Titanic Machine Learning from Disaster

The sinking of the Titanic is one of the most infamous shipwrecks in history. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. It is observed that some groups of people were more likely to survive than others, such as women, children, and the upper-class. In this term paper the Hypothesis is to analyze what sorts of people were likely to survive. In particular, I will apply tools of machine learning to predict which passengers survived the tragedy. Data source in this machine learning algorithm is a standard machine learning data set. Datasets are procured from the Kaggle competition website. Datasets are downloaded in csv format from Kaggle website to use in this term paper. The data procured is messy data which consist of missing values and NaN values. In order to clean the data, the following python codes are written in Ipython notebook.

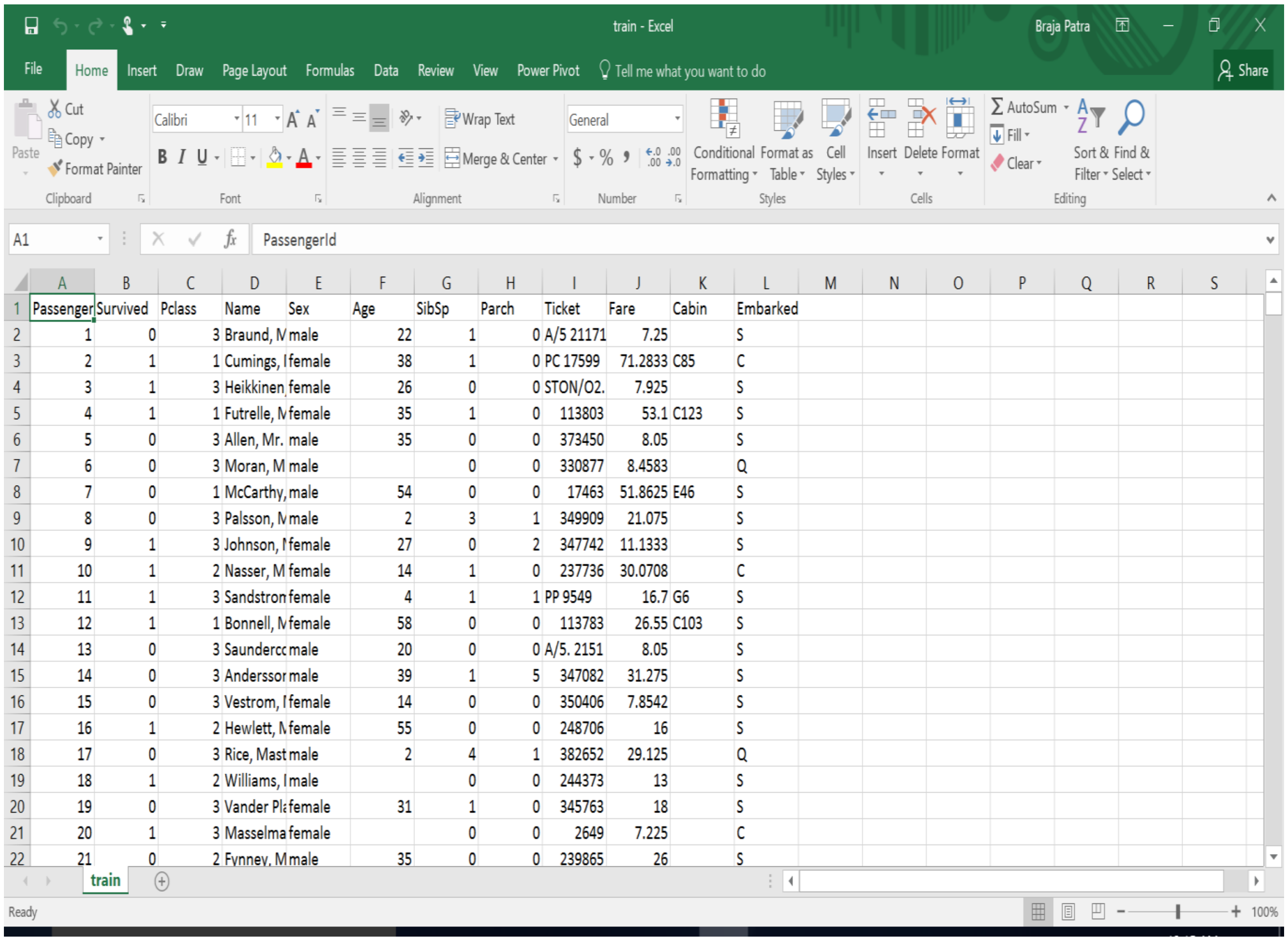
```
df = df.drop(['ticket','cabin'], axis=1)

df = df.dropna()
```

An exploratory analysis is done in the next section (Data Preparation and Data Analysis) of the term paper by using visualization in Python. Thereafter a predictive model (in Predictive Model section) is built by using Random Forest in Python. Random Forest is run on 891 observations in train.csv data set and later on 418 observations in test.csv data. Finally, the Hypothesis is established or proved by the resultant model which shows the passengers who survived and those who could not survive. Additionally, the model also gives the mean accuracy score which seems to be quite high. In the Appendix section all the Python codes are stated which are used in this term paper.

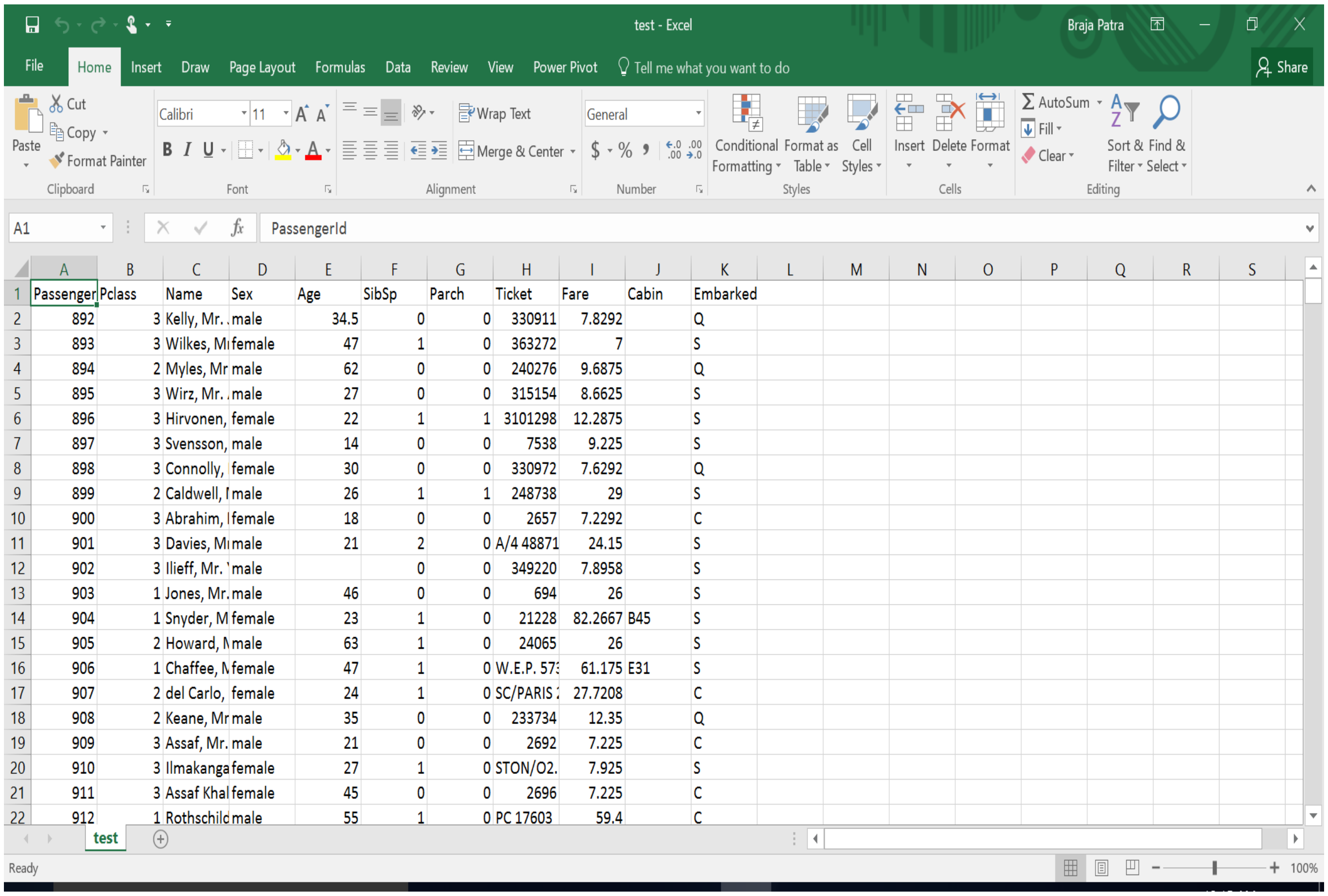
Data Preparation and Data Analysis:

Refer to the below screenshots of the datasets:



A1 PassengerId

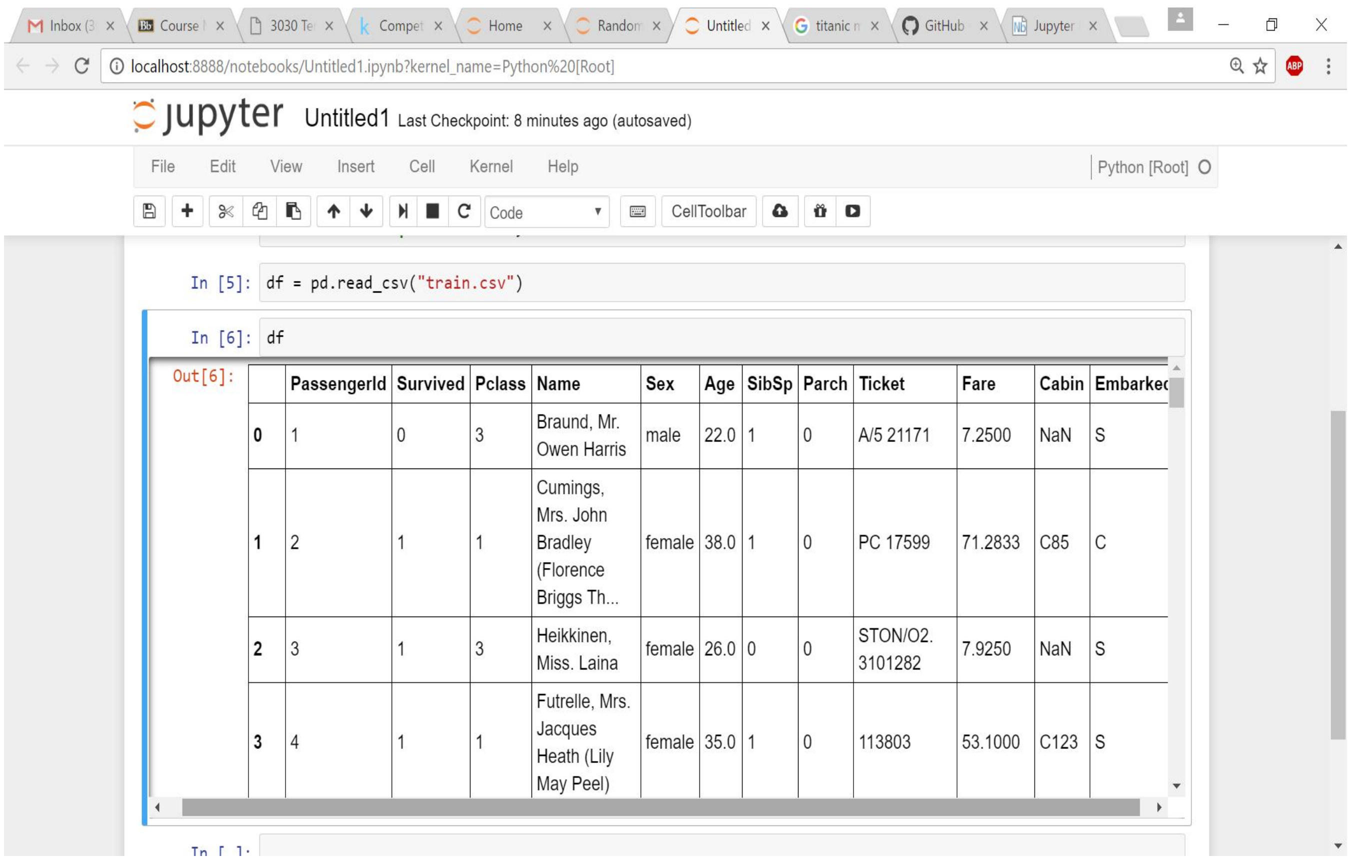
Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, I	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen,	female	26	0	0	STON/O2.	7.925		S
4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
6	0	3	Moran, M	male		0	0	330877	8.4583		Q
7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
9	1	3	Johnson, M	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
11	1	3	Sandstron	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
13	0	3	Saunderco	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Anderssor	male	39	1	5	347082	31.275		S
15	0	3	Vestrom, I	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, M	female	55	0	0	248706	16		S
17	0	3	Rice, Mast	male	2	4	1	382652	29.125		Q
18	1	2	Williams, I	male		0	0	244373	13		S
19	0	3	Vander Pla	female	31	1	0	345763	18		S
20	1	3	Masselma	female		0	0	2649	7.225		C
21	0	2	Fynney, M	male	35	0	0	239865	26		S



	A	B	C	D	E	F	G	H	I	J	K
1	Passenger	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	892	3	Kelly, Mr. .	male	34.5	0	0	330911	7.8292		Q
3	893	3	Wilkes, Mı	female	47	1	0	363272	7		S
4	894	2	Myles, Mr	male	62	0	0	240276	9.6875		Q
5	895	3	Wirz, Mr. .	male	27	0	0	315154	8.6625		S
6	896	3	Hirvonen,	female	22	1	1	3101298	12.2875		S
7	897	3	Svensson,	male	14	0	0	7538	9.225		S
8	898	3	Connolly,	female	30	0	0	330972	7.6292		Q
9	899	2	Caldwell, I	male	26	1	1	248738	29		S
10	900	3	Abrahim, I	female	18	0	0	2657	7.2292		C
11	901	3	Davies, Mı	male	21	2	0	A/4 48871	24.15		S
12	902	3	Ilieff, Mr. '	male		0	0	349220	7.8958		S
13	903	1	Jones, Mr.	male	46	0	0	694	26		S
14	904	1	Snyder, M	female	23	1	0	21228	82.2667	B45	S
15	905	2	Howard, N	male	63	1	0	24065	26		S
16	906	1	Chaffee, N	female	47	1	0	W.E.P. 57:	61.175	E31	S
17	907	2	del Carlo,	female	24	1	0	SC/PARIS :	27.7208		C
18	908	2	Keane, Mr	male	35	0	0	233734	12.35		Q
19	909	3	Assaf, Mr.	male	21	0	0	2692	7.225		C
20	910	3	Ilmakanga	female	27	1	0	STON/O2.	7.925		S
21	911	3	Assaf Khal	female	45	0	0	2696	7.225		C
22	912	1	Rothschild	male	55	1	0	PC 17603	59.4		C

I have two csv files, train.csv and test.csv. Train.csv contains the details of a subset of the passengers on board (891 to be exact) and tells us their details and whether they survived or not. Using the patterns identified in the train.csv data, it can be predicted whether the other 418 passengers on board (found in test.csv) survived. Train.csv file consists of the details of a number of passengers on board the Titanic. It consists of the Pclass which is the class of their ticket (1st, 2nd or 3rd), their name, their sex, age, Subsp (which is the number of siblings / spouses they had on board with them), Parch (which is the number of parents / children they had on board with them), their ticket type, the fare they paid for their ticket, their cabin number, Embarkation point (where they got on: Queenstown, Cherbourg or Southampton), and also tells whether they survived or not (1 = yes, 0 = sadly, not). This is the information used to make the predictions. I want to find here if there is a relationship between one of the variables and ultimate survival.

Below is the summary of the data contained in a Pandas DataFrame. The summary holds lot of information. First, it tells us that we have 891 observations or passengers. Next it shows us all of the columns in DataFrame. Each column tells us something about each of the observations, like their name, sex or age. These columns are called features of the dataset.



```
In [5]: df = pd.read_csv("train.csv")

In [6]: df
```

Out[6]:

	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

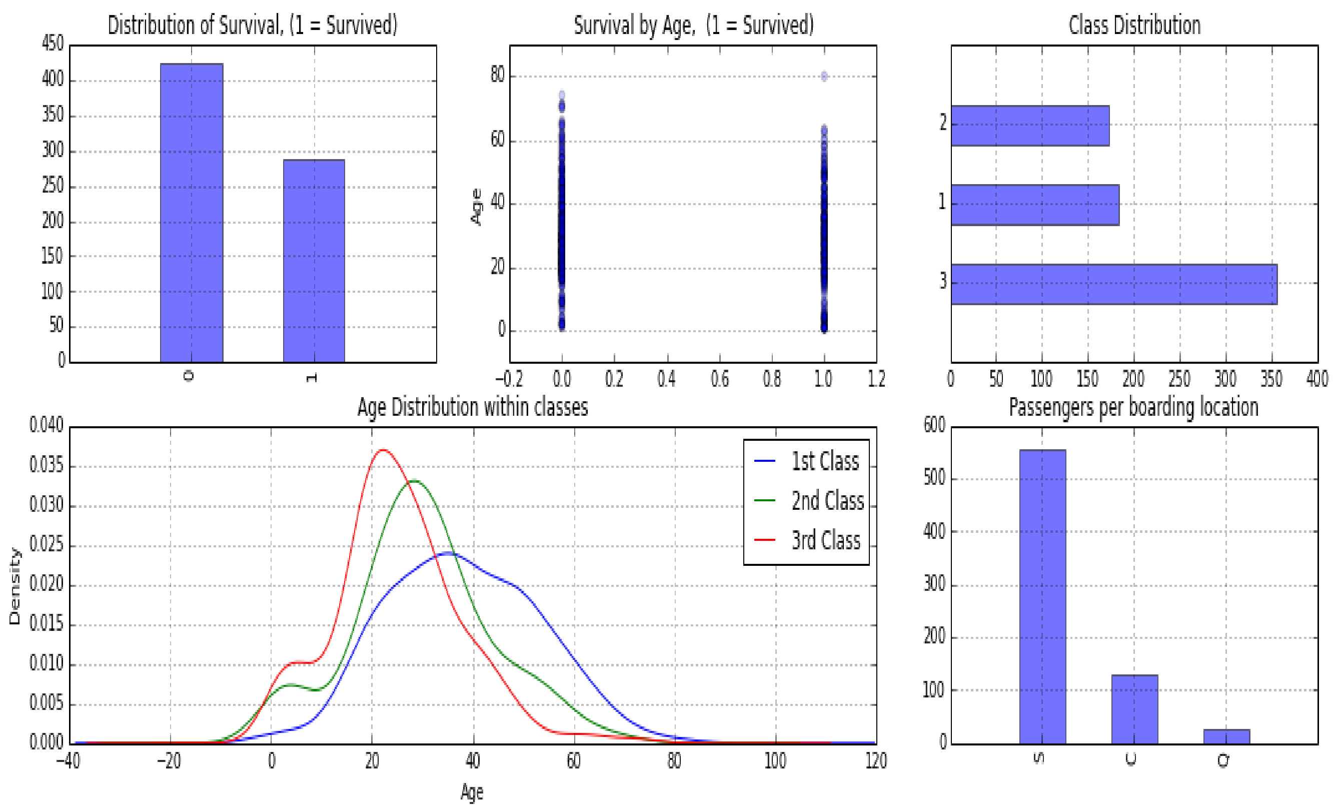
The features “Ticket” and “Cabin” have many missing values and so can’t add much value to the analysis. To handle this issue, I will drop them from the Dataframe. The below line of code drop the features entirely:

df = df.drop(['ticket','cabin'], axis=1)

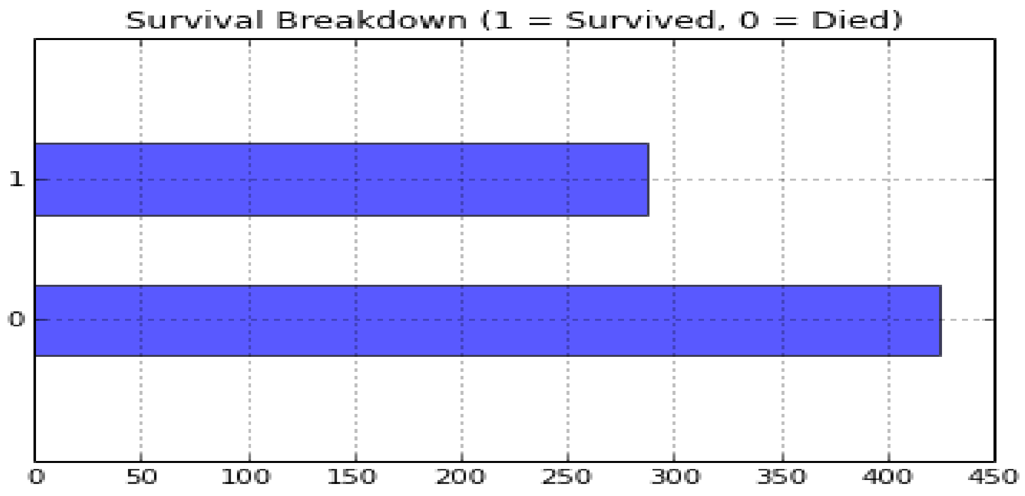
While the below line of code removes the NaN values from every remaining column / feature:

df = df.dropna()

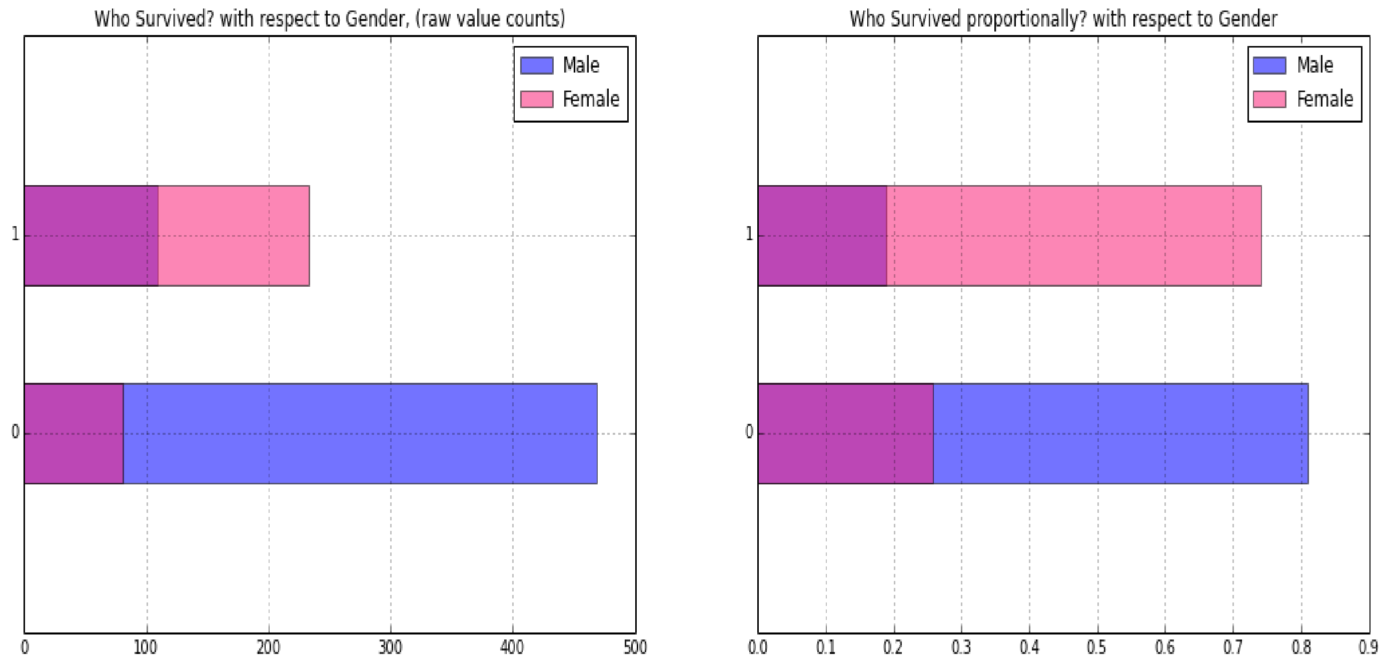
Thus I have a clean and tidy dataset that is ready for analysis. Now the data is visualized with the help of graphical representation in Ipython.



Exploratory analysis is now performed on the dataset. I would like to predict if an individual will survive based on the features in the data like - Traveling Class (called pclass in the data), Sex, Age, Fare Price. A bar graph of those who Survived vs. those who did not is plotted below.

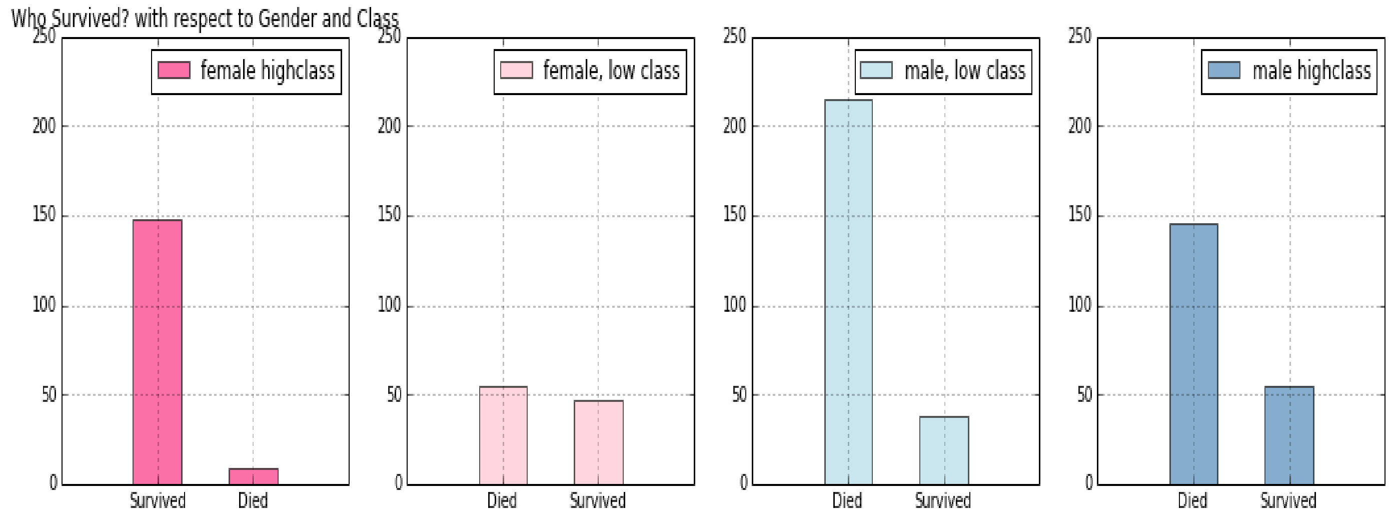


The previous graph is broken down by gender as shown in the below.



As per the above, it’s clear that although more men died and survived in raw value counts, females had a greater survival rate proportionally (~25%), than men (~20%).

Using Pclass feature, classes are bucketed as lowest class or any of the high classes (classes 1 - 2), 3 being the lowest class. I will further break it down by Gender and what Class they were traveling in.



Predictive Model:

Now with the help of the above graph, I have a lot more information on who survived and died in the tragedy. With this deeper understanding, I would like to create a better and more insightful model. This is a typical process in interactive data analysis.

The selected model here is Random Forest. As explained by Wikipedia: Random forests are an ensemble learning method for classification (and regression) that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. A random forest algorithm randomly generates many extremely simple models to explain the variance observed in random subsections of data. Individually these models are all awful. They can be powerful predictive tools, once they are averaged. While the vast majority of those models were extremely poor; they were all as bad as each other on average. So when their predictions are averaged together, the bad ones average their effect on the model out to zero. The thing that remains, if anything, is one or a handful of those models have stumbled upon the true structure of the data. Thus a random forest is an ensemble of decision trees which will output a prediction value, in this case survival. Each decision tree is constructed by using a random subset of the training data. The random forest model shows the process of instantiating and fitting a random forest, generating predictions from the resulting model, and then scoring the results. An acceptable formula for the machine learning algorithm is presented below:

```
formula_ml = 'Survived ~ C(Pclass) + C(Sex) + Age + SibSp + Parch + C(Embarked)'
```

```
Mean accuracy of Random Forest Predictions on the data: 0.945224719101
```

Conclusion:

The Random Forest model thus gives the predicted values of survival for the passengers which is given by y value. Also the model scores the results of predictions which is given by the Mean accuracy of Random Forest Predictions on the data = 0.945224719101. Thus the Hypothesis to predict which passengers were likely to survive is established through the model.

Appendix:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.nonparametric.kde import KDEUnivariate
from statsmodels.nonparametric import smoothers_lowess
from pandas import Series, DataFrame
from patsy import dmatrices
from sklearn import datasets, svm

df = pd.read_csv("train.csv")

df

df = df.drop(['Ticket','Cabin'], axis=1)

# Remove NaN values
df = df.dropna()

df


# specifies the parameters of our graphs
fig = plt.figure(figsize=(18,6), dpi=1600)
alpha=alpha_scatterplot = 0.2
alpha_bar_chart = 0.55

# lets us plot many diffrent shaped graphs together
ax1 = plt.subplot2grid((2,3),(0,0))
# plots a bar graph of those who surived vs those who did not.
df.Survived.value_counts().plot(kind='bar', alpha=alpha_bar_chart)
# this nicely sets the margins in matplotlib to deal with a recent bug 1.3.1
ax1.set_xlim(-1, 2)
# puts a title on our graph
plt.title("Distribution of Survival, (1 = Survived)")

plt.subplot2grid((2,3),(0,1))
plt.scatter(df.Survived, df.Age, alpha=alpha_scatterplot)
# sets the y axis lable
plt.ylabel("Age")
# formats the grid line style of our graphs
```

```
plt.grid(b=True, which='major', axis='y')
plt.title("Survival by Age,  (1 = Survived)")

ax3 = plt.subplot2grid((2,3),(0,2))
df.Pclass.value_counts().plot(kind="barh", alpha=alpha_bar_chart)
ax3.set_ylim(-1, len(df.Pclass.value_counts()))
plt.title("Class Distribution")

plt.subplot2grid((2,3),(1,0), colspan=2)
# plots a kernel density estimate of the subset of the 1st class passangers's
age
df.Age[df.Pclass == 1].plot(kind='kde')
df.Age[df.Pclass == 2].plot(kind='kde')
df.Age[df.Pclass == 3].plot(kind='kde')
 # plots an axis lable
plt.xlabel("Age")
plt.title("Age Distribution within classes")
# sets our legend for our graph.
plt.legend(('1st Class', '2nd Class','3rd Class'),loc='best')

ax5 = plt.subplot2grid((2,3),(1,2))
df.Embarked.value_counts().plot(kind='bar', alpha=alpha_bar_chart)
ax5.set_xlim(-1, len(df.Embarked.value_counts()))
# specifies the parameters of our graphs
plt.title("Passengers per boarding location")
```

```
Out[8]: <matplotlib.text.Text at 0xc889c88>
```

```
plt.figure(figsize=(6,4))
fig, ax = plt.subplots()
df.Survived.value_counts().plot(kind='barh', color="blue", alpha=.65)
ax.set_ylim(-1, len(df.Survived.value_counts()))
plt.title("Survival Breakdown (1 = Survived, 0 = Died)")
```

```
Out[9]: <matplotlib.text.Text at 0xd002b00>
        <matplotlib.figure.Figure at 0xc174e48>
```

```
fig = plt.figure(figsize=(18,6))

#create a plot of two subsets, male and female, of the survived variable.
#After we do that we call value_counts() so it can be easily plotted as a bar
graph.
```

```
#'barh' is just a horizontal bar graph
df_male = df.Survived[df.Sex == 'male'].value_counts().sort_index()
df_female = df.Survived[df.Sex == 'female'].value_counts().sort_index()

ax1 = fig.add_subplot(121)
df_male.plot(kind='barh',label='Male', alpha=0.55)
df_female.plot(kind='barh', color='#FA2379',label='Female', alpha=0.55)
plt.title("Who Survived? with respect to Gender, (raw value counts) "); plt.l
egend(loc='best')
ax1.set_ylim(-1, 2)

#adjust graph to display the proportions of survival by gender
ax2 = fig.add_subplot(122)
(df_male/float(df_male.sum())).plot(kind='barh',label='Male', alpha=0.55)
(df_female/float(df_female.sum())).plot(kind='barh', color='#FA2379',label='F
emale', alpha=0.55)
plt.title("Who Survived proportionally? with respect to Gender"); plt.legend(
loc='best')

ax2.set_ylim(-1, 2)
```

```
Out[10]:(-1, 2)
```

```
fig = plt.figure(figsize=(18,4), dpi=1600)
alpha_level = 0.65

# building on the previous code, here we create an additional subset with in
the gender subset
# we created for the survived variable. I know, thats a lot of subsets. After
we do that we call
# value_counts() so it can be easily plotted as a bar graph. this is repeated
for each gender
# class pair.
ax1=fig.add_subplot(141)
female_highclass = df.Survived[df.Sex == 'female'][df.Pclass != 3].value_coun
ts()
female_highclass.plot(kind='bar', label='female, highclass', color='#FA2479',
alpha=alpha_level)
ax1.set_xticklabels(["Survived", "Died"], rotation=0)
ax1.set_xlim(-1, len(female_highclass))
plt.title("Who Survived? with respect to Gender and Class"); plt.legend(loc='
best')

ax2=fig.add_subplot(142, sharey=ax1)
```

```
female_lowclass = df.Survived[df.Sex == 'female'][df.Pclass == 3].value_count
s()
female_lowclass.plot(kind='bar', label='female, low class', color='pink', alp
ha=alpha_level)
ax2.set_xticklabels(["Died","Survived"], rotation=0)
ax2.set_xlim(-1, len(female_lowclass))
plt.legend(loc='best')

ax3=fig.add_subplot(143, sharey=ax1)
male_lowclass = df.Survived[df.Sex == 'male'][df.Pclass == 3].value_counts()
male_lowclass.plot(kind='bar', label='male, low class',color='lightblue', alp
ha=alpha_level)
ax3.set_xticklabels(["Died","Survived"], rotation=0)
ax3.set_xlim(-1, len(male_lowclass))
plt.legend(loc='best')

ax4=fig.add_subplot(144, sharey=ax1)
male_highclass = df.Survived[df.Sex == 'male'][df.Pclass != 3].value_counts()
male_highclass.plot(kind='bar', label='male, highclass', alpha=alpha_level, c
olor='steelblue')
ax4.set_xticklabels(["Died","Survived"], rotation=0)
ax4.set_xlim(-1, len(male_highclass))
plt.legend(loc='best')
```

```
Out[11]: <matplotlib.legend.Legend at 0xdbbb518>
```

```
# Create an acceptable formula for our machine learning algorithms
formula_ml = 'Survived ~ C(Pclass) + C(Sex) + Age + SibSp + Parch + C(Embarke
d)'

# import the machine learning library that holds the randomforest
import sklearn.ensemble as ske

# Create the random forest model and fit the model to our training data
y, x = dmatrices(formula_ml, data=df, return_type='dataframe')
# RandomForestClassifier expects a 1 demensional NumPy array, so we convert
y = np.asarray(y).ravel()
#instantiate and fit our model
results_rf = ske.RandomForestClassifier(n_estimators=100).fit(x, y)
# Score the results
score = results_rf.score(x, y)
print "Mean accuracy of Random Forest Predictions on the data was: {0}".forma
t(score)
Mean accuracy of Random Forest Predictions on the data was: 0.945224719101
```

Reference:

https://github.com/agconti/kaggle-titanic/blob/master/Titanic.ipynb

https://www.kaggle.com/c/titanic/details/getting-started-with-python