Titanic

August 15, 2019

1 Machine Learning Engineer Nanodegree

1.1 Introduction and Foundations

1.2 Project 0: Titanic Survival Exploration

This porject is for Udacity Machine Learning Engineer Nanodegree and It is modified, solved and imporved by Javad Ebadi

RMS Titanic was a British passenger liner that sank in the North Atlantic Ocean in 1912 after the ship struck an iceberg during her maiden voyage from Southampton to New York City. Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making it one of modern history's deadliest peacetime commercial marine disasters. RMS Titanic was the largest ship afloat at the time she entered service and was the second of three Olympic-class ocean liners operated by the White Star Line. She was built by the Harland and Wolff shipyard in Belfast. Thomas Andrews, chief naval architect of the shipyard at the time, died in the disaster. Reference

In this notebook, we are going to find the features that can predict the survival of a passenger.

There is a famous movie about Titanic and you may be familira with that. If you have watche then this video is nostalgic for you (click to paly from YouTube):

2 Getting, cleaning and pre-processing data

To begin working with the RMS Titanic passenger data, we'll first need to `import` the packages.

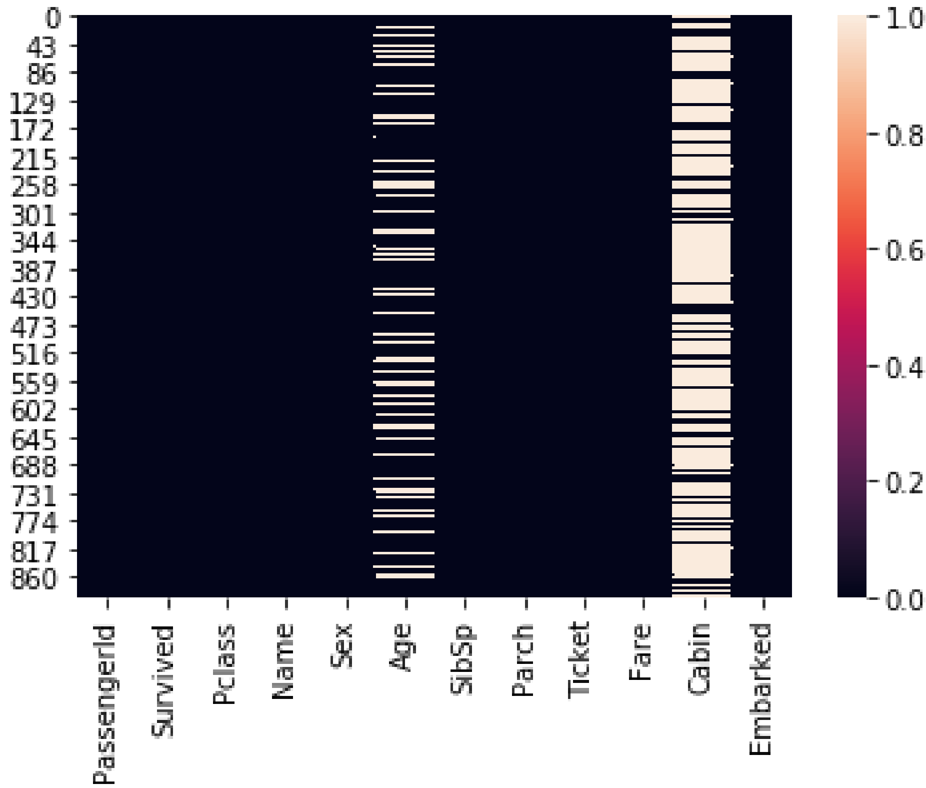
[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Read the dataset
dataset_path = "./titanic_data.csv"
df = pd.read_csv(dataset_path)
df.head()
```

From a sample of the RMS Titanic data, we can see the various features present for each passenger on the ship: - **Survived**: Outcome of survival (0 = No; 1 = Yes) - **Pclass**: Socio-economic class (1 = Upper class; 2 = Middle class; 3 = Lower class) - **Name**: Name of passenger - **Sex**: Sex of the passenger - **Age**: Age of the passenger (Some entries contain `NaN`) - **SibSp**: Number of siblings and spouses of the passenger aboard - **Parch**: Number of parents and children of the passenger aboard - **Ticket**: Ticket number of the passenger - **Fare**: Fare paid by the passenger - **Cabin** Cabin number of the passenger (Some entries contain `NaN`) - **Embarked**: Port of embarkation of the passenger (C = Cherbourg; Q = Queenstown; S = Southampton)

Missing valuse need to be handled in every data analysis problem. We use `seaborn`'s `heatmap` to investigate the missing values in our dataset.

[2]:
```
# looking for missing values in the dataset
sns.heatmap(df.isna())
```

[2]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f26326b67b8>
```



The Cabin and Age column have missing values. In particular the Cabin column has is mainly cotains missing values than non-missing. Thus we have to remove that column from out dataset for now. In addition, we drop the record with missing Age. The reason, we don't drop the Age column is that we think Age is an important parameter in predicting survival. We will re-investiage this prior later.

[3]:
```
df = df.drop(columns=['Cabin'])
df.head()
```

[3]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
```

```
                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Embarked
0      0         A/5 21171   7.2500        S
1      0          PC 17599  71.2833        C
2      0  STON/O2. 3101282   7.9250        S
3      0            113803  53.1000        S
4      0            373450   8.0500        S
```

[4]:
```
df = df[df['Age'].isnull() == False]
```

2.0.1 Categories to numerics

Some columns such as Sex and Embarked are in string format and they have only handful of values. Therefore, to use information of them in the analysis we need to convert them to numerics.

[5]:
```
df['Sex'].replace(['female','male'],[0,1],inplace=True)
df['Embarked'].replace(['S','C','Q'],[-1,0,1],inplace=True)
df.head()
```

[5]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Sex   Age  SibSp  Parch  \
0                            Braund, Mr. Owen Harris    1  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    0  38.0      1      0
2                             Heikkinen, Miss. Laina    0  26.0      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    0  35.0      1      0
4                           Allen, Mr. William Henry    1  35.0      0      0

             Ticket     Fare  Embarked
0         A/5 21171   7.2500      -1.0
1          PC 17599  71.2833       0.0
2  STON/O2. 3101282   7.9250      -1.0
3            113803  53.1000      -1.0
4            373450   8.0500      -1.0
```

Another thing we expect is that information such as Ticket number, Name and PassengerId are not relevant variables to survival of a passanger (unless you beleive in magic or chance strongly).

Thus we remove these columns from our data set too.

[6]:
```
df.drop(columns=['PassengerId','Name','Ticket'],inplace=True)
```

[7]:
```
df.head()
```

[7]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	-1.0
1	1	1	0	38.0	1	0	71.2833	0.0
2	1	3	0	26.0	0	0	7.9250	-1.0
3	1	1	0	35.0	1	0	53.1000	-1.0
4	0	3	1	35.0	0	0	8.0500	-1.0

To have a better prediction and analysis (especially with Machine Learning methods) it is better (mandatory) to normalize our feature. To normalize Age column we divide the age by 100. To normalize Fare column we use min-max normalization.

[8]:
```
# normalize Age
df['Age'] = df['Age']/100.0
df.head()
```

[8]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	0.22	1	0	7.2500	-1.0
1	1	1	0	0.38	1	0	71.2833	0.0
2	1	3	0	0.26	0	0	7.9250	-1.0
3	1	1	0	0.35	1	0	53.1000	-1.0
4	0	3	1	0.35	0	0	8.0500	-1.0

[9]:
```
# normalize Fare
delta = df['Fare'].max() - df['Fare'].min()
df['Fare'] = ( df['Fare'] - df['Fare'].min() )/float(delta)
df.head()
```

[9]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	0.22	1	0	0.014151	-1.0
1	1	1	0	0.38	1	0	0.139136	0.0
2	1	3	0	0.26	0	0	0.015469	-1.0
3	1	1	0	0.35	1	0	0.103644	-1.0
4	0	3	1	0.35	0	0	0.015713	-1.0

3 Exploratory analysis

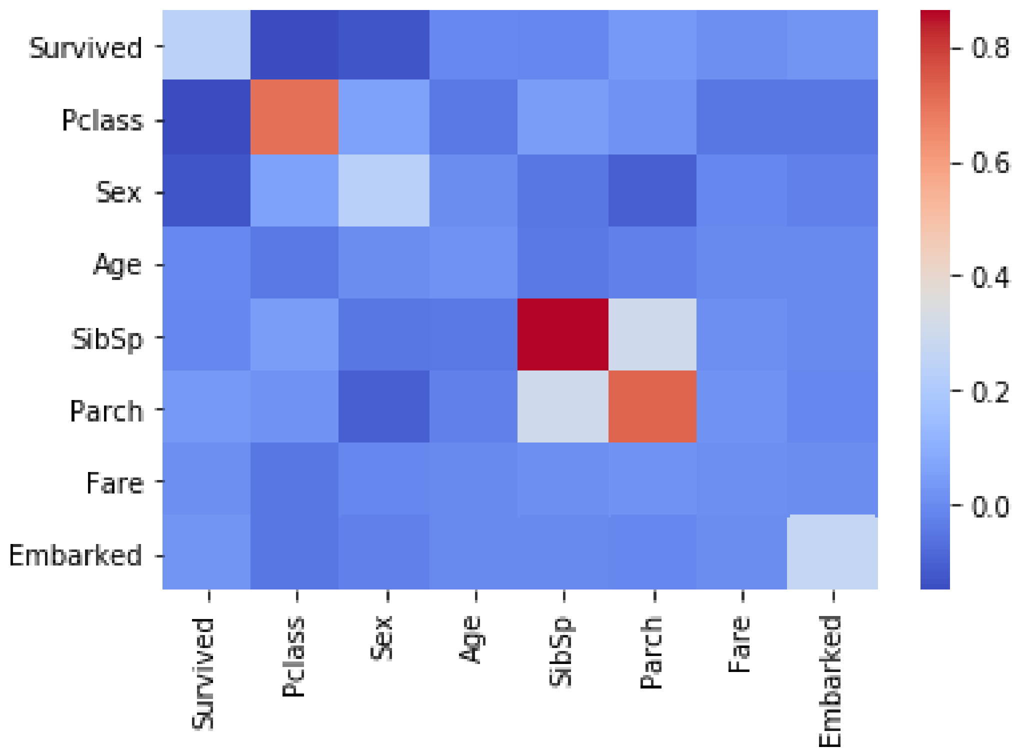
In order to see which features are each other (especially Survival) we plot all scatter plots with pairplot()

[10]:
```
sns.heatmap(df.cov(), cmap='coolwarm')
df.cov()
```

[10]:

	Survived	Pclass	Sex	Age	SibSp	Parch \
Survived	0.241533	-0.148165	-0.127618	-0.005513	-0.007932	0.039133
Pclass	-0.148165	0.702663	0.062801	-0.044960	0.052412	0.018370
Sex	-0.127618	0.062801	0.232247	0.006528	-0.046578	-0.101559

```
Age      -0.005513 -0.044960  0.006528  0.021102 -0.041633 -0.023442
SibSp    -0.007932  0.052412 -0.046578 -0.041633  0.864497  0.304513
Parch     0.039133  0.018370 -0.101559 -0.023442  0.304513  0.728103
Fare      0.013614 -0.047983 -0.009209  0.001441  0.013285  0.018079
Embarked  0.027798 -0.047358 -0.024388  0.000921  0.001952 -0.006274

              Fare  Embarked
Survived  0.013614  0.027798
Pclass   -0.047983 -0.047358
Sex      -0.009209 -0.024388
Age       0.001441  0.000921
SibSp     0.013285  0.001952
Parch     0.018079 -0.006274
Fare      0.010669  0.009531
Embarked  0.009531  0.272025
```

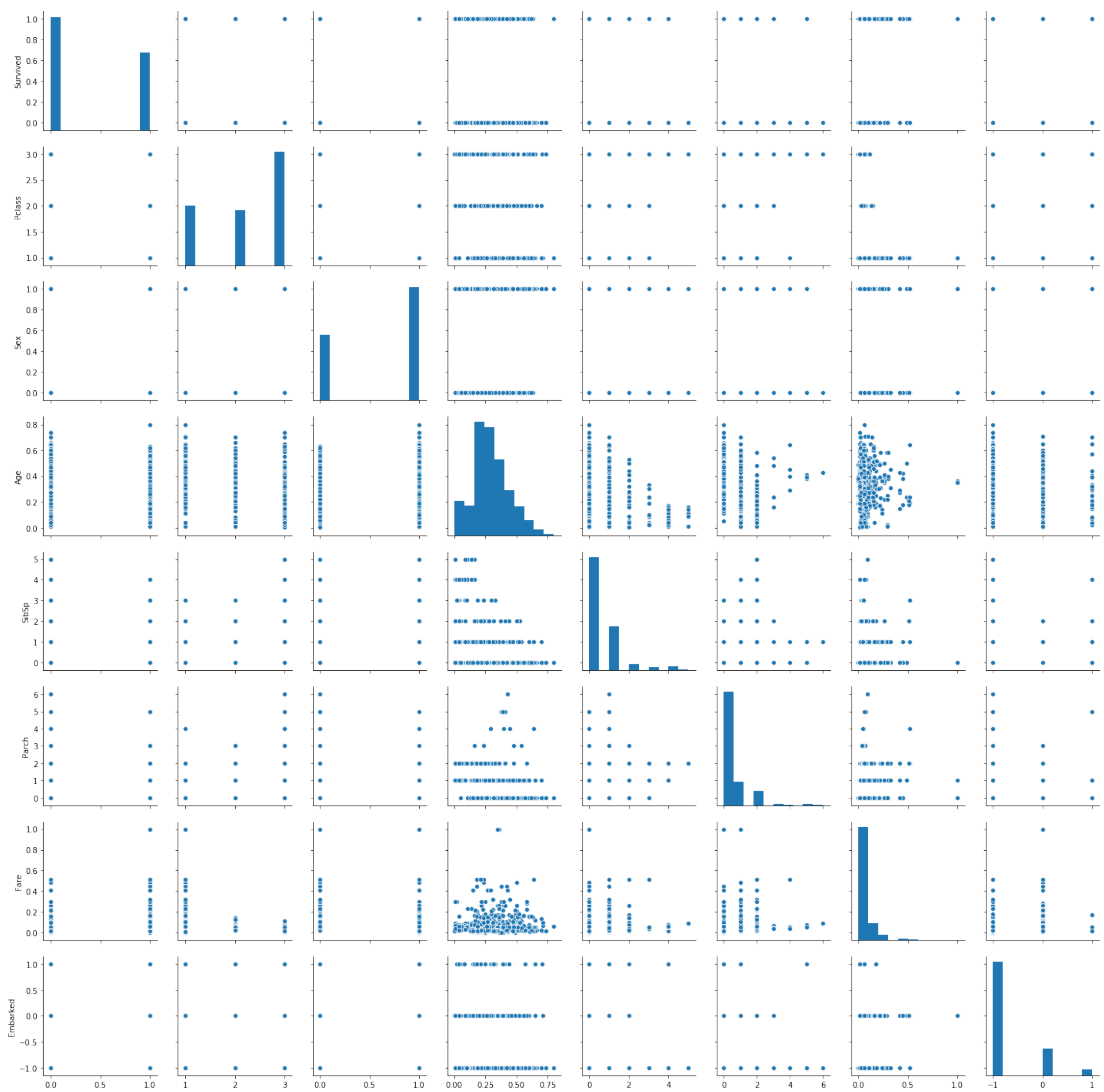


[11]:
```
sns.pairplot(df)
```

```
/home/javad/anaconda2/envs/ml/lib/python3.7/site-
packages/numpy/lib/histograms.py:824: RuntimeWarning: invalid value encountered
in greater_equal
  keep = (tmp_a >= first_edge)
/home/javad/anaconda2/envs/ml/lib/python3.7/site-
```

```
packages/numpy/lib/histograms.py:825: RuntimeWarning: invalid value encountered
in less_equal
  keep &= (tmp_a <= last_edge)
```

[11]:
```
<seaborn.axisgrid.PairGrid at 0x7f2632254f28>
```



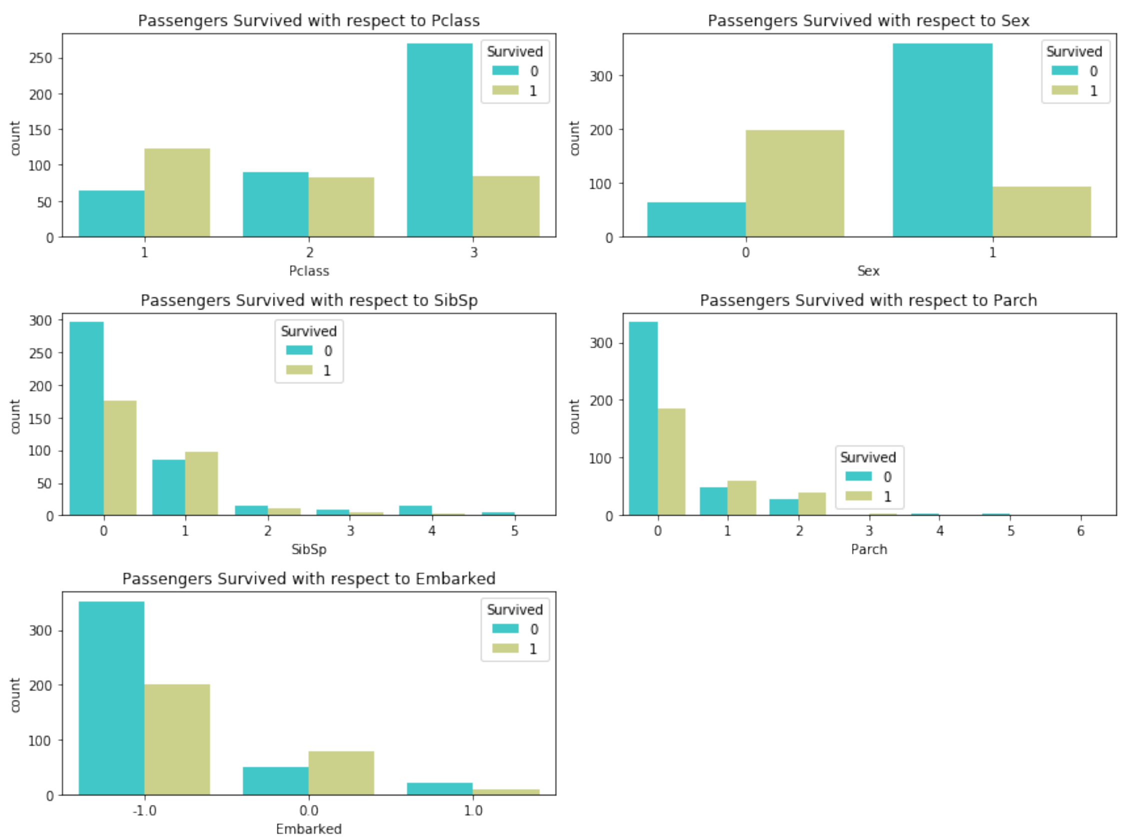
Some observations and insights based on plots If we look at scatter plots of Survival vs other vairables some primary observations are in sight: - Passengers with higher ages are less likely to survive - Passengers with high number of sibling are less likely to survive - Passengers with high number of parents and children are less likely to survive - Passengers with high number of fair paid are more likely to survive

However these plots don't show everything and it is better to investigate more. We plot Survival vs other variables to get new insights.

[12]:
```
df.columns
```

[12]:
```
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
       'Embarked'],
      dtype='object')
```

[13]:
```
list_of_categorycal_variables = ['Pclass','Sex','SibSp', 'Parch', 'Embarked']
fig = plt.figure(figsize=(12,9))
for i in range(0,len(list_of_categorycal_variables)):
    plt.subplot(3,2,i+1)
    sns.countplot(x=df[list_of_categorycal_variables[i]], hue=df['Survived'],␣
 ↪palette='rainbow')
    plt.title("Passengers Survived with respect to "+␣
 ↪list_of_categorycal_variables[i])
    plt.tight_layout()
plt.tight_layout()
```

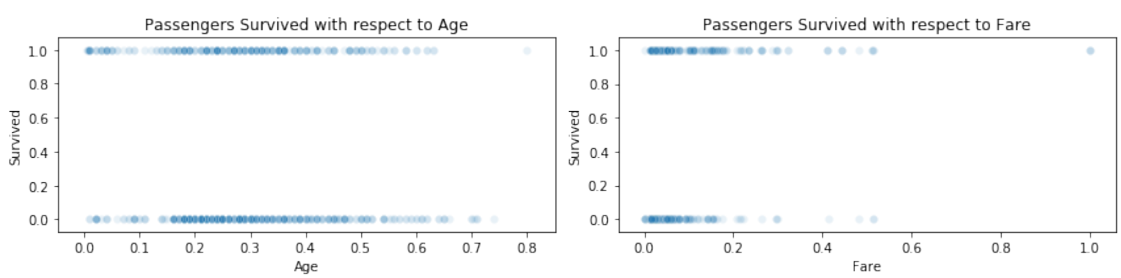


observations

- Females are more likely to survive than men
- Passengers embarked in Cherbourg are more likely to survive

- Passengers with no parents or children are more likely to not survive
- Passengers with Uppre Class are more likely to survive with respect to Lower Class
- Passangers with exactly one spouse or sibiling are more likely to survive in comparison to others

[14]:
```
list_of_categorycal_variables = ['Age','Fare']
fig = plt.figure(figsize=(12,3))
for i in range(0,len(list_of_categorycal_variables)):
    plt.subplot(1,2,i+1)
    sns.scatterplot(x=df[list_of_categorycal_variables[i]],y=df['Survived'],␣
 ↪palette='rainbow', alpha=0.1,)
    plt.title("Passengers Survived with respect to "+␣
 ↪list_of_categorycal_variables[i])
    plt.tight_layout()
plt.tight_layout()
```



observations

- Old Passengers are **slightly** more likely to not survive
- Passengers with high Fare are more likely to surive

4 Prediction

Since we're interested in the outcome of survival for each passenger or crew member, we can remove the **Survived** feature from this dataset and store it as its own separate variable `outcomes`. We will use these outcomes as our prediction targets.

[15]:
```
# store Survived columns of the dataset in new data frame called outcomes
outcomes = pd.DataFrame(df['Survived'])
outcomes.head()

# remove Survived column from the df DataFrame
df = df.drop(labels=['Survived'], axis=1)
```

[16]:
```
# Need to be completed
```

5 Conclusion

—> need to say something about data analysis

—> fun part:

but at the end we should remember all of these data analysis and finding feautres to predict survived passengers are nothing. The only thing that matters is **LOVE** and **LOVE** is the ultimate feature to survive …

[ ]:
```
!ipython nbconvert --to HTML Titanic.ipynb
!ipython nbconvert --to PDF Titanic.ipynb
```