest")  
install.packages("caret")
```

```
> install.packages("tidyverse")
```

1.2. Library Loading

Once installed, we load the packages into our R session to use their functions:

```
library(tidyverse)  
library(randomForest)  
library(caret)
```

```
> library(tidyverse)
```

2. Data Loading and Initial Exploration

2.1. Setting the Working Directory

The next step is to download the data from Kaggle and save it in our working directory. First, we check our current directory with the `getwd()` command:

```
> getwd()  
[1] "C:/Users/nadiv/Documents"  
> |
```

2.2. Downloading the Data

The data can be found on the Kaggle Titanic competition page. We will download the dataset, which includes the `train.csv` and `test.csv` files.

<https://www.kaggle.com/competitions/titanic/data>

Data Explorer


93.08 kB

 gender_submission.csv


 test.csv

 train.csv

Summary

▸  3 files

▸  25 columns

 **Download All**

2.3. Loading Data into R

Once the .csv files are in our working directory, we load them into R as two separate dataframes: train_data for model training and test_data for prediction.

```
> train_data <- read.csv("train.csv")
> test_data <- read.csv("test.csv") )
```

2.4. Initial Visualization

We can get a first look at the structure and a summary of the training data with the str() and summary() commands:

```
> str(train_data)
'data.frame': 891 obs. of 12 variables:
 $ PassengerId: int 1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : int 0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass     : int 3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : chr "Braund, Mr. Owen Harris" "Cumings, Mrs. John Br
 adley (Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, M
 rs. Jacques Heath (Lily May Peel)" ...
 $ Sex        : chr "male" "female" "female" "female" ...
 $ Age        : num 22 38 26 35 35 NA 54 2 27 14 ...
 $ Sibsp      : int 1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int 0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : chr "A/5 21171" "PC 17599" "STON/O2. 3101282" "11380
 3" ...
 $ Fare       : num 7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : chr "" "C85" "" "C123" ...
 $ Embarked   : chr "S" "C" "S" "S" ...

> summary(train_data)
 PassengerId      Survived      Pclass
 Min.   : 1.0      Min.   :0.0000   Min.   :1.000
 1st Qu.:223.5     1st Qu.:0.0000   1st Qu.:2.000
 Median :446.0     Median :0.0000   Median :3.000
 Mean   :446.0     Mean   :0.3838   Mean   :2.309
 3rd Qu.:668.5     3rd Qu.:1.0000   3rd Qu.:3.000
 Max.   :891.0     Max.   :1.0000   Max.   :3.000

      Name                Sex                Age
 Length:891      Length:891      Min.   : 0.42
 Class :character Class :character 1st Qu.:20.12
 Mode  :character Mode  :character Median :28.00
                                     Mean  :29.70
                                     3rd Qu.:38.00
                                     Max.  :80.00
                                     NA's  :177

      Sibsp      Parch      Ticket
 Min.   :0.000   Min.   :0.0000   Length:891
 1st Qu.:0.000   1st Qu.:0.0000   Class :character
 Median :0.000   Median :0.0000   Mode  :character
 Mean   :0.523   Mean   :0.3816
 3rd Qu.:1.000   3rd Qu.:0.0000
 Max.   :8.000   Max.   :6.0000

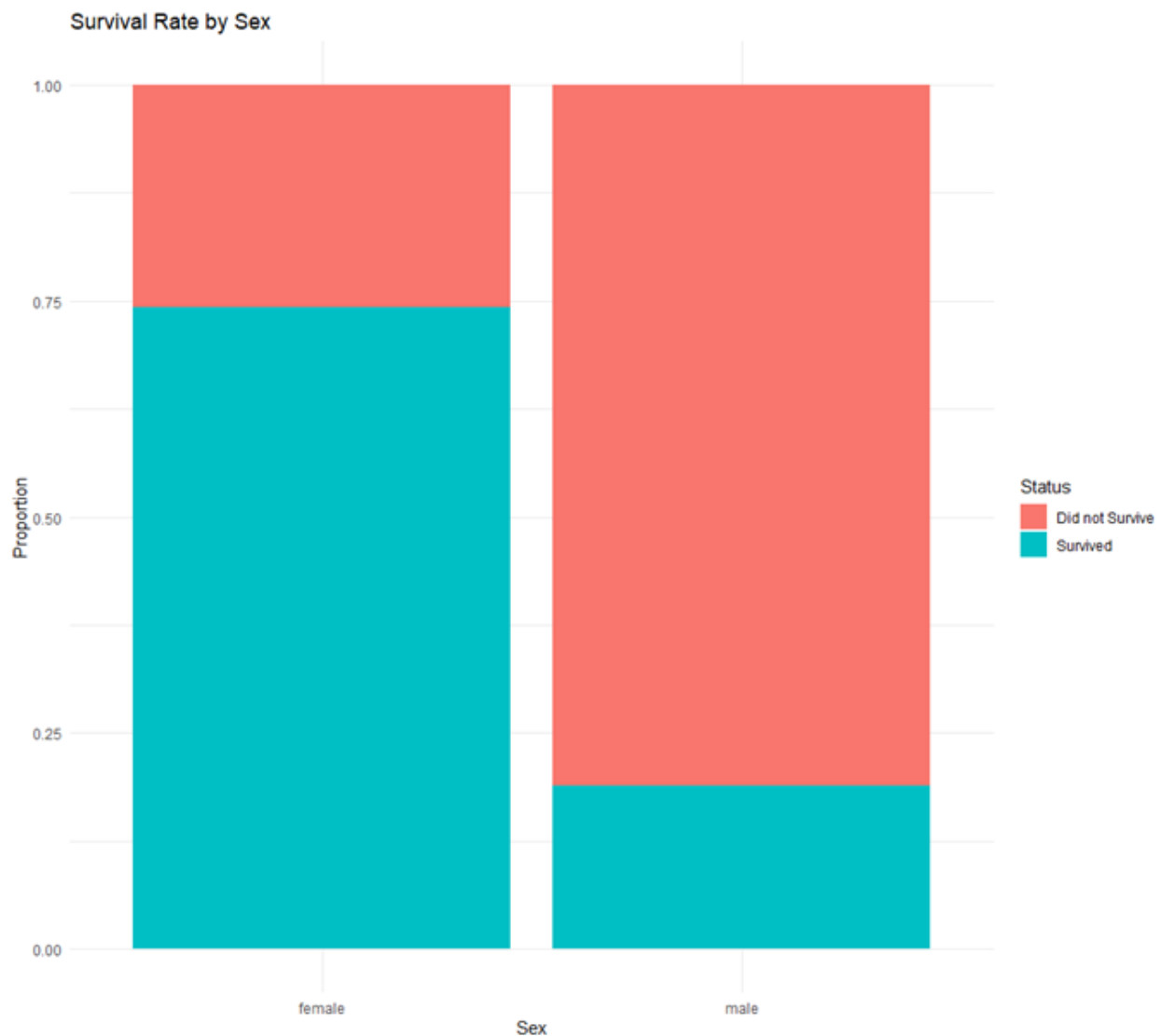
      Fare      Cabin      Embarked
 Min.   : 0.00   Length:891   Length:891
 1st Qu.: 7.91   Class :character   Class :character
 Median :14.45   Mode  :character   Mode  :character
 Mean   :32.20
 3rd Qu.:31.00
 Max.   :512.33
```

3. Exploratory Data Analysis (EDA)

In this phase, we explore the data to find patterns and insights that help us better understand the problem.

Insight 1: Did more women survive than men?

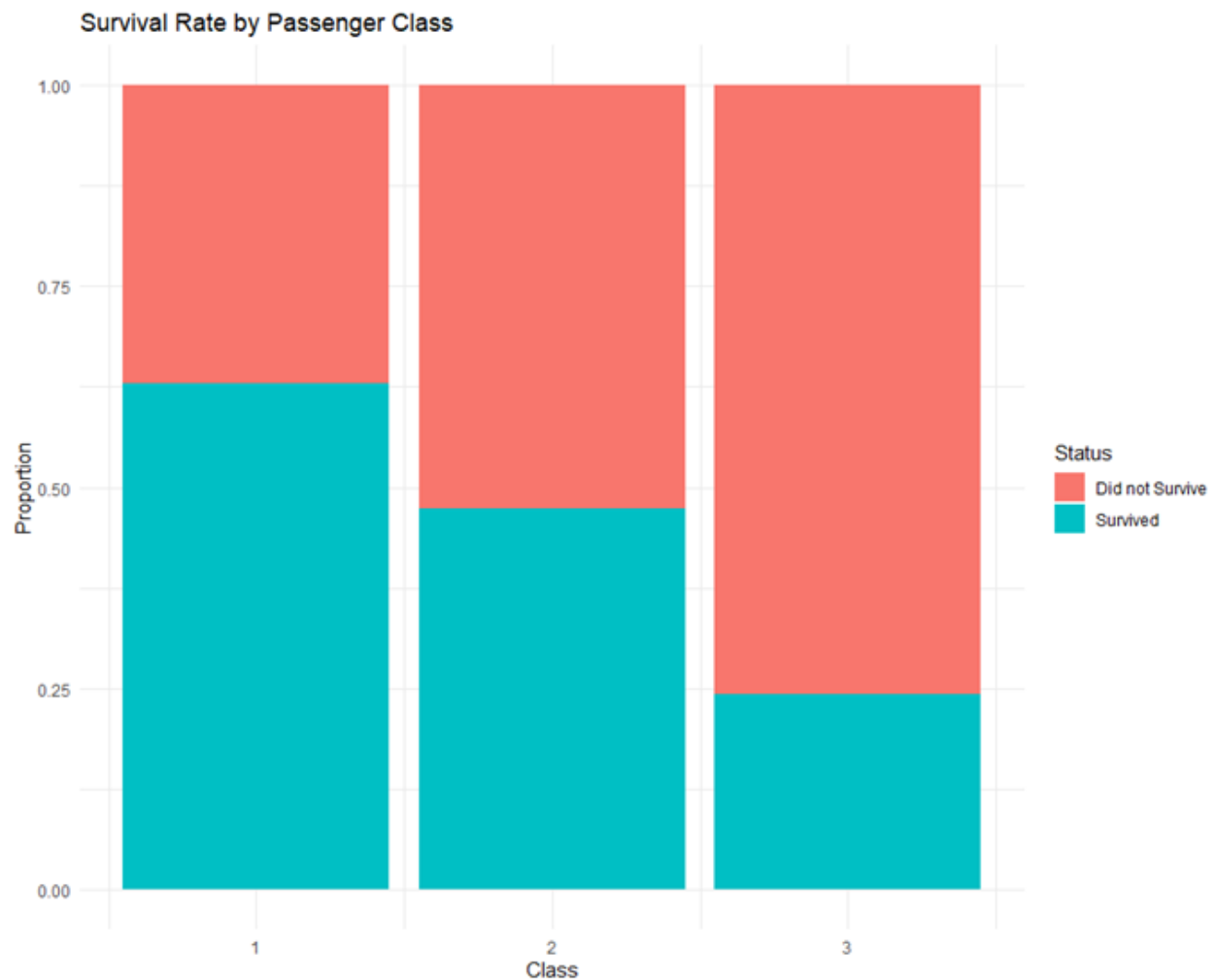
The famous "women and children first" motto seems to be clearly reflected in the data.



As the chart shows, **the survival proportion for women was drastically higher** than for men. This suggests that the passenger's sex will be a highly predictive variable.

Insight 2: Did social class matter?

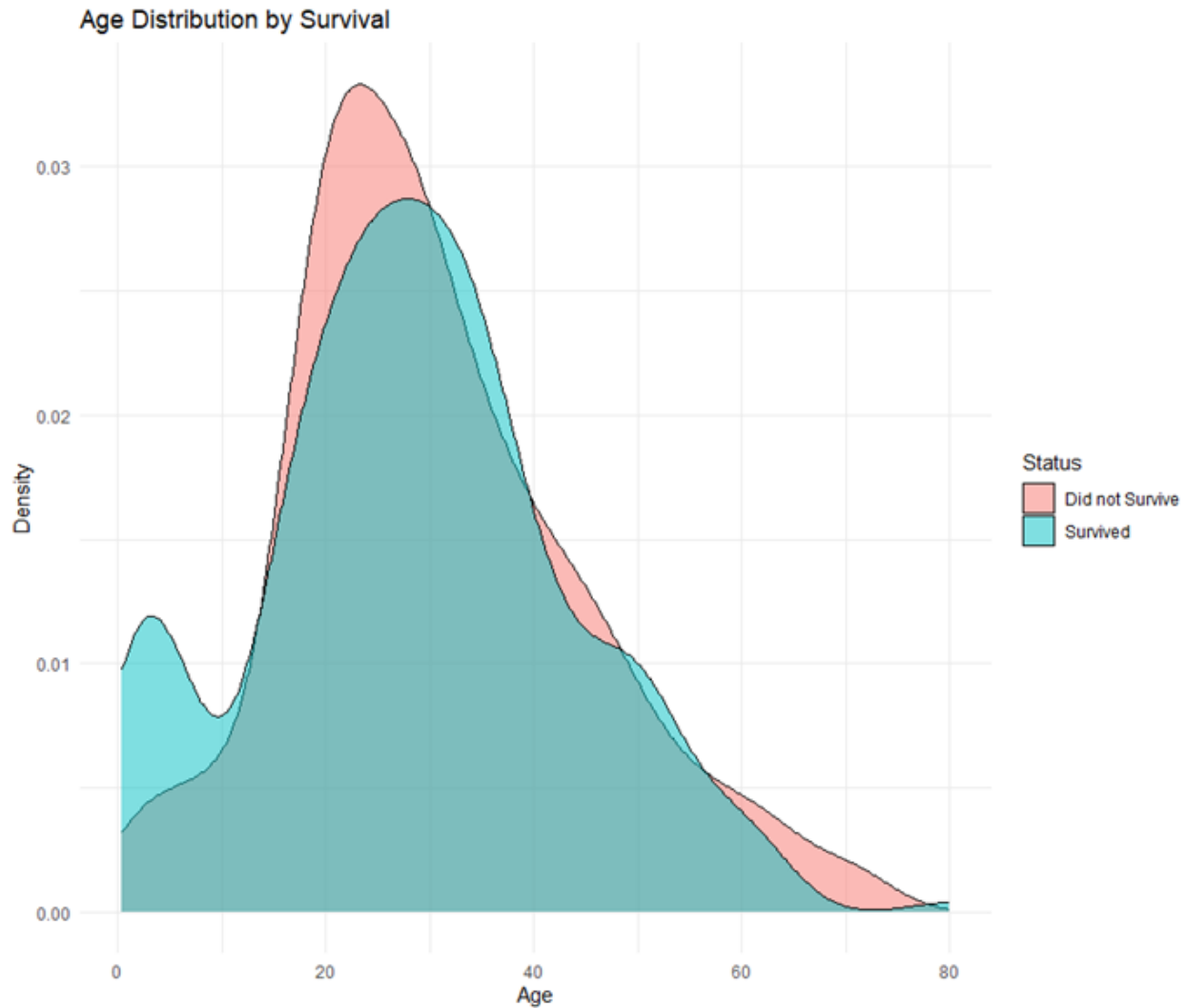
We analyze whether the ticket class influenced the probability of survival.



Survival is **strongly correlated with class**. More than 60% of first-class passengers survived, while only 25% of third-class passengers did. Socioeconomic status played a crucial role.

Insight 3: Were children prioritized?

We examine the age distribution of survivors.



A **notable survival peak is observed in the 0 to 10-year-old range**, reinforcing the idea that children were prioritized for rescue. It is also highlighted that a large proportion of the deceased were between 20 and 40 years old.

4. Feature Engineering

The model's performance can be drastically improved by preparing and transforming existing variables to create more informative ones.

4.1. Cleaning and Preprocessing

To apply transformations consistently, we first merge the training and test datasets. Next, we handle missing values using a technique called **imputation**:

- **Age and Fare:** We fill the null values with the **median** of each column, as it is a robust measure against outliers.

- **Embarked (Port of Embarkation):** We replace missing values with the **mode**, i.e., the most frequent port.

This process results in a complete and clean dataset, ready for modeling.

```
> test_data$Survived <- NA
> full_data <- rbind(train_data, test_data)
> median_age <- median(full_data$Age, na.rm = TRUE)
> full_data$Age[is.na(full_data$Age)] <- median_age
> mode_embarked <- names(sort(table(full_data$Embarked), decreasing = TRUE))[1]
> full_data$Embarked[full_data$Embarked == ""] <- mode_embarked
> median_fare <- median(full_data$Fare, na.rm = TRUE)
> full_data$Fare[is.na(full_data$Fare)] <- median_fare
> |
```

4.2. Creating New Variables

After cleaning, we enrich the dataset with new features:

- **Title:** We extract passenger titles (e.g., "Mr", "Mrs", "Miss") from the Name variable. We group less common titles into a single "Rare" category to simplify the model.
- **FamilySize:** We create this variable by summing the number of siblings/spouses (SibSp) and parents/children (Parch).

Finally, we convert categorical variables like Pclass, Sex, and the new Title to the **factor** type, a crucial step for the machine learning model to interpret them correctly.

```
> full_data$Title <- gsub('(.*, )|(\\..*)', '', full_data$Name)
> rare_titles <- c('Dona', 'Lady', 'the Countess', 'Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer')
> full_data$Title[full_data$Title %in% rare_titles] <- "Rare"
> full_data$Title[full_data$Title %in% c('Mlle', 'Ms')] <- 'Miss'
> full_data$Title[full_data$Title == 'Mme'] <- 'Mrs'
> full_data$FamilySize <- full_data$SibSp + full_data$Parch + 1
>
> full_data$Sex <- as.factor(full_data$Sex)
> full_data$Embarked <- as.factor(full_data$Embarked)
> full_data$Title <- as.factor(full_data$Title)
> full_data$Survived <- as.factor(full_data$Survived)
> full_data$Pclass <- as.factor(full_data$Pclass)
>
>
> train_final <- full_data[!is.na(full_data$Survived), ]
> test_final <- full_data[is.na(full_data$Survived), ]
> |
```

5. Modeling with Random Forest

For the classification task, we chose a **Random Forest** model. This is an *ensemble* method that builds multiple decision trees and combines their results to obtain a more robust and accurate prediction.

5.1. Model Training

We train the model with the following hyperparameters:

- `ntree = 500`: 500 trees will be built in the forest.
- `mtry = 3`: A random subset of 3 variables will be considered at each node split.
- `importance = TRUE`: The importance of each variable will be calculated.

```
> features <- c("Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked", "Title", "FamilySize")
>
> set.seed(123)
>
> model_rf <- randomForest(
+   x = train_final[, features],
+   y = train_final$Survived,
+   ntree = 500, # Número de árboles
+   mtry = 3,   # Número de variables a considerar en cada división
+   importance = TRUE
+ )
>
> print(model_rf)
```

Call:

```
randomForest(x = train_final[, features], y = train_final$Survived,      ntree = 500, mtry =
3, importance = TRUE)
      Type of random forest: classification
      Number of trees: 500
No. of variables tried at each split: 3
```

OOB estimate of error rate: 16.39%

Confusion matrix:

	0	1	class.error
0	495	54	0.09836066
1	92	250	0.26900585

5.2. Performance Evaluation

Performance was evaluated using the "**Out-of-Bag**" (OOB) estimate, an internal cross-validation technique of the algorithm.

- **Estimated OOB error rate: 16.39%**
- **Overall Accuracy: ~83.61%**

The **confusion matrix** breaks down the model's correct and incorrect predictions:

- **Non-Survivors (Class 0)**: The model was 90.1% correct.
- **Survivors (Class 1)**: The model was 73.1% correct.

The model is notably more effective at predicting who did not survive.

5.3. Variable Importance

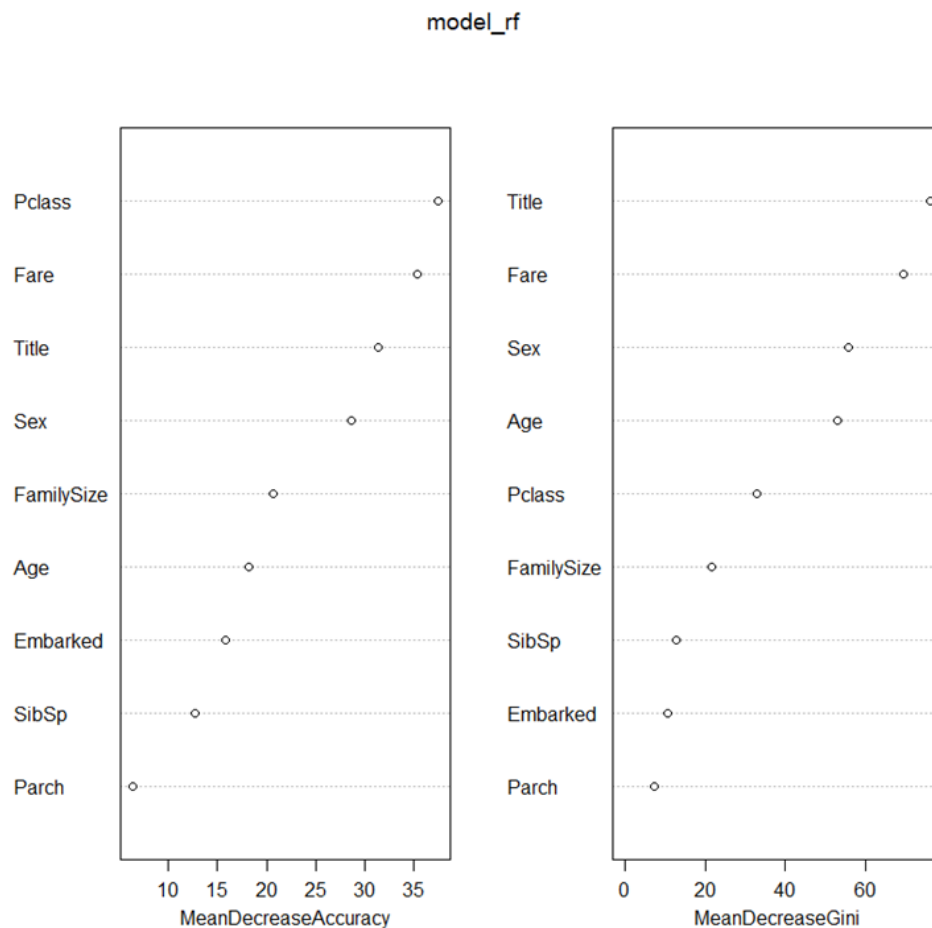
The Random Forest algorithm allows us to measure which variables were most

decisive for the prediction.

```
> importance(model_rf)
```

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
Pclass	18.9974158	33.376683	37.398068	32.81322
Sex	26.3951310	18.079232	28.557247	55.70693
Age	5.8273563	20.978433	18.178039	53.09485
Sibsp	12.4017570	1.784687	12.688605	12.66097
Parch	0.9683835	8.270123	6.349733	7.30709
Fare	20.2569228	27.538989	35.318145	69.73926
Embarked	3.4313307	17.695616	15.824210	10.60402
Title	28.0020278	22.032351	31.398195	76.40196
FamilySize	17.6479232	9.121161	20.717621	21.62115

The **MeanDecreaseAccuracy** and **MeanDecreaseGini** metrics indicate that the most influential variables were, in order of importance: **Title**, **Fare**, **Sex**, and **Pclass**. This result is consistent with our initial exploratory analysis.



6. Prediction and Submission to Kaggle

6.1. A Change of Tools: From randomForest to ranger

During the prediction phase on the test set, persistent anomalies were encountered with the `predict()` function from the `randomForest` package. Despite a thorough data validation process, the errors continued.

To ensure reproducibility and stability, the decision was made to switch to an alternative and more modern implementation of the algorithm: the **ranger** package. This package, implemented in C++, not only resolved the technical issues but also offered far superior computational performance.

6.2. Final Result

The final model was trained using **ranger**, ensuring a high-performance, fast, and robust result. Upon submitting the predictions to Kaggle, an **accuracy score of 77.03%** was achieved.



submission_titanic_ranger.csv
Complete · now

0.77033
