**RMS Titanic**

The curious case of the RMS Titanic- the super luxurious ocean liner from British shipping company White Star Line that went deep down under on her maiden voyage in early 1912- has managed to not just intrigue generations of artists and filmmakers for whom this subject has been the founding material of a plethora of prominent works of art.. but also garner a lot of public attention due to the magnitude of its ill fate.

Data scientists too have not been far behind in joining the Titanic bandwagon. And with good reason. After all, a data scientist’s chief concern is to foresee the unforeseen.. predict the unpredictable. And what better case study can one work on..than the historic RMS Titanic itself. A disaster that shook up the world and continues to haunt many till date. One that could have been averted or perhaps the extent of its damage (to be read as: loss of precious life) reduced.

So, donning the Data Scientist hat, let us find pressing answers to the burning question on everyone’s minds..

Was it a matter of survival of the fittest..or the richest?

Let us uncover the mystery today.

Problem Definition

As with any data science problem, we will begin by defining our problem statement, which would help to serve as the focal point in our project.

In this case, we will be using the data collected from the different passengers aboard the fateful RMS Titanic. And our principal objective is to predict the chances of survival of an arbitrary passenger given the various connected details such as:

* PassengerId : Unique Identification Number for each passenger (Integer data type)
* Pclass : Ticket Class in which the passenger was traveling (1- First Class, 2- Second Class or 3-Third Class) (Integer data type)
* Name : Name of the passenger (Object data type)
* Sex : Gender for every passenger (Male or Female – Categorical data type)
* Age : Age of passengers in number of years (Float data type)
* SibSp : Number of siblings and/ or spouse aboard the ship (Integer data type)

0 -- No sibling/ spouse,

1 -- one sibling/ spouse,

2 -- one sibling and spouse,

3 – two siblings and spouse,

4 – three siblings and spouse,

5 – four siblings and spouse

* Parch : Number of parents and/ or children aboard Titanic (Integer data type)

0 -- No parent/ child,

1-- one parent/ child,

2 -- a parent with a child or both parents or two children,

3 – a parent with two children or both parents with a child,

4 – a parent with three children or both parents with two children,

5 – a parent with four children or both parents with three children

* Ticket : Ticket Number
* Fare : Passenger Fare
* Cabin : Cabin Number allocated to each passenger
* Embarked : Port of Embarkation ( S- Southampton in England, C- Cherbourg in France, Q- Queenstown in Ireland)

TARGET variable **Survived** : whether or not a passenger survived ( 1 – Survived or 0 – Did not survive) ( Integer data type)

For this, we are presented with a single dataset containing details for 891 passengers that were aboard the ship.

The task at hand then is to create a machine learning model using this dataset to accurately predict the survivors and those that weren’t as lucky. This makes it a classification problem.

Data Analysis

Let us begin by importing the required, standard python libraries first.

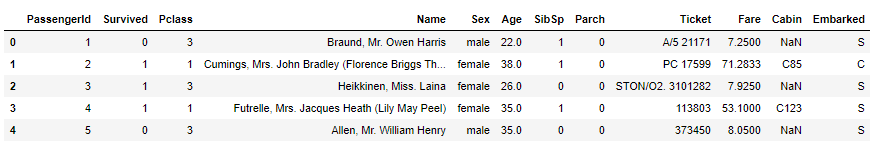
Viz. Pandas for basic data analysis and Numpy for some number crunching.



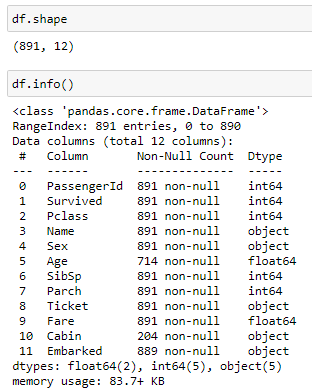
Then, we will load our models with some ammunition (our data!).



The customary sneak-peak before we are ready to roll.. with our analysis.

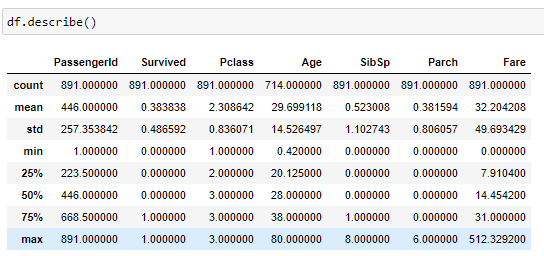
****

We take a brief overview of the dataframe using .shape property and .info() function.



With this, we can get a sense of the complete dataset including the number of observations, the different attributes, the data types for each feature, any null values present, etc.

We have 2 float features, 4 integer attributes and 5 object ones. Null values can be seen in the Age and Cabin fields. Target variable is integer data type with 2 values- 0 and 1.

****

The .describe() function too lets out interesting facts, if you were to let your thoughts hover a bit.

The mean of the target variable – Survived stands at 0.38 indicating that only 38% of the 891 passengers in this dataset lived to tell the tale.

Titanic had on board passengers aged as young as 4 months to as old as an incredible 80 years.

It had large families traveling together with the number of siblings and spouse (presumably 0 or 1) ranging from 0 to 8. And number of parents and children being anywhere between 0 and 6.

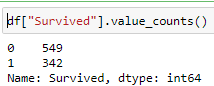
Average fare was 32 with the maximum fare going up to 512.

EDA

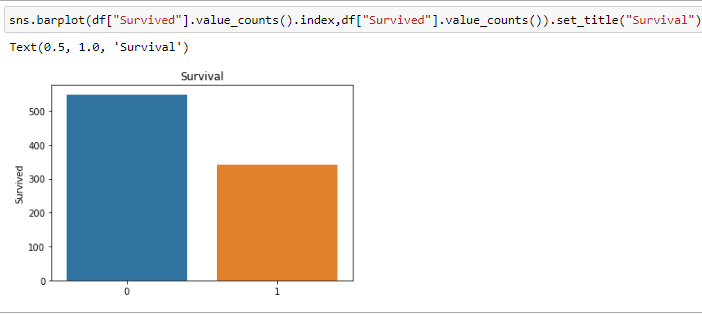


Visualisation libraries imported, let us get down to some visual exploration.

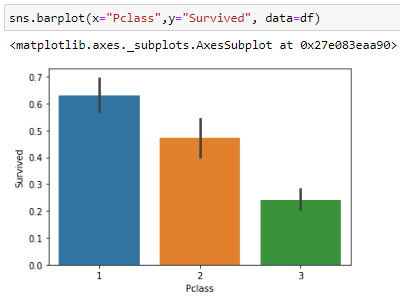
Checking the survivor count with .value\_counts function, we get the survivor count as 342.



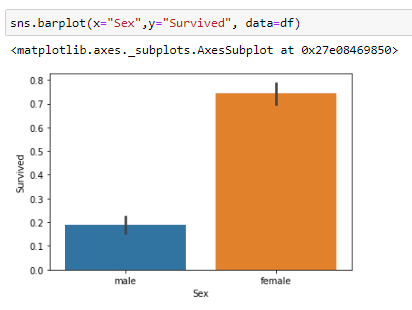
Plotting this gives us the overall chance of survival.

****

The odds of survival is rather low.

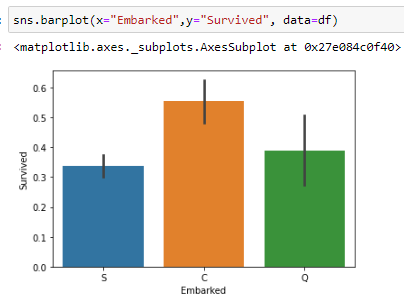
****

Examining the class-wise survival chances reveals that passengers in the first class had 62% chance of surviving, second class passengers had 48% chances whereas those traveling in third class had a 25% chance at life after Titanic.

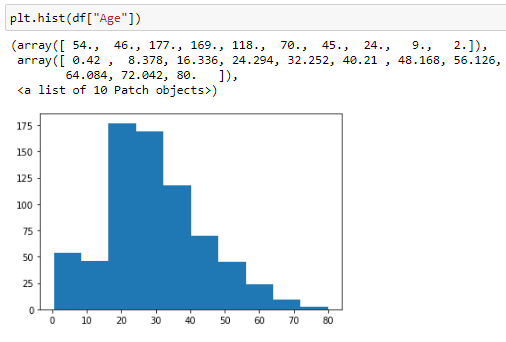
****

Women had a whopping 72% survival rate as compared to men who had a meagre 19% chance.

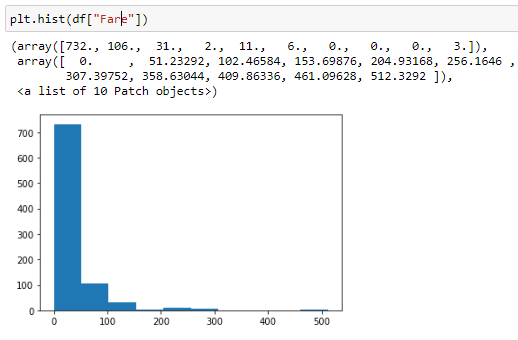
Let us see if the ports of embarkation had any influence over survival.

****

Cherbourg had the highest number of survivors.

****

A histogram on Age depicts a young population aboard the ship with most aged between 20-40 years.

****

Fare is largely concentrated in the less than 100 range with the exceptions of a few high values that make this a right tailed distribution.

With this, we conclude our visual explorations and progress towards processing and preparing our data. For its final journey.

Pre-processing Pipeline

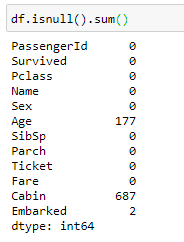
Here, we will be needing the encoders and scalers.

The skewness functions and outlier detections. To prune and prime our raw data.

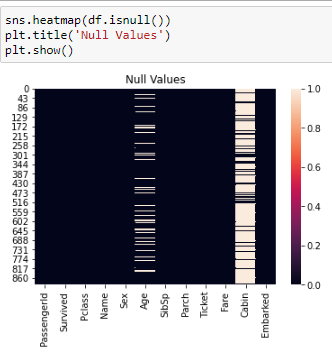


First, we will deal with the null values in the dataset.

Look for them with the .isnull() function like so:



We see 177 null values in Age column, 687 in Cabin feature and 2 null values in Embarked.

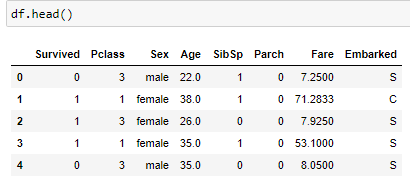
One other way to check for null values in the data is to use a heatmap to represent them.****

We notice that Cabin has too high a proportion of null values to be used meaningfully to make any predictions. Let us proceed to drop this from our dataset.

We also have features such as the PassengerID, Name and Ticket which are unique to each passenger and do not possibly influence the chances of survival.

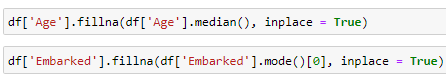
****

Our dataset now looks like this:

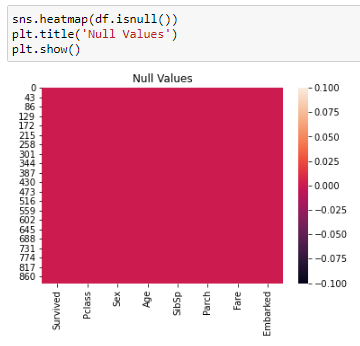
****

With just the relevant details we need to feed our model.

Imputing the missing values in Age with median value and Embarked with mode value:

****

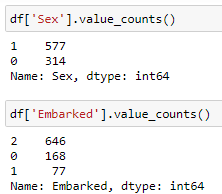
A quick look at the null values again here before we move on to encoding.

****

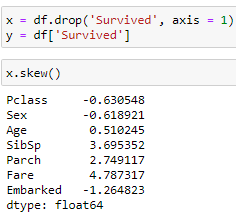
We will use the Label Encoder to transform Sex and Embarked columns, which are categorical.

****

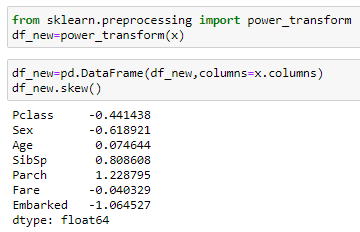
Viewing the value types:



Next, we split the data into test and train sets. And check the skewness present in the ‘x’ data.



Skewness above the threshold value of -+0.5 is observed. Hence, we will proceed to normalize this skewed data with the power transform function.



We see that skewness is now well within threshold levels and we can safely proceed to use the Standard Scaler technique to scale the wide ranged values of the dataset.

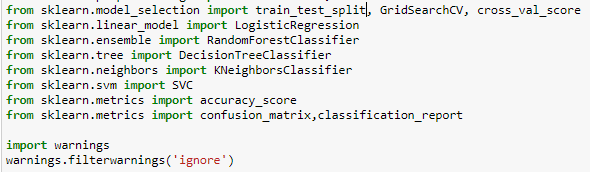


We have now reached the end of the pre-processing pipeline and our data is now all prepped-up to throw out all the right answers.

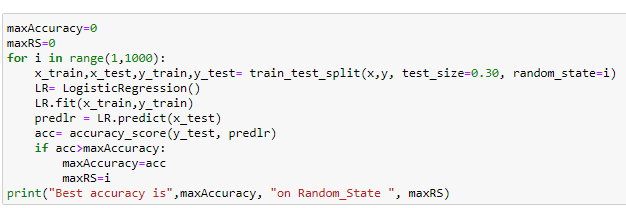
Building Machine Learning Models

Again, we start this stage by importing all the libraries and machine learning algorithms.

Being a classification problem, we have taken into consideration classifier models such as the Logistic Regression, Random Forest Classifier, Decision Tree Classifier, K Neighbours Classifier, Support Vector Classifier.



Before we dive into the nitty-gritties of model building, we will first arrive at the optimal random state using Logistic Regression as our baseline model.



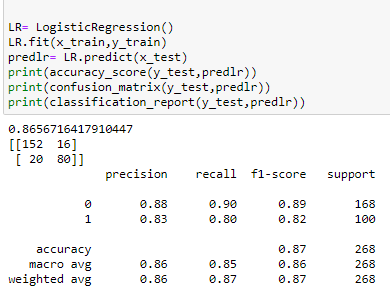
Using a for loop on a train test split, keeping the test size as 0.30, we get the best random state at 710 with the highest accuracy score achieved at 86.57%.

With this random state of 710 as standard, we create the train test split.

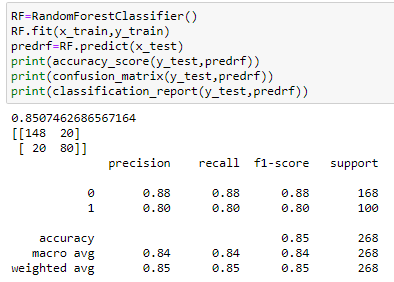


Now, it is time to test each of our classifier models on the data and ascertain the performance of accuracy scores for each of them.

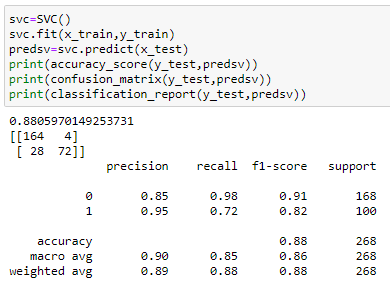
We begin with Logistic Regression where we achieve 86.57% accuracy score.



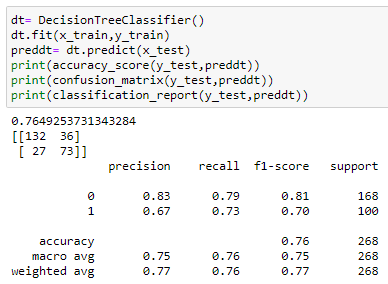
Followed by the Random Forest Classifier model which gives out an accuracy score of 85.07%.

****

Next up is the Support Vector Classifier model which achieves an accuracy score of 88.06%.

****

Decision Tree Classifier model is on the lower end of the accuracy spectrum at 76.49%.

****

Lastly, we have the K Neighbors Classifier scoring 85.82% in accuracy levels.

The best performer among all the models seems to be the Support Vector Classifier at 88.06%.

But, let’s not conclude just yet. It may be a case of over-fitting.

And we are about to find out if it truly is so.

Cross-validation comes to our rescue here and with this, we are able to test each and every portion of the dataset to overcome any shortcomings or defects in our predictions.

Since cross validation splits the dataset into a specific number of folds n ( n is 5 in our case) and runs the training and testing process n number of times such that every fold acts as an evaluation fold at least once, the problem of over and under fitting is taken care of.

****

Cross Validation Score of Logistic Regression Model: 0.7901450003138535

****

Cross Validation Score of Random Forest Classifier Model: 0.8081099742640137



Cross Validation Score of Support Vector Classifier Model: 0.8170610758897746

****

Cross Validation Score of Decision Tree Classifier Model: 0.7744334944447931

****

Cross Validation Score of K Nearest Neighbours Classifier Model: 0.7968928504174251

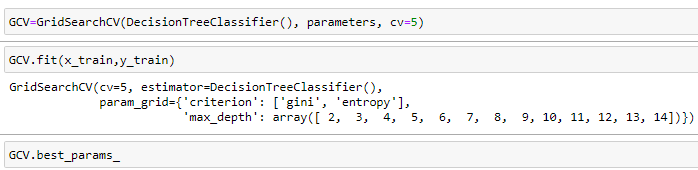
Comparing the accuracy scores obtained earlier with the cross validated scores derived now, we arrive at the conclusion that the Decision Tree Classifier is the best performing model. As the deviation between the accuracy score and cross validated score is minimal for the same.

Now that we have zeroed in on our classifier model, it is time to find out the hyper parameters that will most influence our score for the better.

For Decision Tree Classifier, we will be tuning max\_depth and criterion parameters for optimal performance.

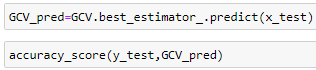


Using Grid Search CV and these parameters passed in, we use x\_train and y\_train fitting to arrive at the best parameters.

****

{'criterion': 'entropy', 'max\_depth': 8}

With criterion as entropy and max depth of 8 , we predict values on y\_test and draw comparisons on predictions with x\_test values.



This gives us an accuracy score of 84.32%

And with this, we store the GCV best estimator model in a pickle file using Joblib library. This enables us to use the model directly for predictions outside of the Jupyter notebook.

Concluding Remarks

This machine learning project helped us to model predictions for survival of passengers given a set of related details such as gender, age, ticket class, fare, family accompanying, port of embarkation, etc.

We have gone through the rigours of model building, in that, the in depth data analysis, data exploration and visualisation, feature engineering, data pre processing pipelines have all been extensively covered.

We have also computed optimal random state, tested multiple classifier models, checked cross validation scores to arrive at the best fit model.

Next, we have used hyper parameter tuning on our best fit model, that is, the Decision Tree Classifier model to arrive at an even better and top of the line machine learning model.

It is now time for me to bid good bye until I meet you again with another interesting case study with new challenges and learnings, that we can wrap our minds around and crank up a super machine learning model with new algorithms, new techniques and many more fun things to learn. Adios.