Introduction:

In this project I will be analyzing the titanic dataset that I obtained from kaggle. I will take a deep look into it and see what factors were involved in deciding the fate for the passengers on board. In addition to creating visualizations I will also be running various hypothesis tests to get a more clear representation for how certain features correlate and affect others. At the end, I will be creating a logistic regression classification model and a boosting model to predict whether or not a passenger will live or not depending on various features.

In [1]:
```
# importing the needed dependencies
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
from math import ceil
```

In [2]:
```
# importing the data set.
df = pd.read_csv('train.csv')
print('This data set contains', len(df), 'observations.')
df.head(3)
```

```
This data set contains 891 observations.
```

Out[2]:

	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

Some features are not necessary for this project:

I will start cleaning up this data set by removing the features that are not needed.

In [3]:
```
columns_to_keep = ['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Fare']
df = df[columns_to_keep]
df.head(3)
```

Out[3]:

	Survived	Pclass	Name	Sex	Age	Fare
0	0	3	Braund, Mr. Owen Harris	male	22.0	7.2500
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	71.2833
2	1	3	Heikkinen, Miss. Laina	female	26.0	7.9250

In [4]:
```
# taking a look at the data types.
df.dtypes
```

Out[4]:
```
Survived      int64
Pclass        int64
Name         object
Sex          object
Age         float64
Fare        float64
dtype: object
```

In [5]:
```
# Now lets look to see how many missing values there are.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   Fare      891 non-null    float64
dtypes: float64(2), int64(2), object(2)
memory usage: 41.9+ KB
```

Handling the missing values:

The only column that has missing values is the 'Age' column. It is only missing 177 values which isn't an extremely large amount for the size of this data set. Therefore I will impute the missing values with the average age.

One more small edit:

The fare column has too many floating numbers onto it. This doesn't really make sense since it is supposed to represent dollars. Therefore I will remove it to two decimal places.

In [6]:
```
# Filling in the missing age categories with the average age.
df.fillna(value={'Age': np.mean(df.Age)}, inplace=True)
df.Age = df.Age.apply(ceil)
```

In [7]:
```
# changing the fare so it looks more like normal currency.
df.Fare = np.round(df.Fare, decimals=2)
```

Feature engineering:

The feature that I will be adding to this data set is the passengers maritial status. I feel like it might be easier for an individual (especially man) to be saved if he is with his wife rather than being alone. The way I will do this is by using the name column and some simple research.

If the name column has 'Mrs.' then its known that the women is married. Unfortunately that is the only name title that gives away this information. For the men and the other women with ambiguous titles I will be guessing whether or not they are married.

According to google, the average age for marriage for men in the 1910's was 25 years old. For women, the age was 22. Using this information I will create another column to input the individuals maritial status.

In [8]:
```
# First lets look at how we need to extract the passengers title.
df.Name[:5]
```

Out[8]:
```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
Name: Name, dtype: object
```

In [9]:
```
import re
df2 = df.Name.str.extract('(,.+\.)')[0].str.lstrip(',').str.rstrip('.').str.strip(' ')
df2.name = 'Title'
```

In [10]:
```
df = pd.concat([df, df2], axis=1).drop(columns=['Name'])
```

In [11]:
```
df.head(3)
```

Out[11]:

	Survived	Pclass	Sex	Age	Fare	Title
0	0	3	male	22	7.25	Mr
1	1	1	female	38	71.28	Mrs
2	1	3	female	26	7.92	Miss

In [12]:
```
# Creating the column to show maritial status.
df['Married'] = df.apply(
    lambda row: 1 if (row['Age'] >= 22 and row['Sex'] == 'female'
    and (row['Title'] == 'Mrs' or row['Title'] != 'Miss'))
    or (row['Age'] >= 25 and row['Sex'] == 'male')
    else 0, axis=1
)
df.drop(columns=['Title'], axis=1, inplace=True)
```

In [13]:
```
# 0 is for single and 1 is for married.
df.head(3)
```

Out[13]:

	Survived	Pclass	Sex	Age	Fare	Married
0	0	3	male	22	7.25	0
1	1	1	female	38	71.28	1
2	1	3	female	26	7.92	0

Some more Feature engineering:

The next feature that I will add to the data set is based off of age. I will bin the ages and create categorical variables based off of them. For anybody under 6 years old I will label them as a baby. For anybody from 6-12 I will label them as a child. For 13-18 They will be a teenager. For 19-59 They will be an adult. For 60 onwards they will be labeled as a senior.

In [14]:
```
# creating this new feature:
df['Bin_age'] = pd.cut(df.Age, bins=[0,5,12,18,59,100], labels=['Baby', 'Child', 'Teenager', 'Adult', 'Senior'])
df.head(3)
```

Out[14]:

	Survived	Pclass	Sex	Age	Fare	Married	Bin_age
0	0	3	male	22	7.25	0	Adult
1	1	1	female	38	71.28	1	Adult
2	1	3	female	26	7.92	0	Adult

Cleaning the data is done.. Now onto the analysis:

First, I will be doing some multivariate analysis on gender and the survival column.

In [15]:
```
# How does survived and gender compare to eachother?
print('Comparing the number of people and their gender to living or dying: \n')
print(pd.crosstab(df.Survived.map({0: 'Died', 1: 'Lived'}), df.Sex))
print('\nComparing the percentage of people and their gender to living or dying: \n')
ct_percentage = pd.crosstab(df.Survived.map({0: 'Died', 1: 'Lived'}), df.Sex, normalize=True)
print(round(ct_percentage * 100, 2))
```

```
Comparing the number of people and their gender to living or dying:

Sex       female  male
Survived
Died          81   468
Lived        233   109

Comparing the percentage of people and their gender to living or dying:

Sex       female   male
Survived
Died        9.09  52.53
Lived      26.15  12.23
```

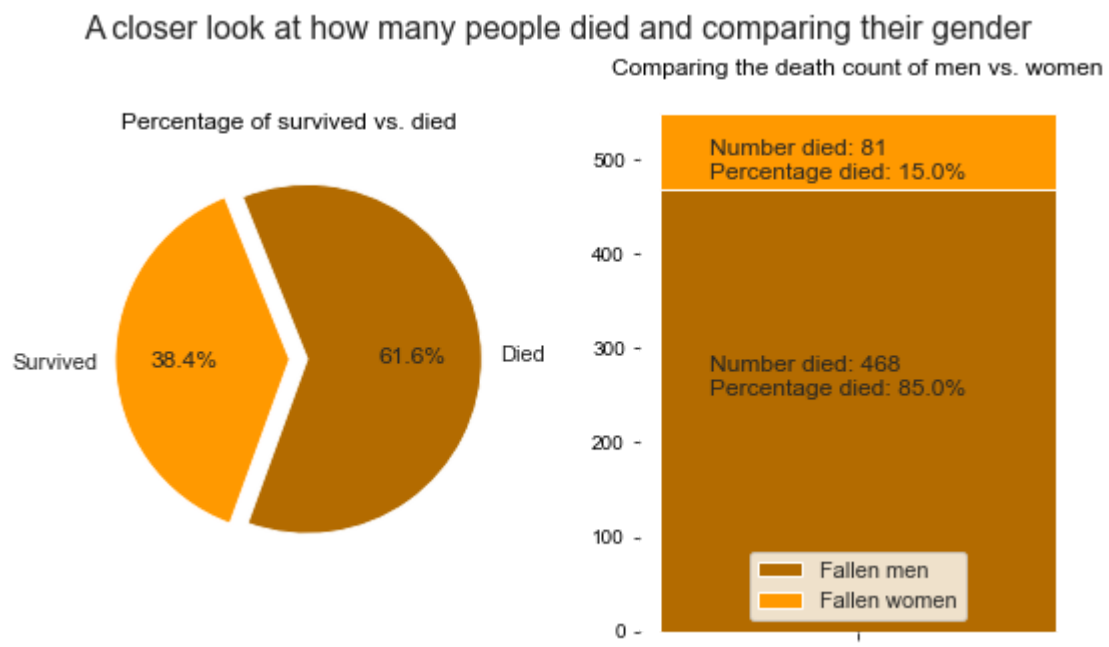
In [16]:

```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(9,5))
fig.subplots_adjust(wspace=0.3)
sns.set()

ratios = df.Survived.value_counts(normalize=True).values
labels = ['Died', 'Survived']
explode = [0.1, 0]
color = ['#b36b00', '#ff9900']
ax1.pie(ratios, autopct='%1.1f%%', startangle=250, labels=labels, explode=explode, colors=color)
ax1.set_title('Percentage of survived vs. died')

fallen_men = sum((df.Survived == 0) & (df.Sex == 'male'))
fallen_women = sum((df.Survived == 0) & (df.Sex == 'female'))
sns.set_style('white')
ax2.bar(' ', fallen_men, label='Fallen men', color='#b36b00')
ax2.text(-0.3,250,
         f'Number died: {fallen_men} \nPercentage died: {(round(fallen_men/np.sum([fallen_men, fallen_women]),2)
        fontsize=12)
ax2.bar(' ', fallen_women, bottom=fallen_men, label='Fallen women', color='#ff9900')
ax2.text(-0.3,480,
         f'Number died: {fallen_women} \nPercentage died: {(round(fallen_women/np.sum([fallen_men, fallen_women]
        fontsize=12)
ax2.legend(loc='lower center')
sns.despine(left=True, bottom=True)
ax2.set_title('Comparing the death count of men vs. women')
plt.suptitle('A closer look at how many people died and comparing their gender', fontsize=16)

plt.show()
```



No hypothesis testing needed here..

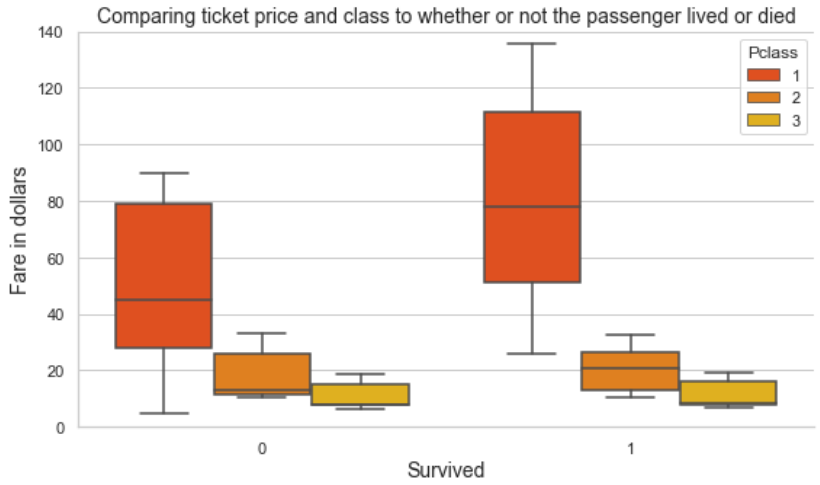
It's pretty obvious to see that gender played the biggest role in deciding an individuals fate.

Some multivariate analysis:

Next, I will be adding in the 'Pclass' and 'Fare' column to the gender and surivival column to do some multivarious analysis. The 'Pclass' represents the class that the ticket belonged to. Pclass 1 is the most prestigious and 3 is the least.

In [18]:
```
plt.subplots(figsize=(20,5))
plt.subplot(1,2,1)
sns.set_style('whitegrid')
sns.boxplot(data=df, y='Fare', x='Survived', hue='Pclass', whis=.5, fliersize=False, palette='autumn')
plt.ylim(0,140)
plt.ylabel('Fare in dollars', fontsize=14)
plt.xlabel('Survived', fontsize=14)
plt.title('Comparing ticket price and class to whether or not the passenger lived or died', fontsize=14)
sns.despine()

plt.subplot(1,2,2)
sns.set_style('whitegrid')
sns.stripplot(data=df, y='Fare', x='Survived', hue='Sex', palette='autumn', dodge=True, size=3)
plt.ylim(0,140)
plt.ylabel('Fare in dollars', fontsize=14)
plt.xlabel('Survived', fontsize=14)
plt.title('Comparing ticket price and gender to whether or not the passenger lived or died', fontsize=14)
sns.despine()
plt.show()
```



Still a bit hard to tell if 'Pclass' or 'Fare price' has anything to do with survival chance.

Therefore I will be running some hypothesis testing to see if there is signifance between gender and fare price. I will use a two sample t-test to compare the fare price paid for those who died and those who survived.

In [20]:
```
# Using the fare price paid for both men and women.
from scipy.stats import ttest_ind
t, p = ttest_ind(df.Fare[df.Survived==0], df.Fare[df.Survived==1])
print('Comparing significance for fare price paid by survivors and those who did not. Gender is combined:')
print(p)

# what about for the men?
t, p = ttest_ind(df.Fare[(df.Survived==0) & (df.Sex=='male')], df.Fare[(df.Survived==1) & (df.Sex=='male')])
print('Doing the same but now for only the men:')
print(p)

# what about for just women?
t, p = ttest_ind(df.Fare[(df.Survived==0) & (df.Sex=='female')], df.Fare[(df.Survived==1) & (df.Sex=='female')])
print('And now only for the women:')
print(p)
```

```
Comparing significance for fare price paid by survivors and those who did not. Gender is combined:
6.1231427673095645e-15
Doing the same but now for only the men:
3.531526324937067e-05
And now only for the women:
9.502679461899834e-05
```

The results:

Null hypothesis states that fare paid by the survivors and the ones who died has nothing to do with eachother. The results say otherwise for all genders. Therefore I will be accepting the alternate hypothesis. This means that how much an individual paid for their ticket helped decide their fate. The more wealthy people had a greater chance of survivor regardless of gender.

Chi Squared hypothesis test:

Now I will do a chi2 test to see if there is any way that age has anything to do with deciding the fate of a passenger. The null hypothesis is that it does not. First I will do a contingency test between the binned ages and survival. Next I will do the chi2 test to look at the pval.

In [21]:
```
from scipy.stats import chi2_contingency
ct = pd.crosstab(df.Bin_age, df.Survived)
chi2, pval, dof, expected = chi2_contingency(ct)
print('Pval:', pval)
print('\t\t\t\t\t Actual outcome  Expected outcome')
for x,y,z in zip(ct.index,ct.values,np.round(expected)):
    print('Age range: ',x,'\t[Died Lived]\t',y, '\t',z)
```

```
Pval: 0.0001731428976526032
                                        Actual outcome  Expected outcome
Age range:  Baby        [Died Lived]    [13 31]         [27. 17.]
Age range:  Child       [Died Lived]    [16  9]         [15. 10.]
Age range:  Teenager    [Died Lived]    [40 30]         [43. 27.]
Age range:  Adult       [Died Lived]    [461 265]       [447. 279.]
Age range:  Senior      [Died Lived]    [19  7]         [16. 10.]
```

Besides gender, age also had to do with deciding fate:

According to the pval it looks like age also played a role in deciding who lived. Babies had better odds at survivor regardless of gender and socioeconomic class. Unfortunately it looks like children were not so lucky. Babies were saved, everybody else, not so much.

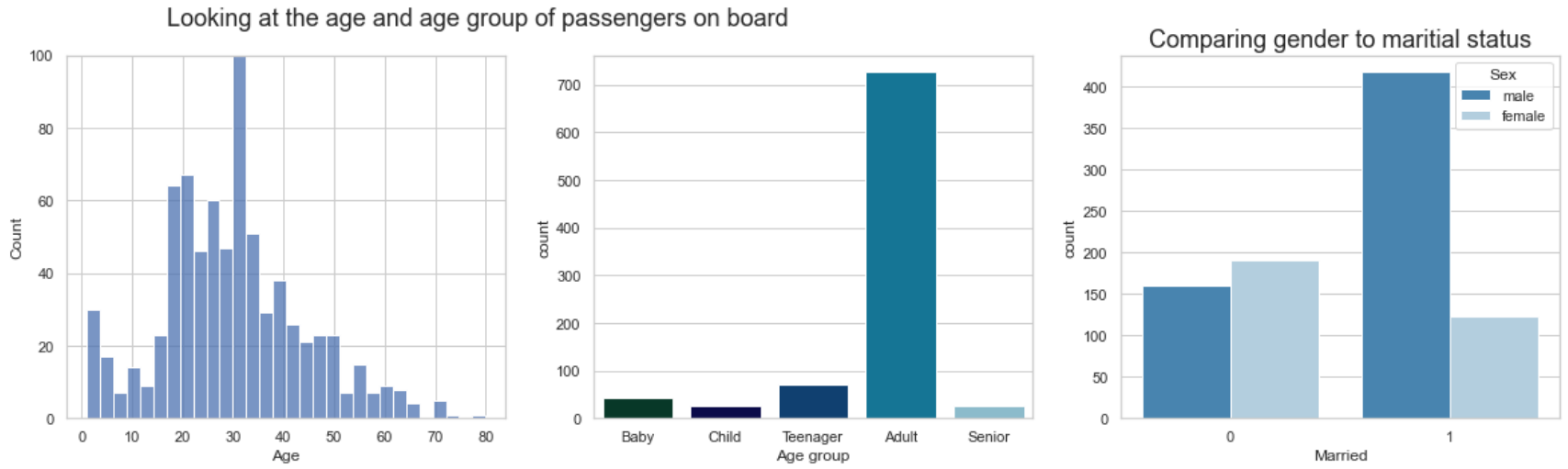
A few more visuals:

Let's take a look at a few more visuals.

In [22]:
```
plt.subplots(figsize=(20,5))
plt.subplot(1,3,1)
sns.histplot(data=df, x='Age')
plt.ylim(0,100)

plt.subplot(1,3,2)
sns.countplot(data=df, x='Bin_age', palette='ocean')
plt.suptitle('Looking at the age and age group of passengers on board', fontsize=18, ha='right')
plt.xlabel('Age group')

plt.subplot(1,3,3)
sns.countplot(data=df, x='Married', hue='Sex', palette='Blues_r')
plt.title('Comparing gender to maritial status', fontsize=18)
plt.show()
```



Machine Learning section:

In this final section I will be creating a logistical classification model and an Ada boosting classification model to predict whether an individual would survive the titanic voyage or not.

In [23]:
```
# first I will get the values all ready to implement into a ml algorithm.
# changing gender into binary:
df.Sex = df.Sex.map({'male': 0, 'female': 1})

# next I will standardize the age and fare features.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df.Age = scaler.fit_transform(df.Age.values.reshape(-1,1))

df.Fare = scaler.fit_transform(df.Fare.values.reshape(-1,1))

# One Hot Encoding the Bin_age column:
df = pd.get_dummies(data=df, columns=['Bin_age'])
```

In [24]:
```
df.head(3)
```

Out[24]:

	Survived	Pclass	Sex	Age	Fare	Married	Bin_age_Baby	Bin_age_Child	Bin_age_Teenager	Bin_age_Adult	Bin_age_Senior
0	0	3	0	-0.597994	-0.502449	0	0	0	0	1	0
1	1	1	1	0.633232	0.786776	1	0	0	0	1	0
2	1	3	1	-0.290188	-0.488958	0	0	0	0	1	0

In [25]:
```
# now its time to split it into X and y:
X = df.iloc[:, 1:]
y = df.iloc[:, 0]
```

In [26]:
```
# now its time to split into a train and test set:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X.values, y.values, train_size=0.85, test_size=0.15, random_
```

In [27]:
```
# first will try with a standard logistic regressor classifier:
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression().fit(x_train, y_train)
lr_predict = lr.predict(x_test)
```

In [28]:
```
from sklearn.metrics import accuracy_score, recall_score, precision_score, confusion_matrix
print('Accuracy score:\t\t', accuracy_score(y_test, lr_predict))
print('Recall score:\t\t', recall_score(y_test, lr_predict))
print('Precision score:\t', precision_score(y_test, lr_predict))
```

```
Accuracy score:          0.7388059701492538
Recall score:            0.6
Precision score:         0.6666666666666666
```

In [29]:
```
print(confusion_matrix(y_test, lr_predict))
print('It predicted that  a lot more people would survive than really.')
```

```
[[69 15]
 [20 30]]
It predicted that  a lot more people would survive than really.
```

In [30]:
```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(random_state=22, n_estimators=200, learning_rate=0.1).fit(x_train, y_train)
ada_predict = ada.predict(x_test)

print('Accuracy score:\t\t', accuracy_score(y_test, ada_predict))
print('Recall score:\t\t', recall_score(y_test, ada_predict))
print('Precision score:\t', precision_score(y_test, ada_predict))
print(confusion_matrix(y_test, ada_predict))
```

```
Accuracy score:          0.7313432835820896
Recall score:            0.62
Precision score:         0.6458333333333334
[[67 17]
 [19 31]]
```

Final thoughts about the algorithm and project in general:

The accuracy score for both the models are mediocre with similar predictive power at 73%. Looking at the confusion matrix it looks like the logistic regressor model does a bit better with the false positives and a little worse with the false negatives than the boosting algorithm. However, overall the logistic regressor performs a little bit better. This is a bit surprising to me but I think that if I were to tweak the Ada Boosting model some more then I could find a combination that outperforms the simple logistic regression model.

Overall, I found dissecting the data set very amusing. I always knew that gender and age played an important role in deciding who lived and died but did not know it was so extreme as my findings show. I also found it interesting that socio economics in regards to passenger income also played a role in survival regardless of gender. I can only imagine what kind of bribes and offers were being made in that unfortunate time.