CENG 3521
DATA MINING
TITANIC PROJECT

Fatma Karadağ 170709061
Gizem PESEN 170709050

https://github.com/Titanic-Project.

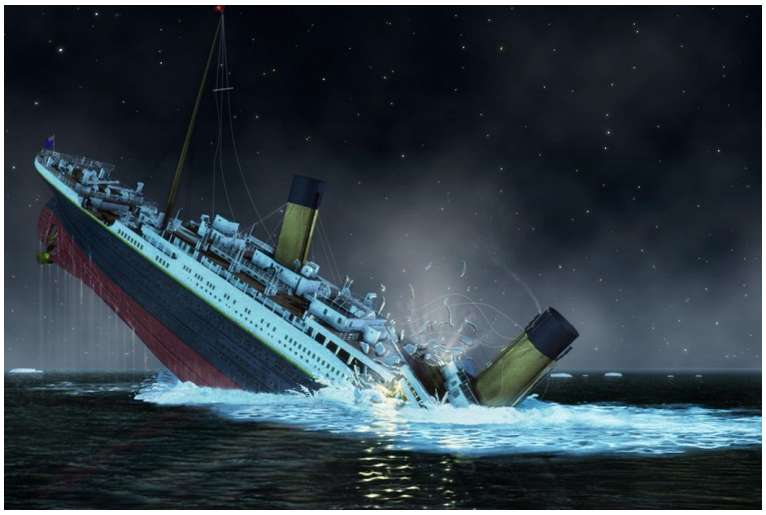
Thursday 21st January, 2021

Abstract

In this homework, we tried to calculate the survival probabilities of people in the titanic ship disaster by using the classification types we learned in the data mining course. We did some studies on our data by producing sub-problems ourselves.In short, we wanted to show how data mining reveals the results of a disaster.

Contents

1 Introduction



RMS (Royal Mail Ship) The sinking of the Titanic is one of the most painful shipwrecks in history. On April 15, 1912, on its maiden voyage, the Titanic crashed into an iceberg, and 2224 passengers and 1502 crew were killed. This tragic disaster shocked the international community and led to better safety regulations for ships. One of the reasons the accident caused such a loss of life was that there were not enough lifeboats for the passengers and crew. Some classes of people (such as women, children, and upper classes) were more likely to survive than others, although there were certain luck elements to survive. In this assignment, we tried to complete the analysis of what kind of people could survive in this struggle.

2 Data

2.1 Test dataset

Let's start by getting to know the datasets presented to us.We will use the test set (test.csv) to see how well our model performs on test data. We will try to predict survivors. There are 11 columns in the test set and the names of the columns are as follows: Passengerid, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked

2.2 Train dataset

We will use the training set (train.csv) to build our machine learning model. There are 12 columns in the train set and the names of the columns are as follows:
Passengerid, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked

2.2.1 The classes of the data

Survived: survival (0 = No, 1 = Yes)
Pclass: ticket class (1 = 1., 2 = 2., 3 = 3.)
Sex: gender
Sibsp: Number of siblings / spouses on the Titanic
Parch: Number of parents / children on Titanic
Ticket: ticket number
Mouse: charge
Cabin: cabin number
Embarked: embarked port

* After opening the file in **Jupyter notebook**, we imported the necessary libraries for data analysis.

2.3 Define test and training sets with pandas library

2.3.1 Test csv file

Listing 1: test

```
test_dataset = pd.read_csv(r'C:\Users\pesen\Desktop\new\test.csv') # for Fatma
    (r'C:\Users\Fatma\Desktop\new\test.csv')
```

2.3.2 Train csv file

Listing 2: train

```
train_dataset = pd.read_csv(r'C:\Users\pesen\Desktop\new\train.csv') #for Fatma
    (r'C:\Users\Fatma\Desktop\new\train.csv')
```

* After defining, we look at which columns we have in our data sets.
* We Printed Columns and Rows for Training and Testing.
* We looked at the total number of passengers in the two data sets.
* We looked at the survival rate.
* We looked at the first 5 lines in the training set.

3 Data Visualition

For this informations you can visit our all project from ;
urlhttps://github.com/Titanic-Project.

4 Feature Engineering

4.1 Age

We do not want to delete all rows with missing age values, so we will replace the missing ones. As you can see, we have a sloping distribution according to age, and the median should be a good choice for substitution.

One thesis was that the average age for passenger classes was different. Professional advancement often comes with increasing age and experience. Therefore, people with higher socio-economic status are on average older. If we break down by gender, we see that there is still a difference as women are generally younger. In a final step, I checked the number of cases to make sure there are still enough cases in each category. We will use these median values to replace the shortcomings.

Listing 3: age

```
#replace the missings values with the medians of each group
df_all['Age']= df_all.groupby(['Pclass','Sex'])['Age'].apply(lambda
    x:x.fillna(x.median()))
```

4.2 Fare

Listing 4: fare

```
df_all.loc[df_all['Fare'].isnull()]
```

We have one missing fee value in the entire data set. Mr. Thomas was in passenger class 3, traveled alone and flew to Southhampton. We will take other cases from people in this category and replace the missing Fee with the median of this group.

Listing 5: loc cases which are similar to Mr.Thomas and use the median of fare to replace the missing for his data set

```
mr_thomas=df_all.loc[(df_all['Pclass']==3)&(df_all['SibSp']==0)&(df_all['Embarked']=='S')]['Fare'].median()
print(mr_thomas)
```

4.3 Cabin

There are too many missing values, but we must use the cabin variable because it can be an important determinant. The cabins of the first class were on decks A, B or C, a mixture of these were on D or E and the third class was mainly on f or g. We can recognize the deck from the first letter.

Listing 6: cabin missing values

```
#keep all first letters of cabin in a new variable and use "M" for each missing
df_all['Deck'] = df_all['Cabin'].apply(lambda s: s[0] if pd.notnull(s) else 'M' )
```

4.4 Embarked

There are only two shortcomings to start with. As we have already tried for the wage case, we can look at similar situations to replace the missing value. It is logical that people who paid a similar amount, who also had a 1st class ticket and were on the same deck got on board from the same location.

Listing 7: embarked

```
df_all.loc[df_all['Embarked'].isnull(),'Embarked'] = 'S'
```

4.5 Result

We've filled in all missing values in our dataset and haven't left a column yet. We used statistical methods for age and wage, created a new category for the cabin, and did some research for shipboard losses.

4.6 Techniques we will use so far:

Grouping continuous variables (ex: Age) - Create new attributes from existing variables (ex: Title) - Tag coding for non-numeric properties (ex: Gender) - A hot coding for categorical features (ex: Pclass)

* After this process, we obtain information about the data using the .info () method.

Listing 8: get-dummies

```
passengers = []
columnss = ['Pclass', 'Sex', 'Embarked']
for columns in columns:
  passengers.append (pd.get-dummies (train-dataset [columns]))
```

Thanks to the code above, we have completed separate transformations of our attributes named Pclass, Sex and Embarked in a list called passengers. As a result of the transformation, the attributes started to consist of column headings of the categorical classes they contained. To give an example, if the x row has the property of the y column, the matching cell value becomes 1, the others have the value 0.

5 Data Cleaning

5.1 Check missing columns

* we should examine the missing columns to clean it.

Listing 9: missing columns

```
print("Missings in the train data: ")
display(train_dataset.isnull().sum())

print("Missings in the test data: ")
display(test_dataset.isnull().sum())
```

5.2 Clean gender

* We used the concat method, which is a method of Pandas library, to obtain a single DataFrame. After using the method, we printed our variable on the console in order to view the result. The axis we mentioned here means "from the xth index value".

Listing 10: concat passengers

```
passengers = pd.concat (passengers, axis = 1)
```

* We have two attributes named female and male. One takes a value of 1 while the other takes a value of 0. We can delete one of the two attributes and continue with one.

Listing 11: drop male

```
passengers = passengers.drop (['male'], axis = 1)
```

5.3 Drop the useless and missing columns

* After this process, we were able to combine the attributes that we converted with the attributes that we did not transform. However, since there is also a variable named train-dataset that we will use in this merging process, we will extract the attribute from which we transform. Again, I will use the **concat and drop methods** again for these operations.

Listing 12: drop the useless and missing columns

```
train-dataset = pd.concat ((train-dataset, passengers), axis = 1)

train-dataset = train-dataset.drop (('Pclass', 'Sex', 'Embarked'), axis = 1)
```

* X still contained the Survived attribute with 1 column header in it. We subtracted this from input variables.

Listing 13: concat

```
X = np.delete (X, 1, axis = 1)
```

* Then we used my dataset from the scikit library and split it as training 70 percent and test 30 percent with train-test-split.

* Thanks to the code block below, we trained our train data with the **fit () method.** Then, we put our success rate into the variable named score with our test data that we previously shredded.

test size is 30 percent and train size 70 percent

6 Modeling and Predictions

6.1 Classifiers

6.1.1 Train and test

Now we will get our own Survived predictions using the training data set. Then we will get a truth table score by comparing it with the actual Survived values.

Listing 14: train and test

```
X = train_dataset.values
Y = train_dataset['Survived'].values

X = np.delete(X,1,axis=1)

X_train, X_test, y_train, y_test=train_test_split(X,Y,test_size=0.3, random_state=0)
    #test size 30 atadk train test de 70 oldu
```

6.1.2 DecisionTreeClassifier

Listing 15: DecisionTreeClassifier

```
siniflama = tree.DecisionTreeClassifier(max-depth=5)
siniflama.fit(X-train,y-train)
skor = siniflama.score(X-test,y-test)
print("Score: ",skor)
tahminler = siniflama.predict(X)
as-egitim = accuracy-score(tahminler, Y)
print("Doruluk tablosu skoru: ", as-egitim)
```

We obtained the accuracy score of the application above, but the accuracy score alone cannot be a benchmark for success. We used a confusion matrix to look at other criteria (error rate, sensitivity, etc.). For this, we used the **crosstab () method** from the pandas library.

Listing 16: modeling

```
confusion-matrix = pd.crosstab(Y, tahminler, rownames=['Gerek'], colnames=['Tahmin'])
print (confusion-matrix)
```

6.1.3 RandomForestClassifier

Listing 17: RandomForestClassifier

```
model = RandomForestClassifier(criterion =
    'gini',n_estimators=1750,max_depth=7,min_samples_split =6, min_samples_leaf = 6,
    max_features = 'auto', oob_score= True, random_state=42, n_jobs=-1,verbose =1)

model.fit(X_train, y_train)
predictions = model.predict(X_test)

score = model.score(X_test, y_test)
```