Predicting the survival of Titanic Passengers

Deepika Sridhar



Table of Contents

[Problem Statement 4](#_Toc26197815)

[Project Flow 4](#_Toc26197816)

[Business Understanding 4](#_Toc26197817)

[Data Understanding: 5](#_Toc26197818)

[Data Wrangling 7](#_Toc26197819)

[Data Transformation 7](#_Toc26197820)

[Cabin 7](#_Toc26197821)

[Fare 7](#_Toc26197822)

[Relatives 7](#_Toc26197823)

[Missing Values Treatment 8](#_Toc26197824)

[Embarked 8](#_Toc26197825)

[Age 8](#_Toc26197826)

[Cabin 9](#_Toc26197827)

[Outlier Value Treatment 9](#_Toc26197828)

[Age 10](#_Toc26197829)

[Fare Per Head 11](#_Toc26197830)

[Exploratory Data analysis 14](#_Toc26197831)

[Survived 14](#_Toc26197832)

[Pclass 14](#_Toc26197833)

[Sex 15](#_Toc26197834)

[Age 16](#_Toc26197835)

[Relatives 16](#_Toc26197836)

[Correlation Map 17](#_Toc26197837)

[Correlation Conclusion 17](#_Toc26197838)

[Machine Learning Analysis 19](#_Toc26197839)

[Steps 19](#_Toc26197840)

[Data Preparation 19](#_Toc26197841)

[Machine Learning Models 20](#_Toc26197842)

[Logistic Regression 20](#_Toc26197843)

[Random Forest Classifier 21](#_Toc26197844)

[XGboost 21](#_Toc26197845)

[Evaluation 22](#_Toc26197846)

[Hyper Parameter Tuning 23](#_Toc26197847)

[Evaluation 24](#_Toc26197848)

[Confusion Matrix 24](#_Toc26197849)

[Precision Recall Curve 24](#_Toc26197850)

[F-Score 25](#_Toc26197851)

[ROC Curve 25](#_Toc26197852)

[ROC Score 26](#_Toc26197853)

[Conclusion 27](#_Toc26197854)

# Problem Statement

## Project Flow



1. Start with the Business Problem
   1. Biggest ships of the time sank with a survival rate of less than 33%
   2. Some groups of people were more likely to survive than the others
2. Convert the Business Problem into a Data Problem
   1. Correlate the datasets so that overfitting can be reduced
3. Solve the Data Problem
   1. Correlate the datasets so that Porosity can be decreased
4. Convert the Data Solution into a Business Solution
   1. Decrease Porosity on a consistent basis to increase Yield
   2. Increase Yield to reduce costs and increase profits and customer satisfaction

## Business Understanding

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

We need to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (i.e. name, age, gender, socio-economic class, etc.) so that in the future, if Titanic were to set sail again, we can increase the survival rate for the passengers.

## Data Understanding:

The primary data objective is to determine which parameters correlate best with Survival rate. The given state of ship is that there is no digital thread throughout the entire process. This leads to inconsistent results in the end and no way of tracking what input affects the output. The only way to understand this process further is to digitally connect each parameter to the next one, piece by piece and begin to understand the relationship between them. The dataset has 12 parameters. They are:

|  |  |
| --- | --- |
| **Parameter Name** | **Description** |
| **Name** | This field specifies the name of the Passenger. |
| **Survived** | This tells whether the particular passenger was able to survive the shipwreck or not. This field can take on two possible values: • 1 - Passenger survived the shipwreck • 0 - Passenger was unable to survive the shipwreck |
| **PassengerId** | This is the unique ID given to each passenger travelling on the ship. The values are in the range 1-891. |
| **Ticket** | This parameter is a unique ID given to each purchase of the travel. This is more like a reservation ID and people who are travelling on the same reservation/Ticket will share this ID. |
| **Pclass** | Pclass or passenger class represents the traveling class of commuter. There were three classes. This field is more a representative of a socio-economic class prevalent those times. This field can take on 3 possible values, namely: • Class - 1 • Class - 2  • Class - 3 |
| **Cabin** | Since there are many missing values for Cabin, we replace them with "U". Also, since Cabin has many different values, we create a new parameter called Deck, which contains just the first character of the Cabin. |
| **Sex** | This field informs us about the gender of each passenger. It can take on only two possible values, namely: • Female • Male |
| **Age** | This parameter tells us about the age of the passenger in years. The values are in the range 0.42 to 80 years. |
| **Fare** | This tells us the total amount the ticket purchaser paid for the trip. This is also for the entire reservation and hence people who are travelling on the same reservation/Ticket will have the same details. |
| **Parch** | Parch stands for parents or children travelling onboard with them |
| **SibSp** | SibSp stands for total number of siblings and spouse travelling onboard. |
| **Embarked** | Embarked implies where the travelers embarked from. There are three possible values for Embarked: • Southampton - S • Cherbourg - C • Queenstown - Q |

# Data Wrangling

## Data Transformation

### Cabin

Since there are many missing values for Cabin, we replace them with "U". Also, since Cabin has many different values, we create a new parameter called Deck, which contains just the first character of the Cabin.

*data['Cabin'] = data['Cabin'].fillna("U")*

*data['Deck']=data['Cabin'].str[:1]*

### Fare

Since Fare is the total fare for everyone in a particular reservation, it is better to compute Fare\_per\_head and create a new parameter.

*data['Fare\_per\_head'] = data['Fare']/(data['Relatives'] + 1)*

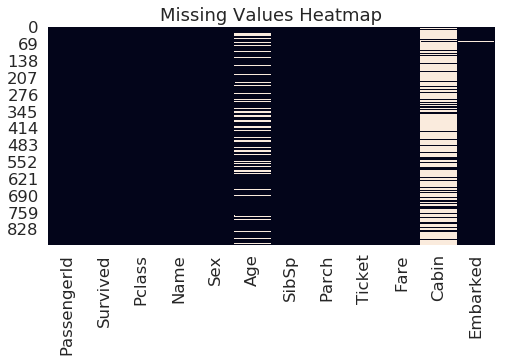
### Relatives

Parch stands for parents or children travelling onboard with them. SibSp stands for total number of siblings and spouse travelling onboard. We can merge these two fields to form a single field called Relatives

*data['Relatives']= data['SibSp'] + data['Parch']*

## Missing Values Treatment

The dataset had a few columns that had a few missing values. Embarked had two missing Values, while column Age and Cabin had 177 and 687 missing values respectively.



### Embarked

There are two missing values in Embarked. Since majority of the passengers boarded from S(Southampton),we can assign the modal value to missing rows.

*data.Embarked.fillna(data.Embarked.mode()[0], inplace = True)*

### Age

It is one of the biggest and somewhat columns to be filled. It has 177 missing values. Since the median value is 28, I used this to fill the missing rows in the data.

|  |  |
| --- | --- |
| **count** | **714** |
| **mean** | **29.699118** |
| **std** | **14.526497** |
| **min** | **0.420000** |
| **0.25** | **20.125000** |
| **0.5** | **28.000000** |
| **0.75** | **38.000000** |
| **max** | **80.000000** |

*data.Age.fillna(data.Age.mode()[0],inplace=True)*

### Cabin

Since there are many missing values for Cabin, we replace them with "U". Also Since Cabin has many different values, we create a new parameter called Deck, which contains just the first character of the Cabin.

*data['Cabin'] = data['Cabin'].fillna("U")*

*data['Deck']=data['Cabin'].str[:1]*

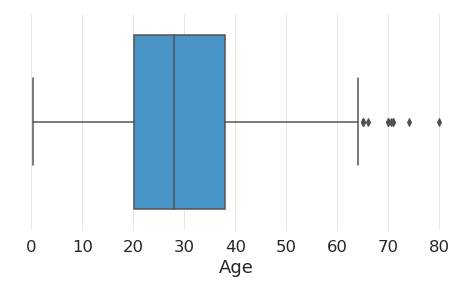
|  |  |
| --- | --- |
| **U** | **687** |
| **C** | **59** |
| **B** | **47** |
| **D** | **33** |
| **E** | **32** |
| **A** | **15** |
| **F** | **13** |
| **G** | **4** |
| **T** | **1** |

## Outlier Value Treatment

Machine learning algorithms are very sensitive to the range and distribution of attribute values. Data outliers can spoil and mislead the training process resulting in longer training times, less accurate models and ultimately poorer results. We now need to examine the outliers in our data which may skew our results. We need to determine what is the best value to replace them with.

### Age

From the below graph, it is evident that there are a few outliers here. We determine values which have a Z-score more than 3 and replace them with a value with Z-score around 3.



Hence values greater than a Z-score of 3 are:

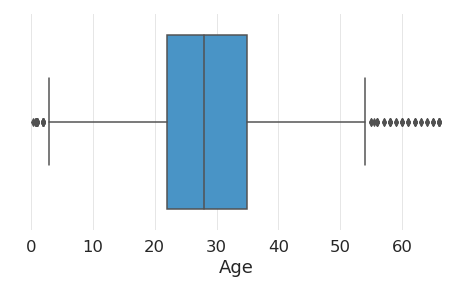
*data['Age'][[ 96, 116, 493, 630, 672, 745, 851]]*

|  |  |
| --- | --- |
| **Row Number** | **Value** |
| **96** | **71** |
| **116** | **70.5** |
| **493** | **71** |
| **630** | **80** |
| **672** | **70** |
| **745** | **70** |
| **851** | **74** |

We replace these values with a value with Z-score around 3. i.e. 66.

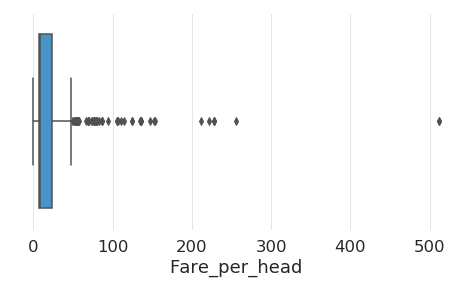
*data['Age'][[ 96, 116, 493, 630, 672, 745, 851]] =66*

Age data looks much better after this treatment.



### Fare Per Head

We repeat the same exercise done above to this field as well. We determine values which have a Z-score more than 3 and replace them with a value with Z-score around 3.



Hence values greater than a Z-score of 3 are:

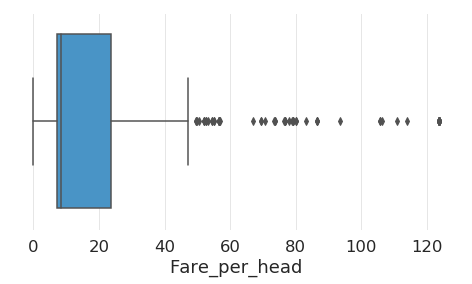
*data['Fare\_per\_head'][[195, 258, 269, 325, 337, 373, 380, 527, 557, 609, 679, 708, 716,730, 737]]*

|  |  |
| --- | --- |
| **Row Number** | **Value** |
| **195** | **146.5208** |
| **258** | **512.3292** |
| **269** | **135.6333** |
| **325** | **135.6333** |
| **337** | **134.5** |
| **373** | **135.6333** |
| **380** | **227.525** |
| **527** | **221.7792** |
| **557** | **227.525** |
| **609** | **153.4625** |
| **679** | **256.1646** |
| **708** | **151.55** |
| **716** | **227.525** |
| **730** | **211.3375** |
| **737** | **512.3292** |

We replace these values with a value with Z-score around 3. i.e. 123.7604.

*data['Fare\_per\_head'][[195, 258, 269, 325, 337, 373, 380, 527, 557, 609, 679, 708, 716,730, 737]]*

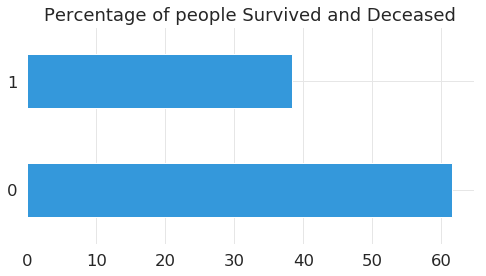
Age data looks much better after this treatment.



# Exploratory Data analysis

### Survived

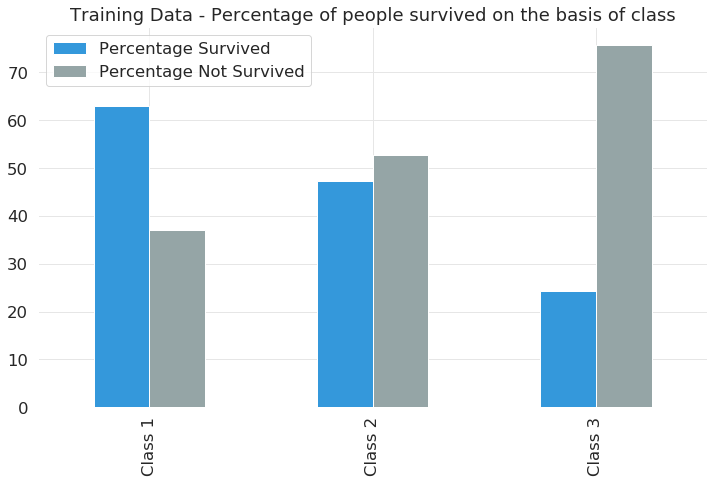
A comparison revealed that more than 67% of the passengers died.



Pclass

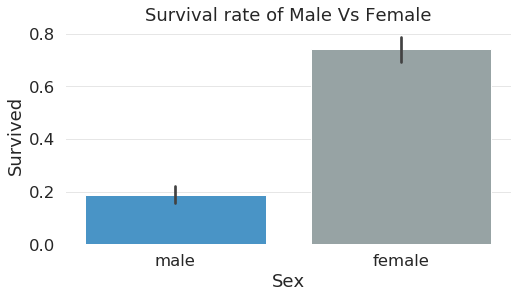
In the dataset, a clear majority **(491)** traveled in the third class, followed by the second **(216)** and then the first **(184)**. Number of passengers in the third class was more than the number of passengers in first and second class combined.

With the missing survived values, more than 65% of the first-class passengers were rescued. The pattern differed for the second-class and third-class survivors as roughly around 55% of the second class passengers lost their lives. The numbers skyrocketed for the third-class passengers. More than 75% of the third-class passengers couldn't survive the disaster.



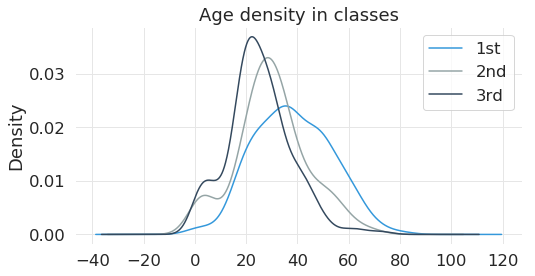
### Sex

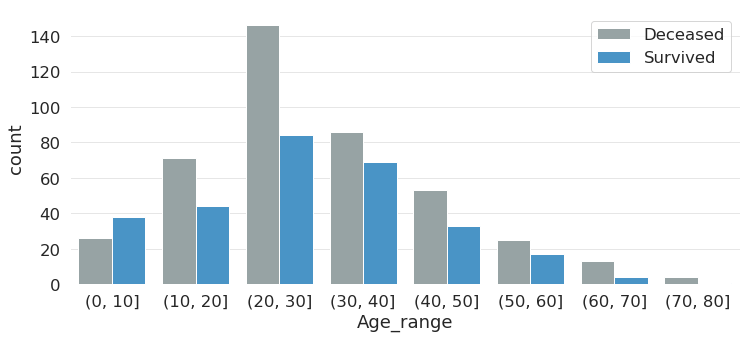
Approximately 35 % of the passengers were women compared to male passengers who constituted the rest 65%. Oddly, the survival rate among women was more than the survival rate amongst men.



Age

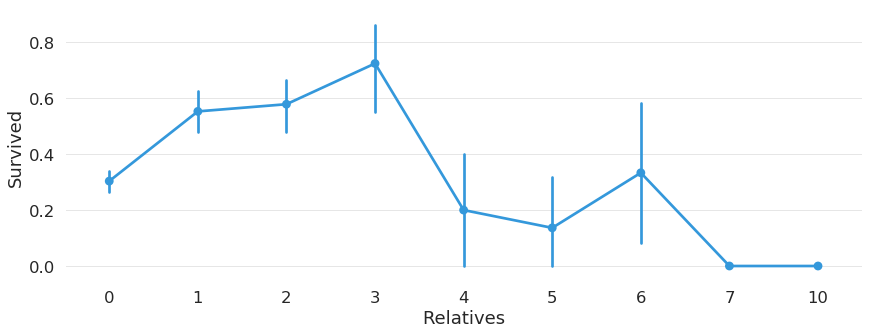
The values are in the range 0.42 to 80 years. The average age of passengers was just below 30 years. Children under the age 10 were more likely to survive than not.





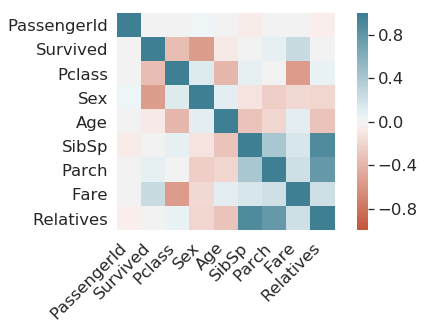
### Relatives

Clearly survival rates for people with 0-3 relatives was the highest.



## Correlation Map

The heatmap tells us the relative correlation between each of the parameters. As evident there is a very strong correlation between Survived and Pclass and Fare.



## Correlation Conclusion

* Pclass Vs Survived
  + They were one of the important indicators of survival
* Sex Vs Survived
  + Women were more likely to survive than men
* Fare Vs Survived
  + People who paid a higher fare were more likely to survive
* Age Vs Survived

Children under the age 10 were more likely to survive than not

# Machine Learning Analysis

## Steps

1. Data Prep
   1. Change the Pclass hierarchy
   2. Continuous attributes must be binned, and Dummy variables created
   3. Remove categorical variables
   4. Train Test split
   5. Variance Inflation Factor
2. Models
   1. Logistic Regression
   2. Random Forest
   3. XGBoost
3. Evaluation
   1. Confusion Matrix
   2. Precision Recall Curve
   3. F-Score
   4. ROC Curve

### Data Preparation

#### Pclass

Since the machine considers 3 > 2 > 1, which is quite opposite to our assumption in Pclass we interchange 3 and 1 classes.

*data['Pclass'].loc[data['Pclass']==3] = 4*

*data['Pclass'].loc[data['Pclass']==1] = 3*

*data['Pclass'].loc[data['Pclass']==4] = 1*

#### Dummy Variable creation to remove continuous variables

Since Age, Relatives and Fare\_per\_head are continuous variable, we need to create dummy variables by binning the Age values and then creating dummy variables out of them.

#### Remove Categorical Variables

We remove categorical variables with the help of Label Encoder and One hot encoding. Label encoder will convert all the categorical values into numeric values. But the machine may think that these numeric values may have some kind of hierarchy. To avoid this problem, we add a step of One hot encoding, which converts all the numeric variables to binary variables.

#### Splitting the Data into training and test set

We now split our data into Training and Test set in a ratio of 70/30. We make use of the pre-existing function called ***train\_test\_split*** to accomplish this. The training set contains the known output in this case, ‘Survived’ column and the model learns on this data to be generalized on other data later on. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset.

#### Information Value Analysis and Variance Inflation Factor

Attribute analysis dives more deeply into individual attributes and tries to tell you which segment of that variable has the strongest connection with the target variable. 0 is the lowest information value an attribute can have and 1 the highest. We then make the decision of whether to include the attribute in our model or not based on the following criteria:

* Information value < 0.1, the parameter or the attribute does not add any value to our analysis and hence is excluded
* Information value > 0.8, the parameter or attribute is suspicious or too good to be true when it comes to predicting the target variable and hence needs to be excluded
* Information value between 0.1 and 0.8, the attribute or parameter has good predictive power and is included in our model.

Variance Inflation Factor talks about multicollinearity in the data. When two or more independent variables are correlated, this may inflate our results and hence such parameters must be removed from our data. The criterion we use here is listed below:

Hence, parameters which have a Variance Inflation factor greater than 5 are removed by our algorithm.

## Machine Learning Models

### Logistic Regression

Logistic Regression is a classification algorithm that uses a sigmoid function with independent values to predict a value for the dependent variable i.e. ***Survived***. We trained and tested this algorithm on our data. We compared the test results with the actual results and the results are listed below:



### Random Forest Classifier

We trained a Random Forest Classifier to predict a value for the dependent variable i.e. ***Survived***. We trained and tested this algorithm on our data. We compared the test results with the actual results and the results are listed below:



### XGboost

We trained a XGboost Classifier to predict a value for the dependent variable i.e. ***Survived***. We trained and tested this algorithm on our data. We compared the test results with the actual results and the results are listed below:



### Evaluation

We use ROC – AUC as a criterion for evaluating the above 3 algorithms. ROC - AUC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, Higher the AUC, better the model is at distinguishing between patients with disease and no disease.



According to the above table, XGBoost Classifier performs the best and hence choose XGBoost Classifier as the algorithm for this problem. We continue with our analyses and Hyper Parameter tuning.

### Hyper Parameter Tuning

Hyper Parameter Tuning is the process of choosing a set of optimal hyperparameters for a machine learning algorithm. The same kind of machine learning algorithms can require different constraints or weights to generalize different data patterns. Hence, we need to tune these parameters so that our model can optimally solve the machine learning problem. We use ***GridSearchCV*** to find our hyperparameters.

We used the following values for finding the optimal parameters:



The attribute ***best\_params\_*** gives the optimal parameter values the machine has chosen for us. The optimal parameter values here are:

|  |  |
| --- | --- |
| **booster** | **gbtree** |
| **colsample\_bytree** | **0.6** |
| **gamma** | **0** |
| **max\_depth** | **5.00** |
| **min\_child\_weight** | **1** |
| **subsample** | **1** |

Using the above parameter values and running the XGBoost algorithm again, we get an ROC\_AUC score of 74.14 which is ~ 75%.

## Evaluation

### Confusion Matrix

It is a performance measurement for our machine learning classification problem. The confusion matrix for our problem is given below:



The first row is about the not-survived - predictions: 136 passengers were correctly classified as not survived (called true negatives) and 21 were wrongly classified as not survived (false positives).

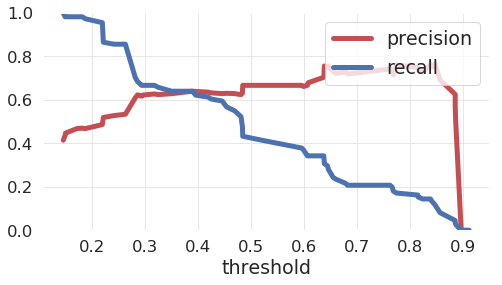
The second row is about the survived - predictions: 67 passengers were wrongly classified as survived (false negatives) and 44 where correctly classified as survived (true positives).

### Precision Recall Curve

Precision is the number of true positives divided by the total number of elements labelled as positive. Recall is the number of true positives divided by the total number of elements which were actually positive.

We plot a Precision Recall curve to study the output of the classifier

As you can see, Recall is rapidly falling at a precision greater than 0.6, so we can choose precsion and recall tradeoff at around 60%



### F-Score

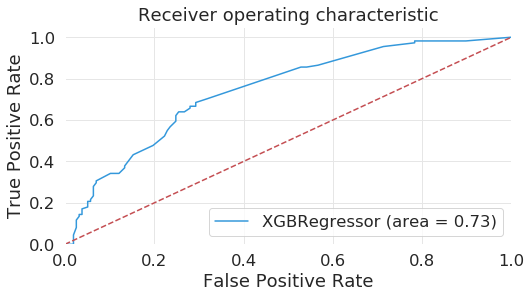
We can combine precision and recall into one score, which is called the F-score. The F-score is computed with the harmonic mean of precision and recall. Note that it assigns much more weight to low values. As a result of that, the classifier will only get a high F-score, if both recall and precision are high.

F -score for our problem is **0.5111111111111111**. This is a low value since our recall values were fairly low.

### ****ROC Curve****

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve. This curve plots the true positive rate (also called recall) against the false positive rate (ratio of incorrectly classified negative instances).

The ROC curve is given below:



The red line in the middle represents a purely random classifier (e.g a coin flip) and therefore your classifier should be as far away from it as possible. Our XGBoost model seems to do a good job.

Of course, we also have a tradeoff here, because the classifier produces more false positives, the higher the true positive rate is.

### ROC Score

The ROC AUC Score is the corresponding score to the ROC AUC Curve. It is simply computed by measuring the area under the curve, which is called AUC. A classifier that is 100% correct, would have a ROC AUC Score of 1 and a completely random classifier would have a score of 0.5.

We got an ROC-AUC-Score of **0.7431284787972686** .

# Conclusion

The best way to increase the survival rate of the passengers if Titanic ever set sail again across the Atlantic is to make sure we issue tickets to passengers whose chance of survival is the highest. The ideal passengers whose chances of survival are highest are:

* Female Passengers
* Passengers with 0-3 relatives on board
* Passengers on First Class Ticket
* Children of age below 10, passengers aged between 10-20 or 30-40
* Passengers embarking from Southampton or Cherbourg