# Project: Titanic - Machine Learning from Disaster

## Introduction:

We used machine learning to create a model that predicts which passengers survived the Titanic shipwreck. On April 15, 1912, during its 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.

Thus, we have built a predictive model that predicts “what sorts of people were more likely to survive” using passenger data (i.e., name, age, gender, socio-economic class, etc).

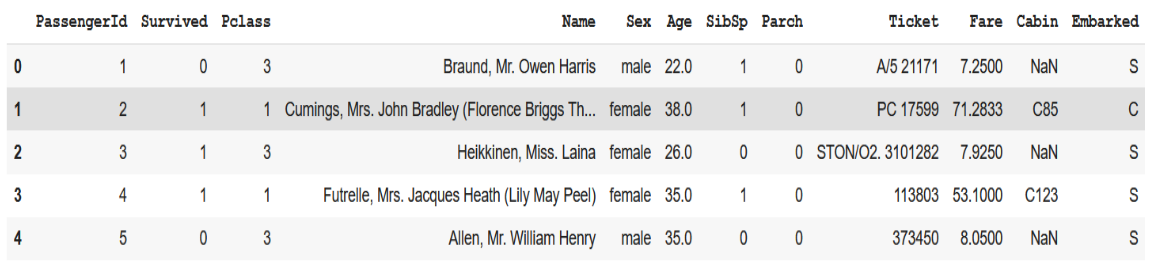
## Dataset:

We used Kaggle competition "Titanic: Machine Learning from Disaster" to retrieve necessary data and evaluate accuracy of our predictions. The historical data has been divided into two groups, a 'training data' and a 'testing data'. For the training set, we are provided with the outcome (whether a passenger survived or not). We used this set to build our model to generate predictions for the test set.

Training and Test data come in CSV file and contain the following fields:

- Passenger ID
- Survival (0 = No; 1 = Yes)
- Passenger Class
- Name
- Sex
- Age
- Number of passenger's siblings and spouses on board
- Number of passenger's parents and children on board
- Ticket
- Fare
- Cabin
- City where passenger embarked

Snapshot:



| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

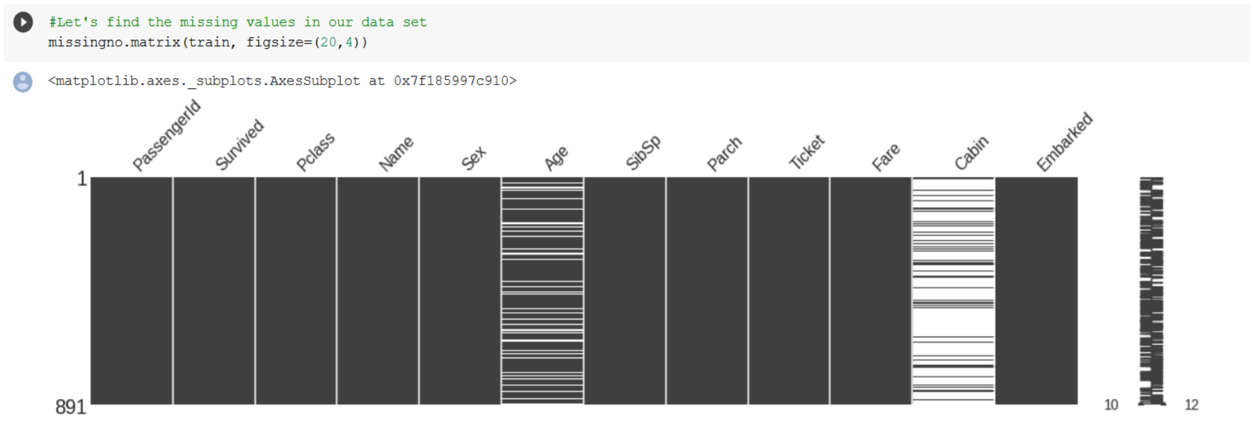
## Data Analysis:

In this data set we have so many different types of data, so before we feed into the machine learning model, we need to analyze the data very carefully as the model prediction is dependent on how we train our model. So, we have numbers, and words, letters, and NaNs (none values that we need to remove). Which means, in the numerical data we have, PassengerId, Survived (we are traying to predict on this column itself so remove this column as of now), Pclass, Age, SibSp, Parch and Fare. The categorical features contain Sex and Embarked. And we don't see any patters in Name, Ticket and Cabin.

Machine learning algorithms perform well with the numerical data, since Pclass, SibSp and Parch are already all in numerical form, and Age too. On the other hand, in our categorical features, we have Sex and Embarked. We have changed these in numerical form using the **LabelEncoder()** function from the **sklearn library**. Finally, Name, Ticket and Cabin are left. For now, we won't worry about the Name column to make a prediction. Same way we don't consider Ticket as well.

The second most important step in data analysis is to find the missing (NaN) values from the features because the model can't predict anything on null data.

Here is the Snapshot:

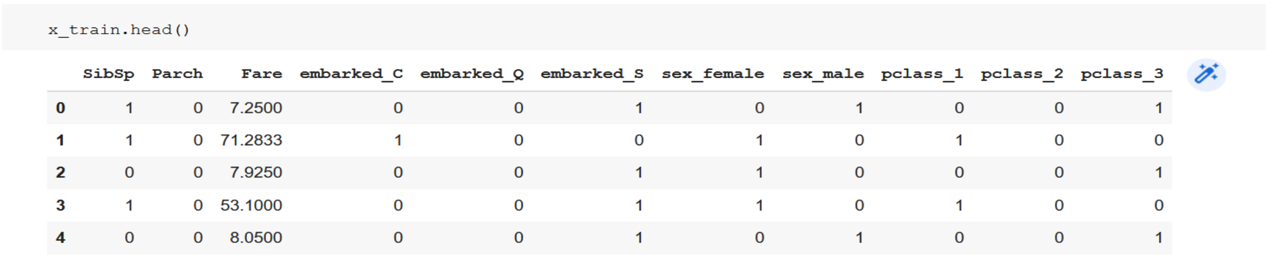


```
#Let's find the missing values in our data set
missingno.matrix(train, figsize=(20,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f185997c910>
```

Clearly, we have so many NaN values (white lines) in Age (177) and Embarked (687). We can fill-up these rows with some dummy values, but it is not the good idea as it is quite huge number. So, we have not used these for training purpose.

Hence, our final training dataset should look like as follow:

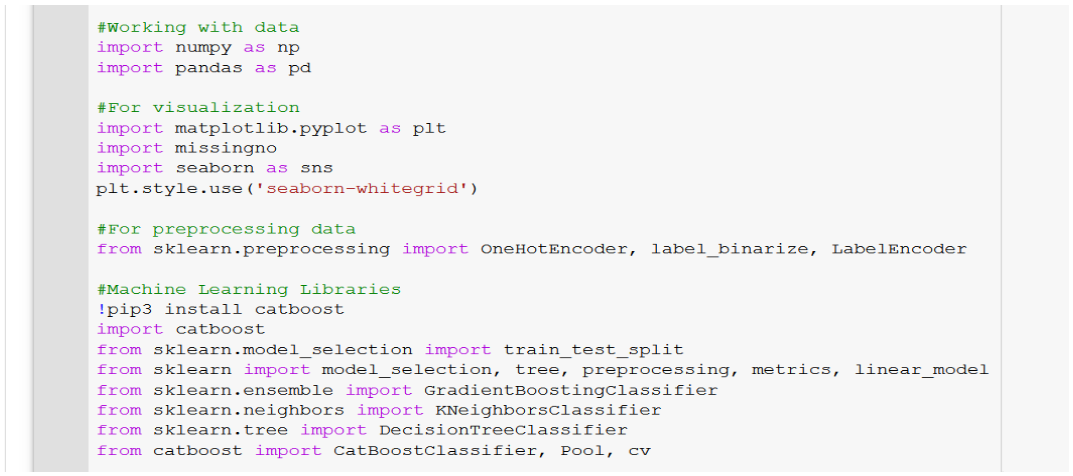


```
x_train.head()
```

| | SibSp | Parch | Fare | embarked_C | embarked_Q | embarked_S | sex_female | sex_male | pclass_1 | pclass_2 | pclass_3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 7.2500 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| **1** | 1 | 0 | 71.2833 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **2** | 0 | 0 | 7.9250 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| **3** | 1 | 0 | 53.1000 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| **4** | 0 | 0 | 8.0500 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

## Key ideas:

As we discussed above, we need to predict weather a passenger survived or not by feeding the training data set into the machine learning model. To create the model, we need following libraries.



```
#Working with data
import numpy as np
import pandas as pd

#For visualization
import matplotlib.pyplot as plt
import missingno
import seaborn as sns
plt.style.use('seaborn-whitegrid')

#For preprocessing data
from sklearn.preprocessing import OneHotEncoder, label_binarize, LabelEncoder

#Machine Learning Libraries
!pip3 install catboost
import catboost
from sklearn.model_selection import train_test_split
from sklearn import model_selection, tree, preprocessing, metrics, linear_model
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from catboost import CatBoostClassifier, Pool, cv
```

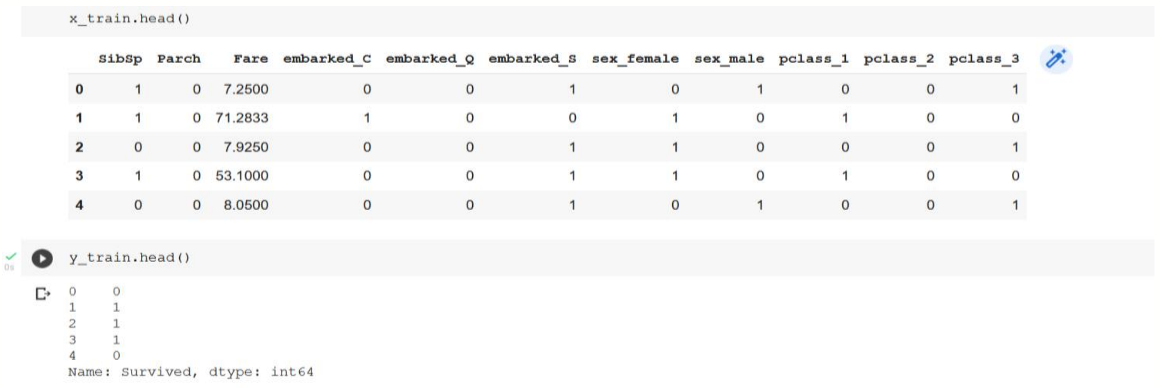
We need Numpy and Pandas to work on vectors and data frame. As you can see above, we have imported different supervised machine learning libraries. Because we have tried KNN, Gradient Boosting Tree, Decision Tree, and CatBoost in order to select the best one (which has highest accuracy). Other libraries for visualization. Lastly, used sklearn preprocessing libraries to encode and/or normalize some features.

**Note:** Running multiple models is fine on our small Titanic dataset. But might not be the best for larger datasets as it requires very high computation process.

**The main key point** in this machine learning model is that we have measured the model's performance not just based on Accuracy, but we have also used **Cross-validation Accuracy**, so the model can perform better on unseen data. Note: we have briefly discussed the difference between Accuracy and Cross-validation Accuracy further.

## Model Training:

Now, subsequently the training of the model will begin on x_test, and y_train (on which we need to predict).
Snapshot:



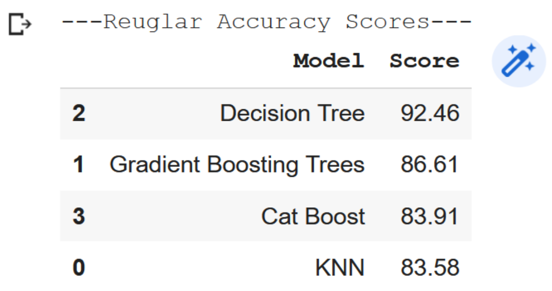
```
x_train.head()
```

|   | SibSp | Parch | Fare | embarked_C | embarked_Q | embarked_S | sex_female | sex_male | pclass_1 | pclass_2 | pclass_3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 7.2500 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| **1** | 1 | 0 | 71.2833 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **2** | 0 | 0 | 7.9250 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| **3** | 1 | 0 | 53.1000 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| **4** | 0 | 0 | 8.0500 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

```
y_train.head()
```

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

After training the following are the **result** of the Accuracy score:



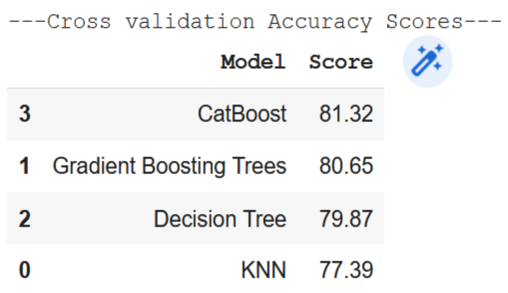
```
---Reuglar Accuracy Scores---
```

| | Model | Score |
|---|---|---|
| 2 | Decision Tree | 92.46 |
| 1 | Gradient Boosting Trees | 86.61 |
| 3 | Cat Boost | 83.91 |
| 0 | KNN | 83.58 |

In regular accuracy, clearly the Decision Tree algorithm has the highest accuracy score.

**However**, again we can't not determine the model's performance based on regular accuracy, so here, Cross Validation is a **very useful technique for assessing the effectiveness of your model**, particularly in cases where we need to mitigate overfitting problem. It is also of use in determining the hyper parameters of your model, in the sense that which parameters will result in lowest test error.

The **result** of the Cross-validation Accuracy score with the fold 10 in all four models:



```
---Cross validation Accuracy Scores---
```

| | Model | Score |
|---|---|---|
| 3 | CatBoost | 81.32 |
| 1 | Gradient Boosting Trees | 80.65 |
| 2 | Decision Tree | 79.87 |
| 0 | KNN | 77.39 |

This gave an Accuracy of **81.32**.

So, the **CatBoost** model seems to be the **best model** (winner) among other three models. It is a simple and easy to use model and the cross-validation accuracy of 81.32 is a **pretty good score** for the Titanic dataset.

**Note:** 10-fold means in simple word, the model shuffles the entire data 10 times, and then evaluate it.

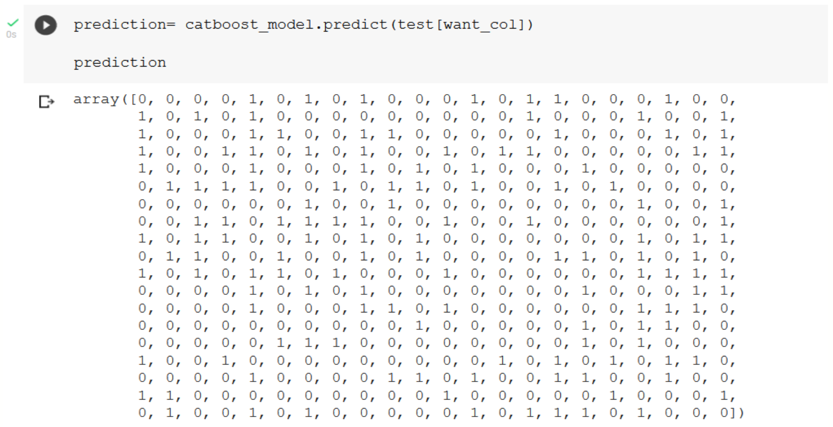
## Accuracy vs Cross-validation Accuracy:

When training a machine learning model, one of the main things that you want to avoid would be overfitting. This is when your model fits the training data well, but it isn't able to generalize and make accurate predictions for data that it hasn't seen before.

Hence, to find out if their model is overfitting, data scientists use a technique called **cross-validation**, where they split their data mainly into **two parts** - the training set, and the validation set. The training set is used to train the model, while the validation set is only used to evaluate the model's performance. The metrics on the training data indicates that how your model is progressing in terms of its training, on the other hand it's metrics on the validation set that let you get a measure of the quality of your model - how well it's able to make new predictions based on data it hasn't seen before.

## Model Predictions:

As the **CatBoost** model is the best model according to our above experiments. Thus, following is the **predictions** on the test data using this model.



```
prediction= catboost_model.predict(test[want_col])

prediction
```

```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0])
```

## Conclusion:

We started with the data exploration to analyze the data, checked about missing data, and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data preprocessing part, we computed missing values, converted features into numeric ones, grouped values into categories and created a few new features. Afterwards we started training 4 different machine learning models, picked one of them (CatBoost) and applied cross validation (10-fold) on it as there is a huge difference between regular accuracy score and cross validation accuracy score. Lastly, we looked at its predictions on testing data using CatBoost model.

Thus, we have **successfully completed** our entire project.

## References:

1) https://www.kaggle.com/c/titanic/data
2) https://scikit-learn.org/stable/modules/preprocessing.html
3) https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right