Machine Learning Engineer Nanodegree

Introduction and Foundations

Project 0: Titanic Survival Exploration

In 1912, the ship RMS Titanic struck an iceberg on its maiden voyage and sank, resulting in the deaths of most of its passengers and crew. In this introductory project, we will explore a subset of the RMS Titanic passenger manifest to determine which features best predict whether someone survived or did not survive. To complete this project, you will need to implement several conditional predictions and answer the questions below. Your project submission will be evaluated based on the completion of the code and your responses to the questions.

> **Tip:** Quoted sections like this will provide helpful instructions on how to navigate and use an iPython notebook.

Getting Started

To begin working with the RMS Titanic passenger data, we'll first need to `import` the functionality we need, and load our data into a `pandas` DataFrame.
Run the code cell below to load our data and display the first few entries (passengers) for examination using the `.head()` function.

> **Tip:** You can run a code cell by clicking on the cell and using the keyboard shortcut **Shift + Enter** or **Shift + Return**. Alternatively, a code cell can be executed using the **Play** button in the hotbar after selecting it. Markdown cells (text cells like this one) can be edited by double-clicking, and saved using these same shortcuts. Markdown (http://daringfireball.net/projects/markdown/syntax) allows you to write easy-to-read plain text that can be converted to HTML.

In [32]:

```
import numpy as np
import pandas as pd

# RMS Titanic data visualization code
from titanic_visualizations import survival_stats
from IPython.display import display
%matplotlib inline

# Load the dataset
in_file = 'titanic_data.csv'
full_data = pd.read_csv(in_file)

# Print the first few entries of the RMS Titanic data
display(full_data.head())
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450

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)

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.
Run the code block cell to remove **Survived** as a feature of the dataset and store it in `outcomes`.

In [33]:

```
# Store the 'Survived' feature in a new variable and remove it from the dataset
outcomes = full_data['Survived']
data = full_data.drop('Survived', axis = 1)

# Show the new dataset with 'Survived' removed
display(data.head())
```

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

The very same sample of the RMS Titanic data now shows the **Survived** feature removed from the DataFrame. Note that `data` (the passenger data) and `outcomes` (the outcomes of survival) are now *paired*. That means for any passenger `data.loc[i]`, they have the survival outcome `outcome[i]`.

To measure the performance of our predictions, we need a metric to score our predictions against the true outcomes of survival. Since we are interested in how *accurate* our predictions are, we will calculate the proportion of passengers where our prediction of their survival is correct. Run the code cell below to create our `accuracy_score` function and test a prediction on the first five passengers.

Think: *Out of the first five passengers, if we predict that all of them survived, what would you expect the accuracy of our predictions to be?*

In [34]:

```
def accuracy_score(truth, pred):
    """ Returns accuracy score for input truth and predictions. """

    # Ensure that the number of predictions matches number of outcomes
    if len(truth) == len(pred):

        # Calculate and return the accuracy as a percent
        return "Predictions have an accuracy of {:.2f}%.".format((truth == pre
d).mean()*100)

    else:
        return "Number of predictions does not match number of outcomes!"

# Test the 'accuracy_score' function
predictions = pd.Series(np.ones(5, dtype = int))
print(accuracy_score(predictions, outcomes[:5]))
```

```
Predictions have an accuracy of 60.00%.
```

> **Tip:** If you save an iPython Notebook, the output from running code blocks will also be saved. However, the state of your workspace will be reset once a new session is started. Make sure that you run all of the code blocks from your previous session to reestablish variables and functions before picking up where you last left off.

Making Predictions

If we were told to make a prediction about any passenger aboard the RMS Titanic who we did not know anything about, then the best prediction we could make would be that they did not survive. This is because we can assume that a majority of the passengers as a whole did not survive the ship sinking. The function below will always predict that a passenger did not survive.

In [35]:

```
def predictions_0(data):
    """ Model with no features. Always predicts a passenger did not survive. "
""

    predictions = []
    for _, passenger in data.iterrows():

        # Predict the survival of 'passenger'
        predictions.append(0)

    # Return our predictions
    return pd.Series(predictions)

# Make the predictions
predictions = predictions_0(data)
```

Question 1

Using the RMS Titanic data, how accurate would a prediction be that none of the passengers survived?
Hint: Run the code cell below to see the accuracy of this prediction.

In [36]:

```
print(accuracy_score(outcomes, predictions))

# Since every survived passanger marked as 1 and died as 0,
# we can sum up all entries to get an amount of survived people.
survivedCounter = 0
for x in outcomes:
    survivedCounter += x

# To calculate amount of died people let's just subtract size
# of survived people from the whole sample.
died = len(outcomes) - survivedCounter

print("Sirvived:",survivedCounter)
print("Died:",died)
print("Died Predicted:",len(predictions))

# Since prediction says that all people were died
# and accuracy is a proportion of passengers where
# our prediction of their survival is correct.
# We can get accuracy score by dividing actual size
# of survived people by the size of our prediction:
print("Prediction Accuracy:", died/len(predictions))
```

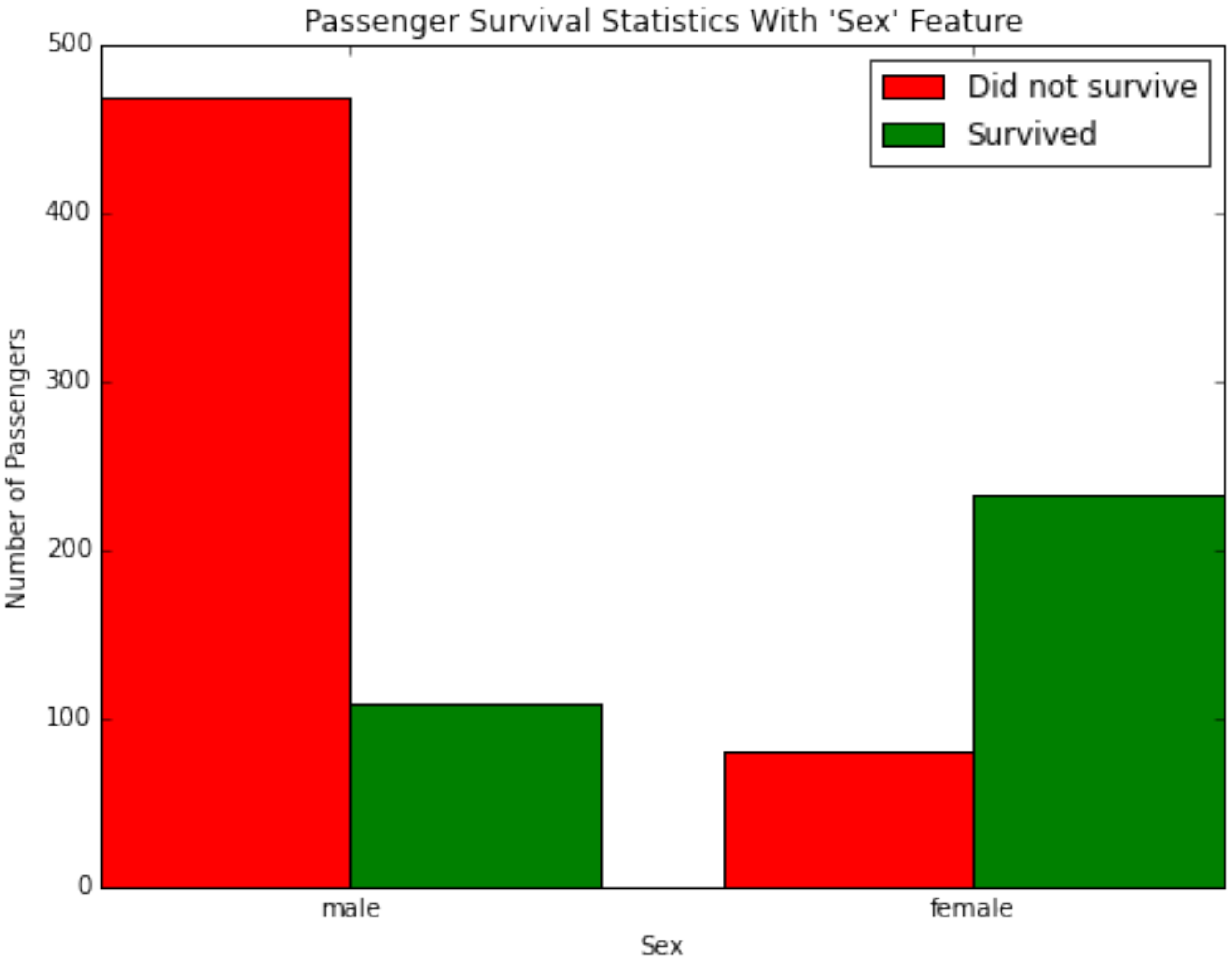
```
Predictions have an accuracy of 61.62%.
Sirvived: 342
Died: 549
Died Predicted: 891
Prediction Accuracy: 0.616161616162
```

Answer: Prediction accuracy is equal to 61.62% which means that 61.62% of people died. In the code block above I also added my algorithm of calcualting accuracy for this very case.

Let's take a look at whether the feature **Sex** has any indication of survival rates among passengers using the `survival_stats` function. This function is defined in the `titanic_visualizations.py` Python script included with this project. The first two parameters passed to the function are the RMS Titanic data and passenger survival outcomes, respectively. The third parameter indicates which feature we want to plot survival statistics across.
Run the code cell below to plot the survival outcomes of passengers based on their sex.

In [37]:

```
survival_stats(data, outcomes, 'Sex')
```



Examining the survival statistics, a large majority of males did not survive the ship sinking. However, a majority of females *did* survive the ship sinking. Let's build on our previous prediction: If a passenger was female, then we will predict that they survived. Otherwise, we will predict the passenger did not survive. Fill in the missing code below so that the function will make this prediction.
Hint: You can access the values of each feature for a passenger like a dictionary. For example, `passenger['Sex']` is the sex of the passenger.

In [38]:

```
def predictions_1(data):
    """ Model with one feature:
            - Predict a passenger survived if they are female. """

    predictions = []
    for _, passenger in data.iterrows():
        outcome = 0
        if passenger['Sex'] == 'female':
            outcome = 1
        predictions.append(outcome)

    # Return our predictions
    return pd.Series(predictions)

# Make the predictions
predictions = predictions_1(data)
```

Question 2

How accurate would a prediction be that all female passengers survived and the remaining passengers did not survive?
Hint: Run the code cell below to see the accuracy of this prediction.

In [39]:

```
print(accuracy_score(outcomes, predictions))
```

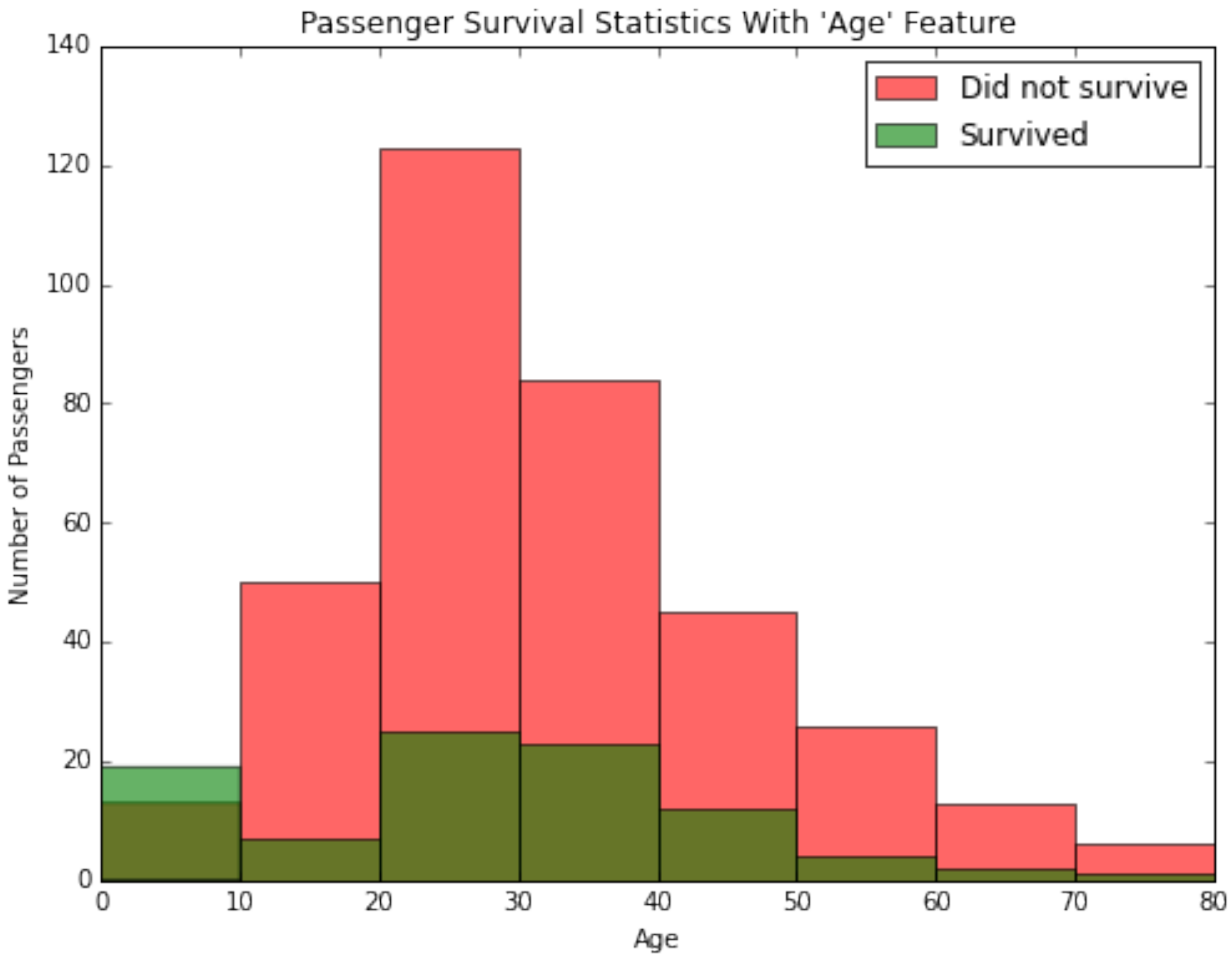
```
Predictions have an accuracy of 78.68%.
```

Answer: Predictions have an accuracy of 78.68%.

Using just the **Sex** feature for each passenger, we are able to increase the accuracy of our predictions by a significant margin. Now, let's consider using an additional feature to see if we can further improve our predictions. Consider, for example, all of the male passengers aboard the RMS Titanic: Can we find a subset of those passengers that had a higher rate of survival? Let's start by looking at the **Age** of each male, by again using the `survival_stats` function. This time, we'll use a fourth parameter to filter out the data so that only passengers with the **Sex** 'male' will be included.
Run the code cell below to plot the survival outcomes of male passengers based on their age.

In [40]:

```
survival_stats(data, outcomes, 'Age', ["Sex == 'male'"])
```



```
Passengers with missing 'Age' values: 124 (16 survived, 108 did no
t survive)
```

Examining the survival statistics, the majority of males younger then 10 survived the ship sinking, whereas most males age 10 or older *did not survive* the ship sinking. Let's continue to build on our previous prediction: If a passenger was female, then we will predict they survive. If a passenger was male and younger than 10, then we will also predict they survive. Otherwise, we will predict they do not survive.
Fill in the missing code below so that the function will make this prediction.
Hint: You can start your implementation of this function using the prediction code you wrote earlier from `predictions_1`.

In [41]:

```
def predictions_2(data):
    """ Model with two features: 
            - Predict a passenger survived if they are female.
            - Predict a passenger survived if they are male and younger than 1
0. """
    
    predictions = []
    for _, passenger in data.iterrows():
        outcome = 0
        
        if passenger['Sex'] == 'female':
            outcome = 1
        elif passenger['Sex'] == 'male' and passenger['Age'] < 10:
            outcome = 1
        
        predictions.append(outcome)
    
    # Return our predictions
    return pd.Series(predictions)

# Make the predictions
predictions = predictions_2(data)
```

Question 3

How accurate would a prediction be that all female passengers and all male passengers younger than 10 survived?
Hint: Run the code cell below to see the accuracy of this prediction.

In [42]:

```
print(accuracy_score(outcomes, predictions))
```

```
Predictions have an accuracy of 79.35%.
```

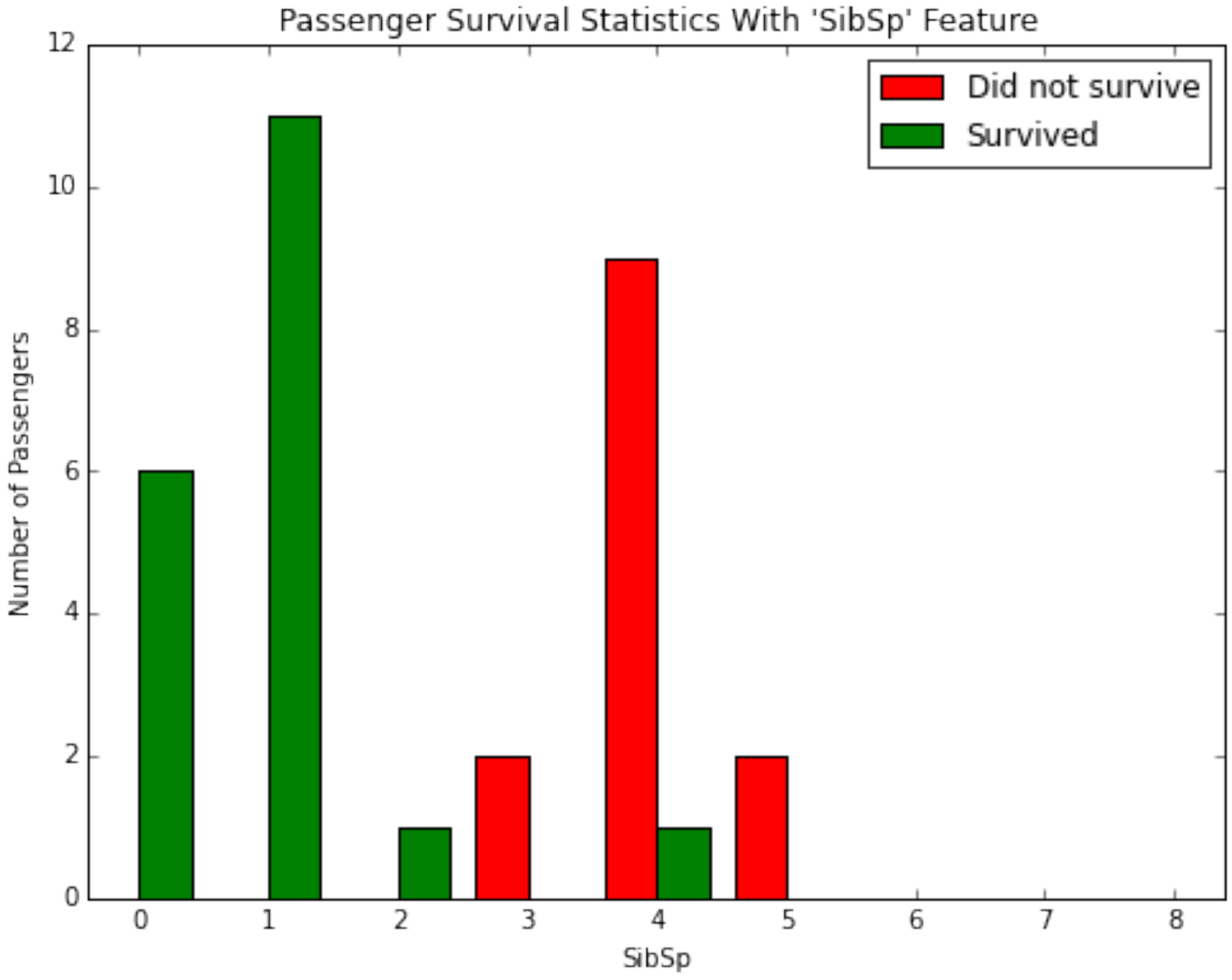
Answer: Predictions have an accuracy of 79.35%. Male passenger younger than 10 had more chance to survive than other male passengers.

Adding the feature **Age** as a condition in conjunction with **Sex** improves the accuracy by a small margin more than with simply using the feature **Sex** alone. Now it's your turn: Find a series of features and conditions to split the data on to obtain an outcome prediction accuracy of at least 80%. This may require multiple features and multiple levels of conditional statements to succeed. You can use the same feature multiple times with different conditions.
Pclass, **Sex**, **Age**, **SibSp**, and **Parch** are some suggested features to try.

Use the `survival_stats` function below to to examine various survival statistics.
Hint: To use mulitple filter conditions, put each condition in the list passed as the last argument. Example: `["Sex == 'male'", "Age < 18"]`

```
In [44]:
```

```
survival_stats(data, outcomes, 'SibSp', ["Sex == 'male'", "Age < 10"])
```

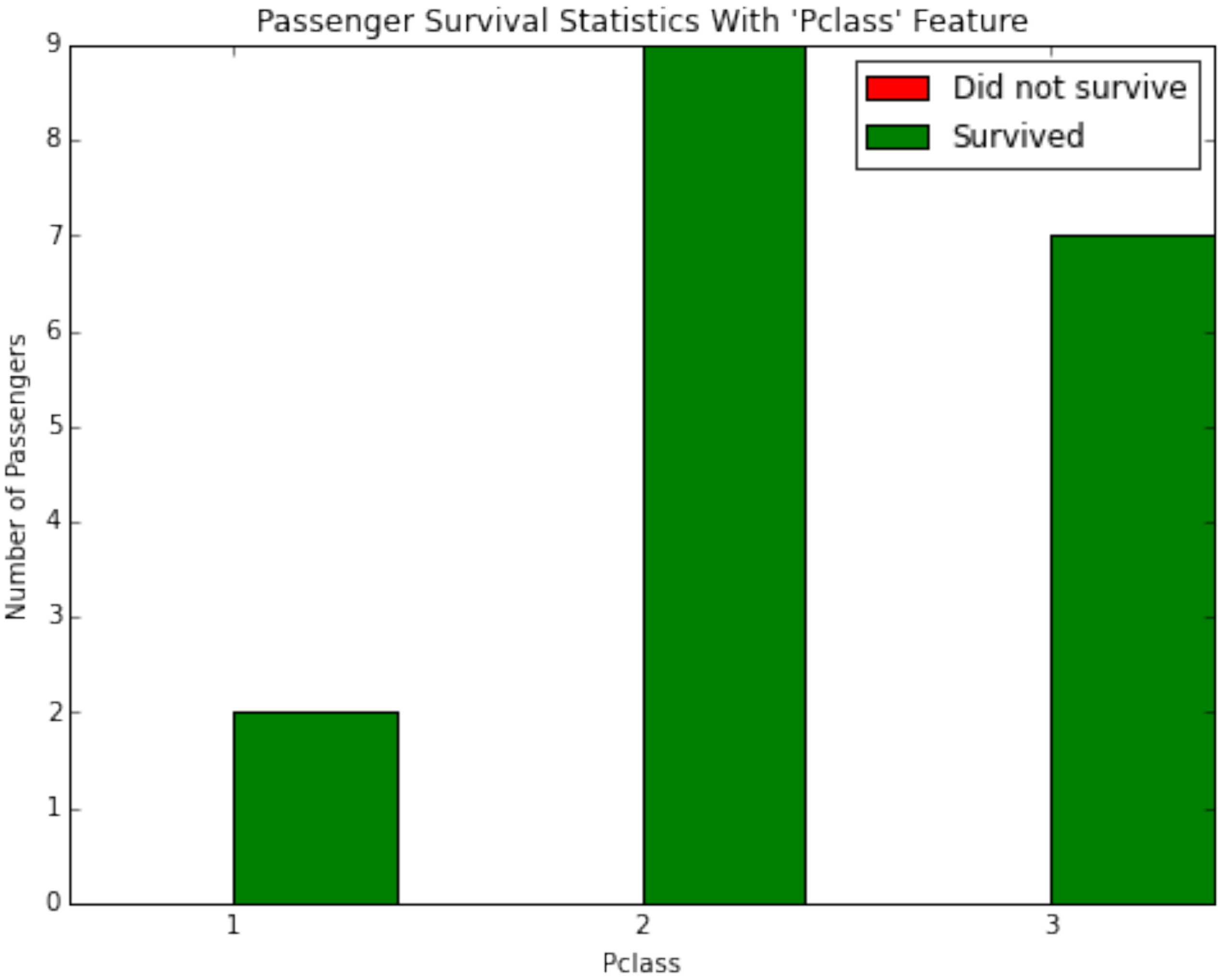


On the graph above we see that all young male passengers who had less than 3 siblings has survived. Based on that let's mark all young male passengers, who had more than 2 siblings as died and who has less than 3 as survived.

Now, let's check if there is other data which we can cut off.

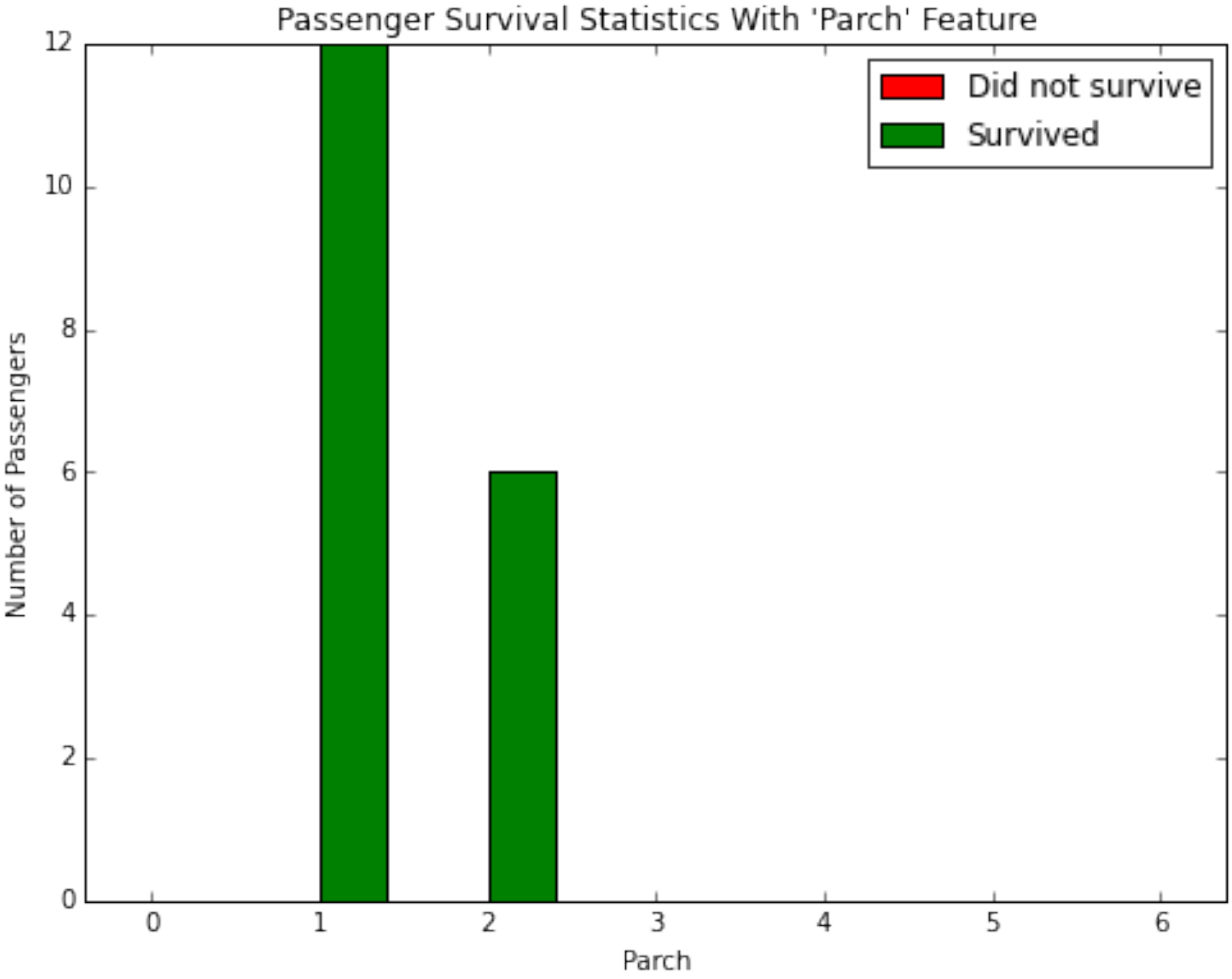
In [45]:

```
survival_stats(data, outcomes, 'Pclass', ["Sex == 'male'", "Age < 10", "SibSp
< 3"])
```



In [46]:

```
survival_stats(data, outcomes, 'Parch', ["Sex == 'male'", "Age < 10", "SibSp <
3"])
```



As we see above, there is no other data based on which we can decide has person or not. It means that there is no way to make this prediction better.

After exploring the survival statistics visualization, fill in the missing code below so that the function will make your prediction.
Make sure to keep track of the various features and conditions you tried before arriving at your final prediction model.
Hint: You can start your implementation of this function using the prediction code you wrote earlier from `predictions_2`.

In [47]:

```
def predictions_3(data):
    """ Model with multiple features. Makes a prediction with an accuracy of at least 80%. """

    predictions = []
    for _, passenger in data.iterrows():
        outcome = 0

        if passenger['Sex'] == 'female':
            outcome = 1
        elif passenger['Sex'] == 'male' \
            and passenger['Age'] < 10 \
            and passenger['SibSp'] < 3 :
            outcome = 1

        predictions.append(outcome)

    # Return our predictions
    return pd.Series(predictions)

# Make the predictions
predictions = predictions_3(data)
```

Question 4

Describe the steps you took to implement the final prediction model so that it got an accuracy of at least 80%. What features did you look at? Were certain features more informative than others? Which conditions did you use to split the survival outcomes in the data? How accurate are your predictions?
Hint: Run the code cell below to see the accuracy of your predictions.

In [48]:

```
print(accuracy_score(outcomes, predictions))
```

```
Predictions have an accuracy of 80.70%.
```

Answer:

Final accuracy score is equal to 80.70%. To achieve this score I took a look on all other features which we didn't explore above, trying to find a group with survived majority. The first of them with likely positive outcome was a group of people with low amount of siblings on the boat(Less than 3). All young male passengers who had less than 3 siblings has survived. Amount of sublings was the most informative feature from the rest. It showed a group with survived majority, while other features(Pclass and Parch) were not so informative. There was some significant information about passenger's social-economic class. All people with class < 3 has survived, while most of people from 3rd class died. But after I applied filtering by amount of siblings this correlation disappeared.

At the end I tried to find which feature helps to find the biggest amount of survived people.

First time I tried to filter young males by amount siblings(SibSp < 3). Amount of survived people - 18. Accuracy score 80.70%

Then I tried to filter young males by their social-economic class(Pclass < 3). Amount of survived people - 11. Accuracy score - 79.91%

Based on that splitting died/survived passengers by amount of siblings looks more accurate

Conclusion

Congratulations on what you've accomplished here! You should now have an algorithm for predicting whether or not a person survived the Titanic disaster, based on their features. In fact, what you have done here is a manual implementation of a simple machine learning model, the *decision tree*. In a decision tree, we split the data into smaller groups, one feature at a time. Each of these splits will result in groups that are more homogeneous than the original group, so that our predictions become more accurate. The advantage of having a computer do things for us is that it will be more exhaustive and more precise than our manual exploration above. This link (http://www.r2d3.us/visual-intro-to-machine-learning-part-1/) provides another introduction into machine learning using a decision tree.

A decision tree is just one of many algorithms that fall into the category of *supervised learning*. In this Nanodegree, you'll learn about supervised learning techniques first. In supervised learning, we concern ourselves with using features of data to predict or model things with objective outcome labels. That is, each of our datapoints has a true outcome value, whether that be a category label like survival in the Titanic dataset, or a continuous value like predicting the price of a house.

Question 5

Can you think of an example of where supervised learning can be applied?
Hint: Be sure to note the outcome variable to be predicted and at least two features that might be useful for making the predictions.

Answer:

Common algorithm could be applied in a financial sphere. Using decision tree models we can decide either give person a credit or not. Feature set could contain client's gender, monthly income, social status(single, married...), years of experience, how many jobs has person changed, are there any other active credits and so on. Decision trees could be also applied in insurance, legal, health care and many other fields.

> **Tip:** If we want to share the results of our analysis with others, we aren't limited to giving them a copy of the iPython Notebook (.ipynb) file. We can also export the Notebook output in a form that can be opened even for those without Python installed. From the **File** menu in the upper left, go to the **Download as** submenu. You can then choose a different format that can be viewed more generally, such as HTML (.html) or PDF (.pdf). You may need additional packages or software to perform these exports.