Machine Learning Engineer Nanodegree

Introduction and Foundations

Project: 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 allows you to write easy-to-read plain text that can be converted to HTML.

In [1]:

```
# Import libraries necessary for this project
import numpy as np
import pandas as pd
from IPython.display import display # Allows the use of display() for DataF
rames

# Import supplementary visualizations code visuals.py
import visuals as vs

# Pretty display for notebooks
%matplotlib inline

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

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

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

In [2]:

```
# 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	Cabin	Emb
0	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

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 `outcomes[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 [3]:

```
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 ==
pred).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(outcomes[:5], predictions)
```

```
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 asked to make a prediction about any passenger aboard the RMS Titanic whom we knew nothing 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 (more than 50%) did not survive the ship sinking.
The `predictions_0` function below will always predict that a passenger did not survive.

In [4]:

```
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 [5]:

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

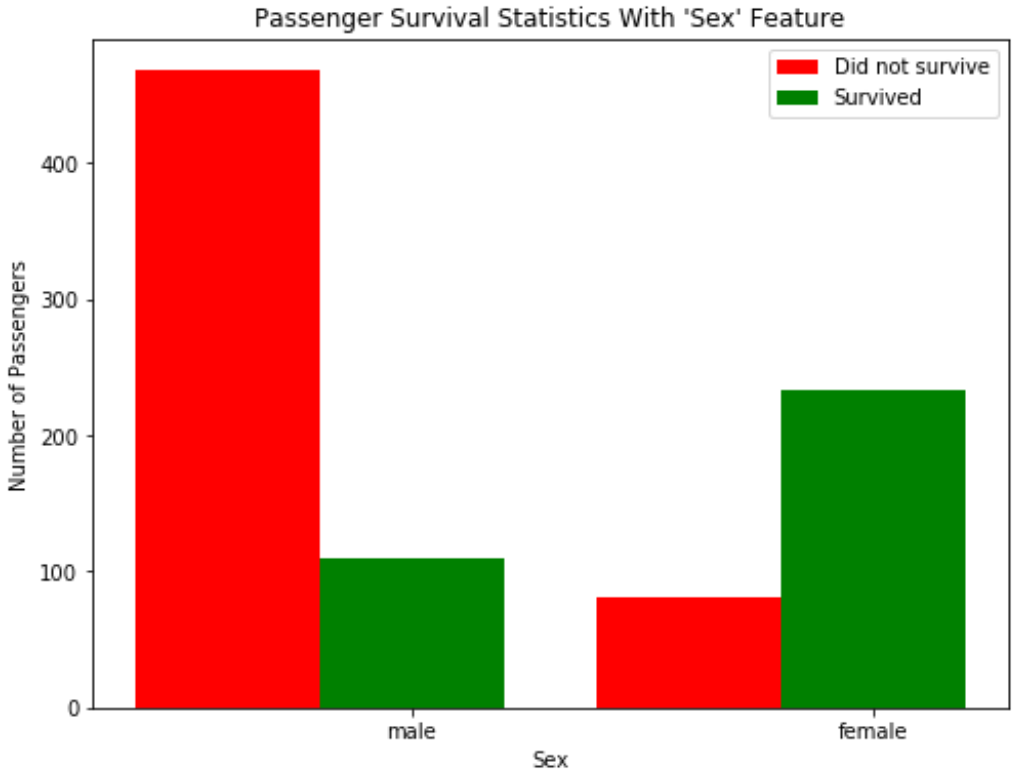
```
Predictions have an accuracy of 61.62%.
```

Answer: Accuracy = **61.62%**

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 [6]:

```
vs.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 [7]:

```
def predictions_1(data):
```

```
    """ Model with one feature:
            - Predict a passenger survived if they are female. """

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

        # Remove the 'pass' statement below
        # and write your prediction conditions here
        if passenger["Sex"] == "female":
            predictions.append(1)
        else:
            predictions.append(0)

    # 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 [8]:

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

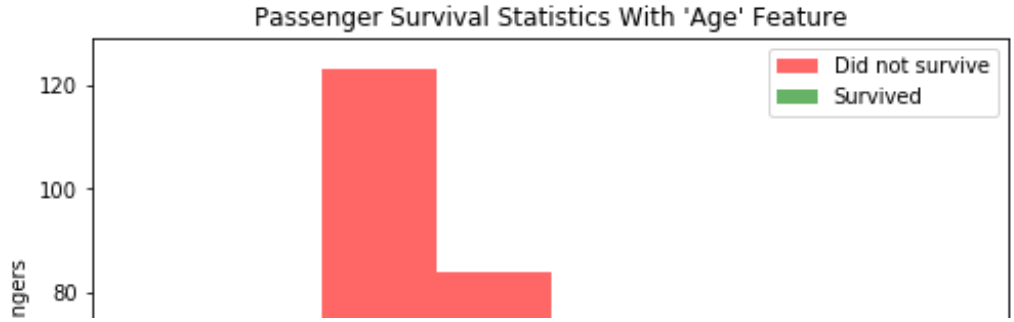
```
Predictions have an accuracy of 78.68%.
```

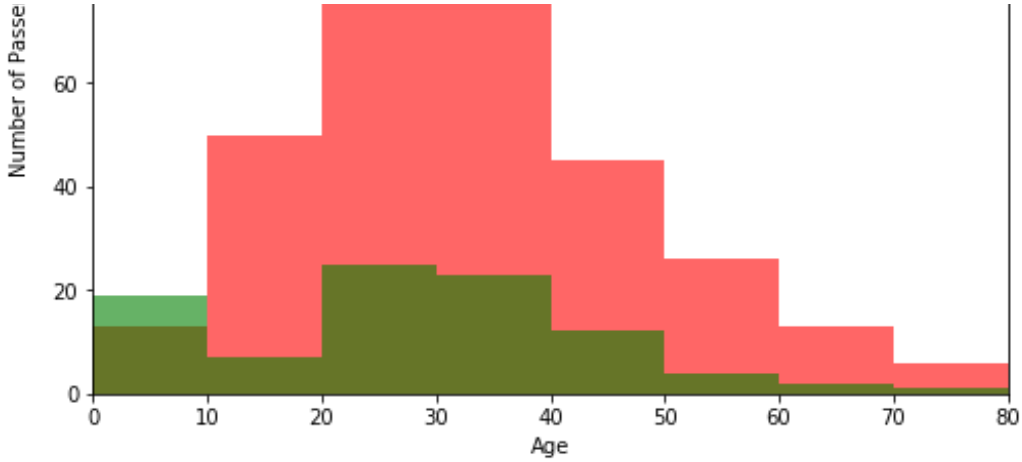
Answer: Accuracy = **78.68%**

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. For example, consider 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 [9]:

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





Examining the survival statistics, the majority of males younger than 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 [10]:

```
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 tha
n 10. """

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

        if passenger["Sex"] == "female":
            predictions.append(1)
        else:
            if passenger["Age"] < 10:
                predictions.append(1)
            else:
                predictions.append(0)

    # 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 [11]:

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

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

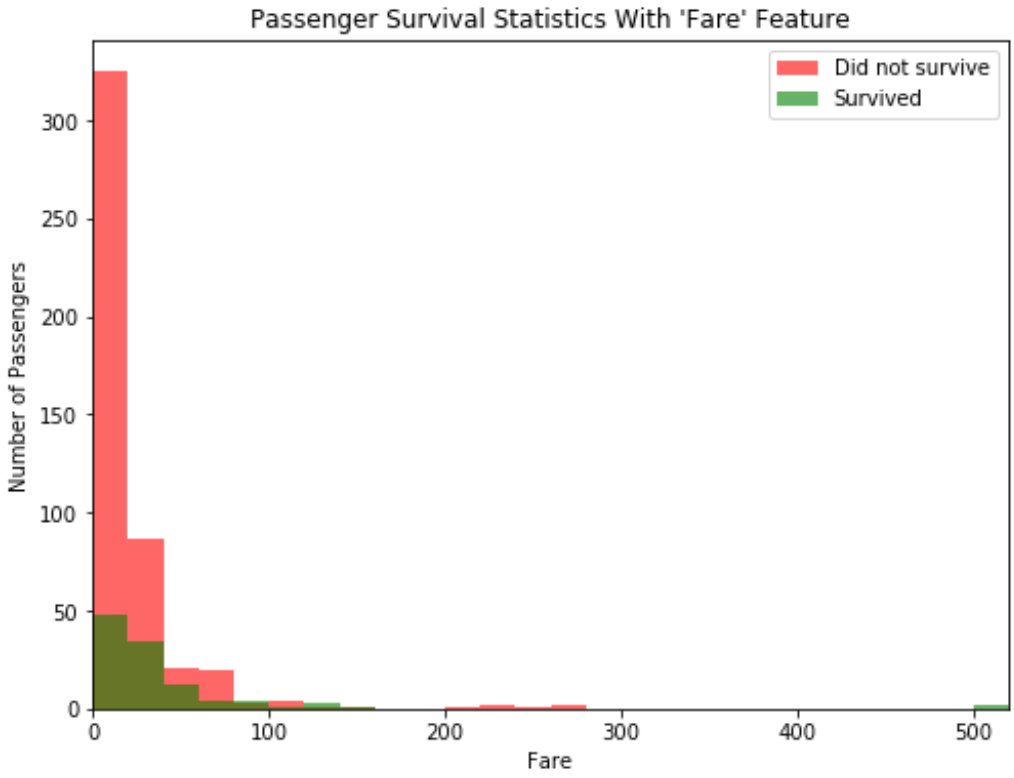
Answer: Accuracy = **79.35%**

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 [12]:

```
vs.survival_stats(data, outcomes, 'Fare', ["Sex == 'male'"])
```



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 [13]:

```
```

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

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

        if passenger["Sex"] == "female":
            if (passenger["SibSp"] not in [3, 4, 8]) and \
               (passenger["Parch"] not in [4, 5, 6]):
                predictions.append(1)
            else:
                predictions.append(0)

        else:
            if (passenger["Age"] < 10) or (passenger["Fare"] > 300):
                predictions.append(1)
            else:
                predictions.append(0)

    # 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 [14]:

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

```
Predictions have an accuracy of 81.03%.
```

Answer: In order to improve the accuracy of the model, I examined several additional features. I kept the initial breakdown of passengers into female and male categories as I found it to be highly predictive of survival, but I further segmented the data to improve the predictions. I split females by number of siblings/spouses aboard and the number of parents/children aboard. I found that females with 3, 4, or 8 siblings/spouses aboard had a lower than 50% survival rate and thus predicted that females meeting this criterion would not survive. The same approach applied for females with 4, 5, or 6 parents/children aboard the Titanic as their survival rate was also below 50%. For males, I kept the survived prediction for anyone less than 10 years old. I included the additional condition that any males who paid more than $300 for fare survived as the survival rates for those paying above this amount was greater than 50%. I found that the most predictive feature was the sex of the passenger which is why I kept the initial split based on sex. I made additional splits in the female group based on siblings/spouses aboard and parents/children aboard because these features were highly predictive of survival for female passengers. For male passengers, I split the passengers based on fare and age because these were the two most predictive features for within the male subset. Overall, the accuracy of my predictions was **81.03%**.

Conclusion

After several iterations of exploring and conditioning on the data, you have built a useful algorithm for predicting the survival of each passenger aboard the RMS Titanic. The technique applied in this project is a manual implementation of a simple machine learning model, the *decision tree*. A decision tree splits a set of data into smaller and smaller groups (called *nodes*), by one feature at a time. Each time a subset of the data is split, our predictions become more accurate if each of the resulting subgroups are more homogeneous (contain similar labels) than before. 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 provides another introduction into machine learning using a decision tree.

A decision tree is just one of many models that come from *supervised learning*. In supervised learning, we attempt to use features of the data to predict or model things with objective outcome labels. That is to say, each of our data points has a known outcome value, such as a categorical, discrete label like `'Survived'`, or a numerical, continuous value like predicting the price of a house.

Question 5

Think of a real-world scenario where supervised learning could be applied. What would be the outcome variable that you are trying to predict? Name two features about the data used in this scenario that might be helpful for making the predictions.

Answer: A real-world scenario where supervised learning could be implemented would be in the prediction of wildfires. For a given plot of land (say 1 square mile), the weather conditions and ground conditions could be the features fed into the model, and the output would be the probability that the region would catch on fire in the next week. This would be a regression problem because the output would be a continuous probability between 0 and 1, although it could be framed as a classification problem by discretizing the probabilities into a 1 or 0. Any probabilites of fire greater than 0.5 would be assigned a 1 (high threat of fire and response needed) and probabilites lower than 0.5 would be labeled a 0 (low threat of fire and no response needed). This is a supervised learning problem because the data fed into the model would be the historical weather and ground conditions that preceded a fire. The historical data would be used to train the model, and then the model would make predictions based on real-time information for given regions (square mile plots). One feature that could be helpful in predicting whether or not a fire will occur in an area would be the amount of rain the area received in the previous two weeks. A higher rainfall level would decrease the chance of a fire. Moreover, the dry vegetation cover on the ground could be a useful predictor of wildfires. Regions with a higher percentage of vegetation cover would have a higher risk for fire. Combining these features would allow for accurate predictions of the wildfire threat.

More information about using machine learning to predict wildfires can be found in "A Data Mining Approach to Predict Forest Fires using Meteorlogical Data" and Historical Wildfire Data can be found through the National Oceanic and Atmospheric Administration (NOAA).

> **Note**: Once you have completed all of the code implementations and successfully answered each question above, you may finalize your work by exporting the iPython Notebook as an HTML document. You can do this by using the menu above and navigating to
> **File -> Download as -> HTML (.html)**. Include the finished document along with this notebook as your submission.