

Titanic Kaggle competition report

1. Introduction

Everybody knows about Titanic tragedy which happened in 1912 on April 15. The death of the Titanic is one of the most notorious shipwrecks in history. And, unfortunately, there were not enough boats for survival for everyone, and because of this, 1502 of 2224 passengers died. Based on data from this tragedy Kaggle created competition for building a ML model which would answer a lot of questions. Kaggle is a portal for Machine learning competitions. There are a lot of dataset for beginning our journey in machine learning and data analysis. The Titanic one is a beginner competition and the most popular one. We should build a ML model which would predict which passenger died or survived.

2. Background

Today we can observe how more and more tasks are solved with the help of machine learning algorithms. What is machine learning? Machine learning is a combination of statistical models and algorithms, which tries to perform specific functions without explicit instructions. In other words, machine learning uses our data to build a model for making some predictions without straightforward logic.

There are different types of machine learning: supervised, unsupervised, semi-supervised and reinforcement learning. They differ by how much human supervision is needed for training the algorithm. For our competition purposes we will use supervised classification machine learning algorithm.

Our datasets and competition is provided by Kaggle web site, this site is one of the most popular places for learning machine learning by practice. There are a lot of interesting and good competitions on Kaggle but one of the most popular competitions in Kaggle for beginner is Titanic. In this competition we have a Dataset with passengers who was on Titanic and we should predict if those people survived or not. There are a lot purposes for this competition. One of them is why some people died more, what categories of people had more chance to survive, all of them is possible by data analysis and machine learning.

Before doing machine learning we should do data analysis to see correlations between people's age, sex, class type and etc. The main time consuming and important part of machine learning is data analysis process. Good data analysis is 70% of the success. That's why we will begin by doing data analysis in the next section.

3. Data analysis

Before training our ML algorithms and making predictions, let's look to the data and do some data analysis. We are given two datasets in this competition – training and testing sets. Training dataset consist of several features. The main difference between the training and testing dataset is absence of the target column. Training dataset is given for making predictions, after training our ML model. For training our machine learning algorithm we will use training dataset.

All features provided by our datasets are:

- Passenger Id: it is an id given for each passenger on the boat
- Pclass: The passenger's class. There are three values: 1,2,3 (first, second and third class)
- Name: Names of the passengers
- Sex: Sex of the passengers
- Age: Ages of the passengers
- SibSp: The number of siblings or spouses traveling with passenger
- Parch: Number of parents and children traveling with the passenger
- Ticket: Number of the ticket
- Fare: The ticket fare
- Cabin: The number of the cabin
- Embarked: Area from which the people was embarked. There are three possible values S,C,Q. (C=Cherbourg, Q=Queenstown, S=Southampton)

We will make our data analysis on Jupiter notebook and we will use Python programming language. After running our Jupiter notebook, we should import some essential libraries for working with datasets and for making some data analysis. We would use Matplotlib, Seaborn – for visualizing our data sets and correlations between features and Pandas – for easier working with datasets.(Image 1)

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Image 1. Importing libraries.

After importing our libraries we can now read our training dataset and convert them into pandas DataFrame and print first five rows of our new DataFrame. (Image 2)

```
# Reading training data
train = pd.read_csv('train.csv')
train.head()
```

	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

Image 2. Reading training data and printing first 5 rows.

We will use the same code for reading testing dataset too. Now we have DataFrames, which we should explore. Exploring data is the most important part of doing machine learning, it is called acquiring domain knowledge. In our case it is Titanic tragedy dataset and we should discover what features are important in predicting the survival of the passengers.

Our training DataFrame has 891 rows and 12 columns, our testing DataFrame has 418 rows and 11 columns, it is missing 'Survived' column. We can print them using pandas. And of course, we can see that our DataFrame has missing values in Age, Cabin and Embarked columns. (Image 3-4)

```
#Trying to see how much data we have and all Features we have. And we can decide that there are missing Age and Cabin columns
print(df_train.shape)
df_train.count()
```

```
(891, 12)
PassengerId    891
Survived        891
Pclass          891
Name            891
Sex             891
Age             714
SibSp           891
Parch           891
Ticket          891
Fare            891
Cabin           204
Embarked        889
dtype: int64
```

Image 3. Printing shape and quantity of each column of the training DataFrame.

```
#Trying to see how much data we have and all Features we have. And we can decide that there are missing Age and Cabin d
print(df_test.shape)
df_test.count()

(418, 11)

PassengerId    418
Pclass         418
Name           418
Sex            418
Age           332
SibSp          418
Parch          418
Ticket         418
Fare           417
Cabin          91
Embarked       418
dtype: int64
```

Image 4. Printing shape and quantity of each column of the testing DataFrame.

To see visually missing columns, we can use 'Missingno' package. First we install this package using pip install. (Image 5)

```
pip install missingno # install package for clearly see what data we miss

Requirement already satisfied: missingno in /Users/dave/anaconda3/lib/python3.7/site-packages (0.4.2)
Requirement already satisfied: matplotlib in /Users/dave/anaconda3/lib/python3.7/site-packages (from missingno) (3.1.0)
Requirement already satisfied: numpy in /Users/dave/anaconda3/lib/python3.7/site-packages (from missingno) (1.16.4)
Requirement already satisfied: scipy in /Users/dave/anaconda3/lib/python3.7/site-packages (from missingno) (1.3.0)
Requirement already satisfied: seaborn in /Users/dave/anaconda3/lib/python3.7/site-packages (from missingno) (0.9.0)
Requirement already satisfied: cycler>=0.10 in /Users/dave/anaconda3/lib/python3.7/site-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/dave/anaconda3/lib/python3.7/site-packages (from matplotlib->missingno) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /Users/dave/anaconda3/lib/python3.7/site-packages (from matplotlib->missingno) (2.4.0)
Requirement already satisfied: python-dateutil>=2.1 in /Users/dave/anaconda3/lib/python3.7/site-packages (from matplotlib->missingno) (2.8.0)
Requirement already satisfied: pandas>=0.15.2 in /Users/dave/anaconda3/lib/python3.7/site-packages (from seaborn->missingno) (0.24.2)
Requirement already satisfied: six in /Users/dave/anaconda3/lib/python3.7/site-packages (from cycler>=0.10->matplotlib->missingno) (1.12.0)
Requirement already satisfied: setuptools in /Users/dave/anaconda3/lib/python3.7/site-packages (from kiwisolver>=1.0.1->matplotlib->missingno) (41.0.1)
Requirement already satisfied: pytz>=2011k in /Users/dave/anaconda3/lib/python3.7/site-packages (from pandas>=0.15.2->seaborn->missingno) (2019.1)
Note: you may need to restart the kernel to use updated packages.
```

Image 5. Installing 'Missingno' package.

Now we can import library and plot the DataFrame to see the columns in which we miss the data. As we can see on image 6, it is Age, Cabin and Embarked columns

```
import missingno
missingno.matrix(df_train, figsize = (30,15))
# plot the data to see what data we miss. IT is clear that we miss some age values and Cabin.
#And some in Embarked
```

<matplotlib.axes._subplots.AxesSubplot at 0x1a23c64278>

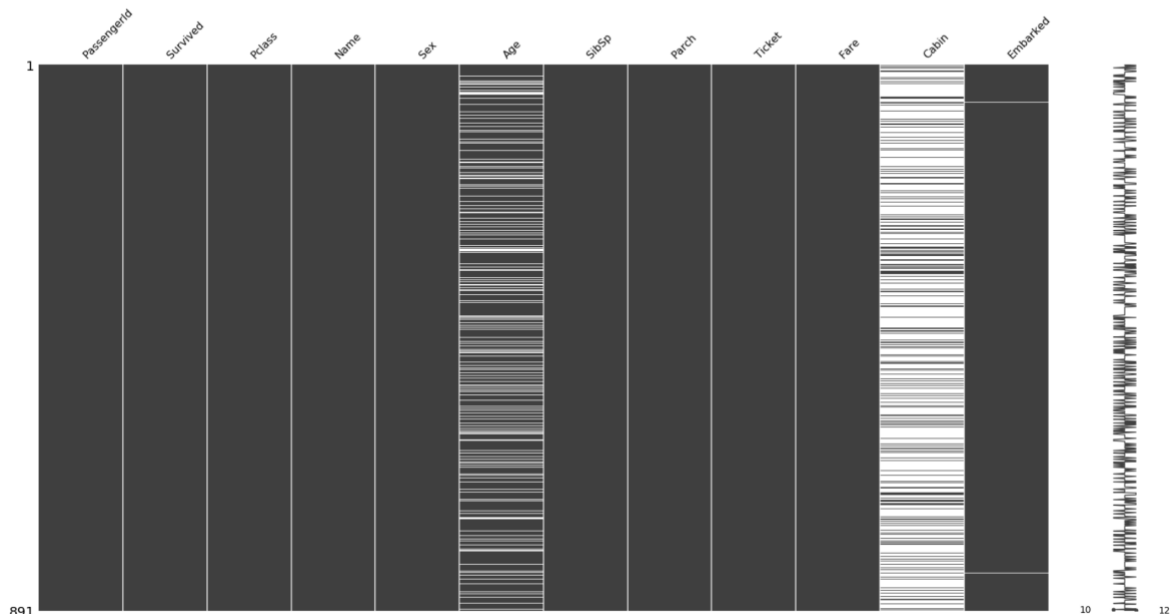


Image 6. Importing 'Missingno' library and plotting DataFrame

We can now explore each feature of our DataFrame. But first let's explore our target column.

3.1 Survived feature

When we printed the head of our DataFrame we see that 'Survived' feature consists of two values 0 and 1, which means 0 – died, 1 – survived. Let's count both values in our data.

```
df_train['Survived'].value_counts() # Values of survived and died people
```

```
0    549
1    342
Name: Survived, dtype: int64
```

Image 7. Values of survived and died people in training DataFrame

We can see that 549 people died and only 342 people from 891 survived. To see clearly, we can plot the column by using seaborn library. (Image 8)

```
sns.countplot(y='Survived', data=df_train) # Plotting survived and died people quantity
<matplotlib.axes._subplots.AxesSubplot at 0x1a28c04f60>
```

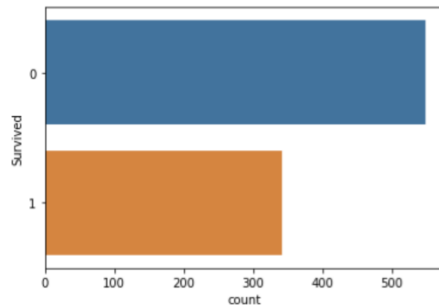


Image 8. Plotting survived and died people quantity.

3.2 Age Feature

First of all, let's see how much missing values our 'Age' feature has. To see that we use built-in `isnull()` method and sum all missing values. We can see on image 9 that it has 177 missing values and we get rid of them when we do machine learning.

```
df_train.Age.isnull().sum() # We know that AGE feature has 177 missing values,
#we should decide what to do with missing values when we do machine learning
```

177

Image 9. Age column missing values quantity.

Next, we scatter plot 'Age' column and 'Survived' columns. This plot will show us that despite the age, there were deaths in any age category, but we can clearly see that the chance for surviving people who are older than 40 is much less. (Image 10)

```
plt.scatter(df_train['Survived'], df_train.Age, alpha=0.1) # Scatter plot to see Age vs Survived.
#Use alpha to see better in ending area
plt.title("Age and Survived")
plt.show()
```

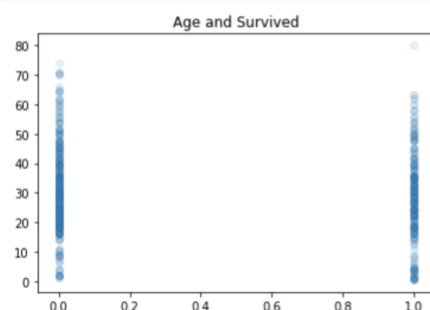


Image 10. Scatter plot Age vs Survived

3.3 Sex Feature

One of the most features in predicting who survived is 'Sex' feature. First let's see quantity of male and female in the DataFrame. By using Value_counts() method we can see that there was 577 – male and 314 – female in our DataFrame. (Image 11)

```
df_train['Sex'].value_counts() # printing quantity of male and female
```

```
male      577  
female    314  
Name: Sex, dtype: int64
```

Image 11. Printing quantity of male and female.

We can see that there was more male than female, but it is not so important. Let's plot the bar with percentile of survived and died male and female. (Image 12-13)

```
# Plotting survived and died men bar plot  
df_train.Survived[df_train['Sex'] == "male"].value_counts(normalize=True).plot(kind='bar',color="#000000")  
plt.title("Men Survived PLOT")  
plt.show()
```

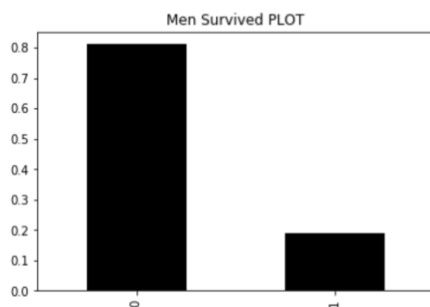


Image 12. Survived men bar plot

```
# Plotting survived and died women  
#But As we can see Men Survived PLOT and Female Survived Plot.  
#Gender has a main role for surviving. 70% of women survived! But almost 80% of men died.  
df_train.Survived[df_train['Sex'] == "female"].value_counts(normalize=True).plot(kind='bar',color="#fa0000")  
plt.title("Women Survived PLOT")  
plt.show()
```

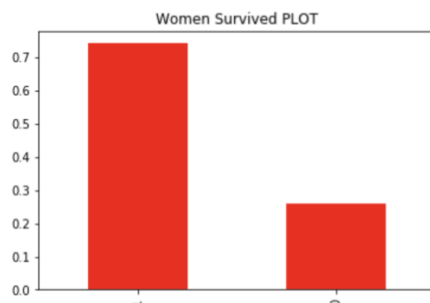


Image 13. Survived woman bar plot

On image 12 we plot bar with male survived data and we can see that only 20% of men survived in the tragedy. The image 13 is contrary of the first image. We can see that approximately 25% of women died and more than 70% survived in the tragedy. And we can conclude that gender plays main role for surviving. For better visualization we can plot the bar plot and visually see that.

```
#compare male and female procent of survival
df_train.Sex[df_train['Survived'] == 1].value_counts(normalize=True).plot(kind='bar',color=["#fa0000","k"])
plt.title("Survived procent")
plt.show()
```

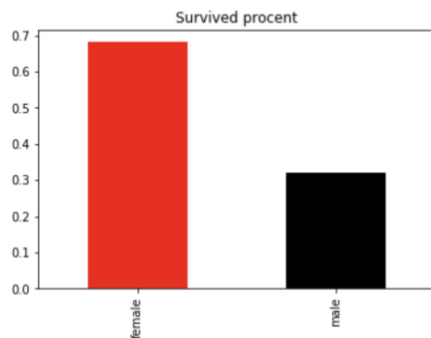


Image 14. Compare survived man and woman

3.4 Class Feature

Titanic had three types of class on board. By printing values of 'Pclass' column we can see that 491 people was in the third class, 216 people in second class and only 184 people in the first class.(Image 15)

```
df_train['Pclass'].value_counts()
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

Image 15. Counting values by classes.

Now we have a question, 'Does class affected survival?', after plotting class column, we can clearly see that class exactly affected survival. Most people survived from first class, half of the second class died and the largest amount of died people are from third class. (Image 16)


```
# I want to see how class helped to survive.
#And after plotting we can clearly see that most of 1st class Survived besides 3rd class.

for x in [1,2,3]:
    df_train[df_train.Pclass == x].plot(kind='kde')
plt.title("Class of Survived people")
plt.legend(("1st class","2nd class","3rd class"))
plt.show()
```

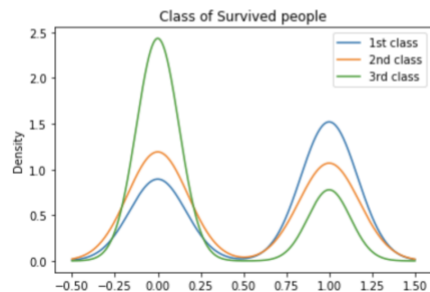


Image 16. Plot of class column by survival

3.5 SibSp Feature

SibSp column shows the number of siblings or spouses. The largest amount of people was alone (608 people), 209 people were with 1 sibling/spouse, 28 people with 2 sibling/spouse, 18 people with 4 sibling/spouse, 16 people with 3 sibling/spouse, 7 people with 8 sibling/spouse and 5 people with 5 sibling/spouse. (Image 17-18)

```
df_train['SibSp'].value_counts()

0      608
1      209
2        28
4        18
3        16
8         7
5         5
Name: SibSp, dtype: int64
```

Image 17. Values of SibSp column

```
#Plotting SibSp column
df_train['SibSp'].value_counts(normalize=True).plot(kind='bar') # SibSp PLOT
plt.title("Siblings/Spouse PLOT")
plt.show()
```

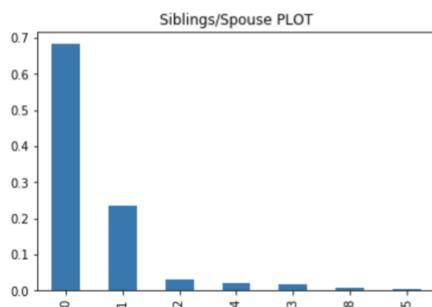


Image 18. Plotting SibSp column

Now we can factor plot each value by survival rate and we can see that people with 1 sibling/spouse survived more than all others. (Image 19)



Image 19. FactorPlot SibSp column by survival rate.

3.6 Embarked Feature

Embarked column has 3 values: C=Cherbourg, Q=Queenstown, S=Southampton. Most of people was embarked in Southampton and least was from Queenstown. (Image 20-21)

```
df_train['Embarked'].value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

Image 20. Values of Embarked column

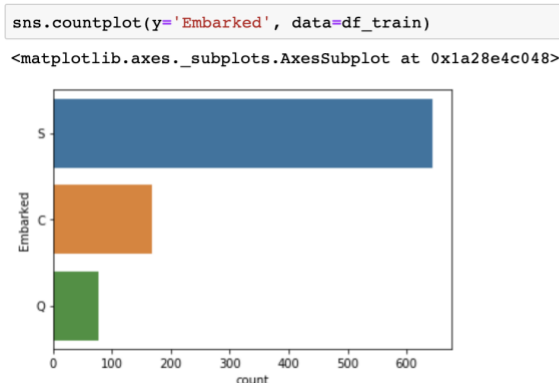


Image 21. Visualizing values of Embarked column

Nothing special, but when we plot 'Embarked' column values by survival, we can see that more than half people embarked from Queenstown survived.(Image 22)

```
#Plotting Embarked column values by survival
sns.catplot(x='Survived', col='Embarked', kind='count', data=df_train)

<seaborn.axisgrid.FacetGrid at 0x1a25d1c518>
```

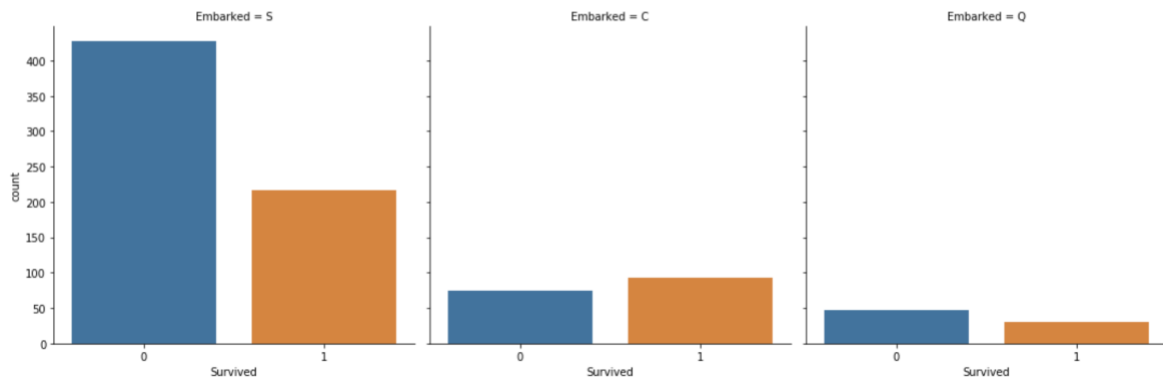


Image 22. Plotting Embarked column values by survival

After doing all this data analysis we can make conclusion that, most important factors for surviving was 'Sex'. It was seen from data than most of women survived in this tragedy. The second most important factor for survival was 'Class'. The chances of surviving in first class are much higher than second or third. And of course we saw from movie that woman with children had more chance for survival. Other features have some impacts for survival too. Next step is to make some predictions using our data and our data analysis.

4. Doing Machine Learning

Now we explored our dataset and know which features has big impact for passenger survival and can start doing machine learning. Let's create new iPython file and import all libraries which we would use.(Image 23)

```
# importing libraries which we will use in future
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
from sklearn import linear_model
from sklearn import ensemble
```

Image 23. Importing libraries

After importing libraries we can now read our datasets by using pandas and drop unused columns in both training and testing datasets.(Image 24-25)

```
# Reading training data
train = pd.read_csv('train.csv')
train.head()
```

	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

```
# Reading our testing data
test = pd.read_csv('test.csv')
test.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Image 24. Reading datasets using pandas

```
# Dropping unused columns
train.drop(columns=['Name', 'Cabin', 'Fare', 'Ticket'], inplace=True)
test.drop(columns=['Name', 'Cabin', 'Ticket', 'Fare'], inplace=True)
```

Image 25. Dropping unused columns by using pandas tools

We saw in exploratory data analysis that we have some missing values and for doing machine learning we should get rid of them.(Image 26)

```
# Checking DataFrame for null values
train.isnull().sum()
```

```
PassengerId    0
Survived        0
Pclass          0
Sex             0
Age            177
SibSp           0
Parch           0
Embarked        2
dtype: int64
```

Image 26. Checking for missing values

We should clean our data from missing values and beside that we should change string values to numerical. Missing values from 'Age' column we will fill with median value of Age' column. 'Sex' column values we will change to – 0 – male, 1 – female. 'Embarked' column values we will change to – 0 – S, 1 – Q, 2 – Q. For all of this we will create clean_data function, because we should use it again for testing data. After all of this we will drop NaN values by using dropna method.(Image 27-28)

```
# Creating function for doing some major cleanings. Changing embarked coloumn values to numeric , filling Age coloumn
# And last dropping all null values.
embarked_port = {"S": 0, "C": 1, "Q": 2}
def clean_data(data):
    data["Age"] = data["Age"].fillna(data["Age"].dropna().median())
    data.loc[data["Sex"] == "male", "Sex"] = 0
    data.loc[data["Sex"] == "female", "Sex"] = 1
    data["Embarked"] = data["Embarked"].map(embarked_port)
train.dropna(inplace=True)
```

Image 27. Creating cleaning function

```
# Using our data cleaning function
clean_data(train)
clean_data(test)
```

Image 28. Applying cleaning function

We can check out our training data and see now that we don't have any missing values.(Image 29)

```
# Re-checking dataframe for null values
train.isnull().sum()

PassengerId    0
Survived        0
Pclass          0
Sex             0
Age             0
SibSp           0
Parch           0
Embarked        0
dtype: int64
```

Image 29. Checking for missing values after cleaning the data

We can start our machine learning. Our data prepared for doing machine learning. First of all, let's create our variables with features and our target for training machine learning algorithm.(Image 30)

```
# Picking up our features for training our model
# And picking our target which should be predicted
feature_names = train[['Pclass', 'Age', 'Sex', 'SibSp', 'Embarked', 'Parch']].values
target = train['Survived'].values
```

Image 30. Picking our features and target values

This is classification problem and for solving it we should use classification machine learning algorithms. First I picked the easiest linear machine learning algorithm "Logistic Regression". SikitLearn library which we imported before(Image 23) has this algorithm predefined, we just declare it and fit our training data to this machine learning algorithm. This algorithm showed 0.797% score(Image 31)

```
# Training machine learning algorithm. Doing logistic regression
# Fitting our values
# printing score of our algorithm
logistic = linear_model.LogisticRegression()
logistic.fit(feature_names, target)
print(logistic.score(feature_names, target))
```

```
0.797752808988764
```

Image 31. Training machine learning algorithm and printing its score

If we use `cross_val_score` function from SikitLearn, we can see cross validation scores from our first algorithm and it is clearly seen that it perform better scores and by using `max()` method we can see maxium score what we get from this model, it showed 0.861%. (Image 32)

```
# Cross validation of our score
scores = model_selection.cross_val_score(logistic, feature_names, target, scoring='accuracy', cv=10)
print(scores)
print(scores.max())
```

```
[0.80555556 0.73611111 0.79166667 0.86111111 0.73239437 0.78873239
 0.77464789 0.77464789 0.82857143 0.8
 0.8611111111111112]
```

Image 32. Cross validation scores and printing max score

We can use another classification machine learning algorithm for comparing scores. We will use Random Forest Classifier, this algorithm uses another approach, it creates decision trees and picks random data. SikitLearn has implemented Random Forest Classifier algorithm. We will predefine `max_depth` of our random forest to 7, minimum split to 4. And we will split our training data into 80% training and 20% testing data, to avoid overtraing.(Image 33-34)

```
# Using Random Forest classifier
# Splitting our training dataframe into splits for better performance and to avoid overtraining
# Printing our classifier score
forest = ensemble.RandomForestClassifier(
    max_depth = 7,
    min_samples_split = 4,
    n_estimators = 1000,
    random_state = 1,
    n_jobs = -1
)

X_train,X_test,y_train,y_test = model_selection.train_test_split(feature_names,target,test_size=0.2,random_state=1)
forest = forest.fit(X_train, y_train)
print(forest.score(feature_names, target))
```

```
0.8581460674157303
```

Image 33. Training Random Forest Classifier and printing score.

```
# printing cross validation scores
scores = model_selection.cross_val_score(forest, feature_names, target, scoring='accuracy', cv=10)
print(scores)
print(scores.max())
```

```
[0.73611111 0.75          0.76388889 0.90277778 0.8028169  0.78873239
 0.81690141 0.76056338 0.88571429 0.84285714]
0.9027777777777778
```

Image 34. Cross validation score and printing max score.

Random Forest Classifier showed better results. For making prediction we should define features for prediction and use our trained model for prediction.(Image 35)

```
# defining our test features
# Testing our Random forest classifier. Making a prediction
test_features_forest = test[["Pclass", "Age", "Sex", "SibSp", "Parch", "Embarked"]].values
prediction_forest = forest.predict(test_features_forest)
```

Image 35. Defining test features and making predictions.

After making predictions, the last step is to save our predictions into csv file for making submission to the Kaggle competition. (Image 36)

```
# Creating a function for saving our predictions in a new file
def write_prediction(prediction, name):
    PassengerId = np.array(test["PassengerId"]).astype(int)
    solution = pd.DataFrame(prediction, PassengerId, columns = ["Survived"])
    solution.to_csv(name, index_label = ["PassengerId"])
```

```
# Saving our predictions
write_prediction(prediction_forest, "submission.csv")
```

Image 36. Creating function and saving our predictions

5. Submission to the Kaggle competition

We predicted and saved our predictions into 'submission.csv' file. Now we should submit it to the Kaggle competition. In the competition page we have 'Submit Predictions' button. We should press it and we will get into submission mode. Submission has two steps. First is to upload our predictions file and write a brief description. (Image 37-38)

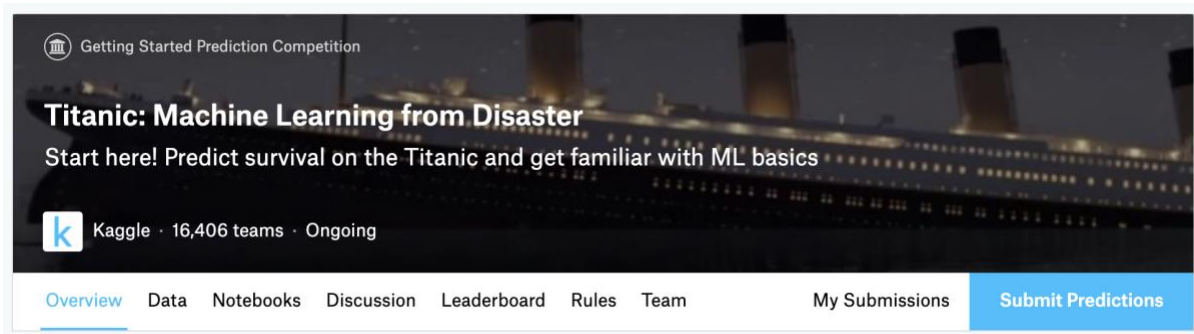


Image 37. Titanic Kaggle competition page








1202	titanium_hare		0.80382	20	9d
1203	Holiday Green		0.80382	3	5d
1204	WhiteGreen		0.80382	43	8d
1205	Wellington Silva		0.80382	19	8d
1206	Davron Karimov		0.80382	10	5m
Your Best Entry ↑ Your submission scored 0.80382, which is not an improvement of your best score. Keep trying!					
1207	javgeros		0.80382	10	7d
1208	Wenxiaofeng_1		0.80382	17	8d

Image 40. Leaderboard of the Titanic competition

6. Results and Feature improvement suggestions

After submitting my predictions, I get 0.80382% of accuracy. Now I am on the 1206 place on the leaderboard. I used two different classification algorithms. One of them is logistic regression. This algorithm just separates our dataset into two categories by using one line. Logistic regression creates a sigmoid function between 0 and 1, and has a boundary for categorization. And one of the disadvantages of logistic regression is that it attempts to predict outcomes based on a set of independent variables, but if we include the wrong independent variables, the model will have little to no predictive value. Another algorithm that I used is Random Forest. Random Forest is an ensemble classifier which consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. That's why Random Forest has several advantages over Logistic Regression, and it showed for more than 10% better accuracy on competition than Logistic Regression. My model had some issues and it can be modified to get better accuracy. There are many techniques for improving our machine learning model. One of the easiest ways is to check feature importance and add features that we dropped or create new features by combining two or three features in one. Another approach is to create categories in our dataset, for example split age into 4-5 categories. Of course, there is a lot of space for future improvements, like I said feature engineering, identify and remove noisy data in some columns, do more data analysis by plotting features against each other. We changed the Embarked column values to numbers, which can lead to incorrect categorization of our passengers, and instead we can create new columns with each port and set the value to 0 and 1. This can lead for a decrease of error percent of our algorithm. One more approach that can improve our score is extensive hyperparameter

tuning for machine learning algorithms or try to use another classification machine learning algorithm.

7. References

1. Aurlien Gron (2017) . Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 1st ed.
2. Laura Igual , Santi Segui (2017). Introduction to Data Science. A Python approach to concepts, techniques and applications.
3. Wes McKinney (2017). Python for Data Analysis. Data wrangling with pandas, numpy, and IPython. Second edition.
4. Sebastian Raschka, Vahid Mirjalili(2017). Python Machine Learning. Machine learning and deep learning with Python, scikit-learn and TensorFlow.
5. Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall.
6. Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). An Introduction to Statistical Learning.
7. Alpaydin, Ethem (2010). Introduction to Machine Learning.
8. Shi, T., Horvath, S. (2006). "Unsupervised Learning with Random Forest Predictors". Journal of Computational and Graphical Statistics.
9. Kaitlin Kirasich,Trace Smith,Bivin Sadler(2018).Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets.