Data Science Project Report

Survival of Passengers in Titanic Dataset

Under the guidance
of
Mr. Sumit Kumar Shukla

Submitted by: Sharath C Nair
Rajadhani Institute of Engineering and Technology,
Trivandrum Kerala

EITHICAL EDUFABRICA (2020)

Contents

Acknowledgement

I would like to express my sincere gratitude to my supervisor Mr. Sumit Kumar Shukla for providing their invaluable guidance, comments and suggestions throughout the course of the project. I would specially thank Mr. Sumit Kumar Sir for getting me the iris dataset for my exploration and successful completion of my project.

About Titanic Dataset

It is one of the most popular datasets used for understanding machine learning basics. It contains information of all the passengers aboard the RMS Titanic, which unfortunately, was shipwrecked. This dataset can be used to predict whether a given passenger survived or not.

Importing libraries and loading the file.

```
In [2]: import numpy as np
        import pandas as pd
        from sklearn import preprocessing
        import matplotlib.pyplot as plt
        plt.rc("font",size=14)
        import seaborn as sns
        sns.set(style="white")#white background style for seaborn plots
        sns.set(style="whitegrid", color_codes=True)
```

Let's load the data in a data frame and check how data looks like..

```
In [3]: # Read CSV train data file into DataFrame
        train_df = pd.read_csv("titanic_train.csv")

        # Read CSV test data file into DataFrame
        test_df = pd.read_csv("titanic_test.csv")

        # preview train data
        train_df.head()
```

Out[3]:

	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
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Observations-

1. As we can see above the data set has 12 columns or features. We already discussed what these features are all about earlier.

2. Here the Survived column is the target variable or class label. Target variable is the feature which needs to be predicted by our models.

3. We have numerical , categorical Type of features.

Fetch some info about data

In [4]:
```
print('The number of samples into the train data is {}.'.format(train_df.shape[0]))
```

```
The number of samples into the train data is 891.
```

In [5]:
```
test_df.head()
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [6]:
```
print('The number of samples into the test data is {}.'.format(test_df.shape[0]))
```

```
The number of samples into the test data is 418.
```

Note: there is no target variable into test data (i.e. "Survival" column is missing), so the goal is to predict this target using different machine learning algorithms such as logistic regression.

Data Quality & Missing Value Assessment¶

In [8]:
```
#Check missing values in train data
train_df.isnull().sum()
```

Out[8]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

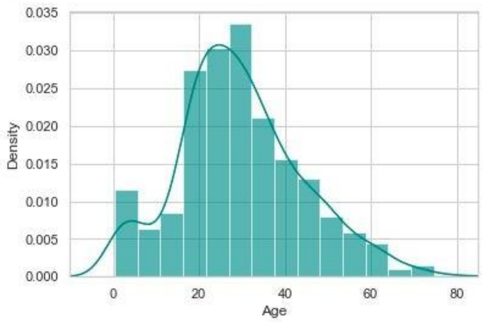
Percentage of Missing age record:

```
In [9]: # percent of missing "Age"
        print('Percent of missing "Age" records is %.2f%%' %((train_df['Age'].isnull().sum()/train_df.shape[0])*100))
```

```
Percent of missing "Age" records is 19.87%
```

~20% of entries for passenger age are missing. Let's see what the 'Age' variable looks like in general.

```
In [10]: ax = train_df["Age"].hist(bins=15, density=True, stacked=True, color='teal', alpha=0.6)
         train_df["Age"].plot(kind='density', color='teal')
         ax.set(xlabel='Age')
         plt.xlim(-10,85)
         plt.show()
```



Since "Age" is (right) skewed, using the mean might give us biased results by filling in ages that are older than desired. To deal with this, we'll use the median to impute the missing values.

```
In [12]: # mean age
         print('The mean of "Age" is %.2f' %(train_df["Age"].mean(skipna=True)))
         # median age
         print('The median of "Age" is %.2f' %(train_df["Age"].median(skipna=True)))
```

```
The mean of "Age" is 29.70
The median of "Age" is 28.00
```

```
In [13]: print('Percent of missing "Cabin" records is %.2f%%' %((train_df['Cabin'].isnull().sum()/train_df.shape[0])*100))
```

```
Percent of missing "Cabin" records is 77.10%
```

77% of records are missing, which means that imputing information and using this variable for prediction is probably not wise. We'll ignore this variable in our model.

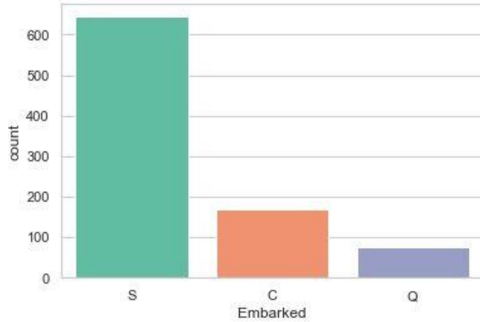
```
In [14]: # percent of missing "Embarked"
         print('Percent of missing "Embarked" records is %.2f%%' %((train_df['Embarked'].isnull().sum()/train_df.shape[0])*1(
```

```
Percent of missing "Embarked" records is 0.22%
```

There are only 2 (0.22%) missing values for "Embarked", so we can just impute with the port where most people boarded.

```
In [15]: print('Boarded passengers grouped by port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton):')
         print(train_df['Embarked'].value_counts())
         sns.countplot(x='Embarked', data=train_df, palette='Set2')
         plt.show()
```

```
Boarded passengers grouped by port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton):
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```




```
In [16]: print('The most common boarding port of embarkation is %s.' %train_df['Embarked'].value_counts().idxmax())
```

```
The most common boarding port of embarkation is S.
```

By far the most passengers boarded in Southhampton, so we'll impute those 2 NaN's w/ "S".

Final Adjustments to Data (Train & Test)

```
In [17]: train_data = train_df.copy()
         train_data["Age"].fillna(train_df["Age"].median(skipna=True), inplace=True)
         train_data["Embarked"].fillna(train_df['Embarked'].value_counts().idxmax(), inplace=True)
         train_data.drop('Cabin', axis=1, inplace=True)
```

check missing values in adjusted train data

```
In [18]: # check missing values in adjusted train data
         train_data.isnull().sum()
```

```
Out[18]: PassengerId    0
         Survived       0
         Pclass         0
         Name           0
         Sex            0
         Age            0
         SibSp          0
         Parch          0
         Ticket         0
         Fare           0
         Embarked       0
         dtype: int64
```

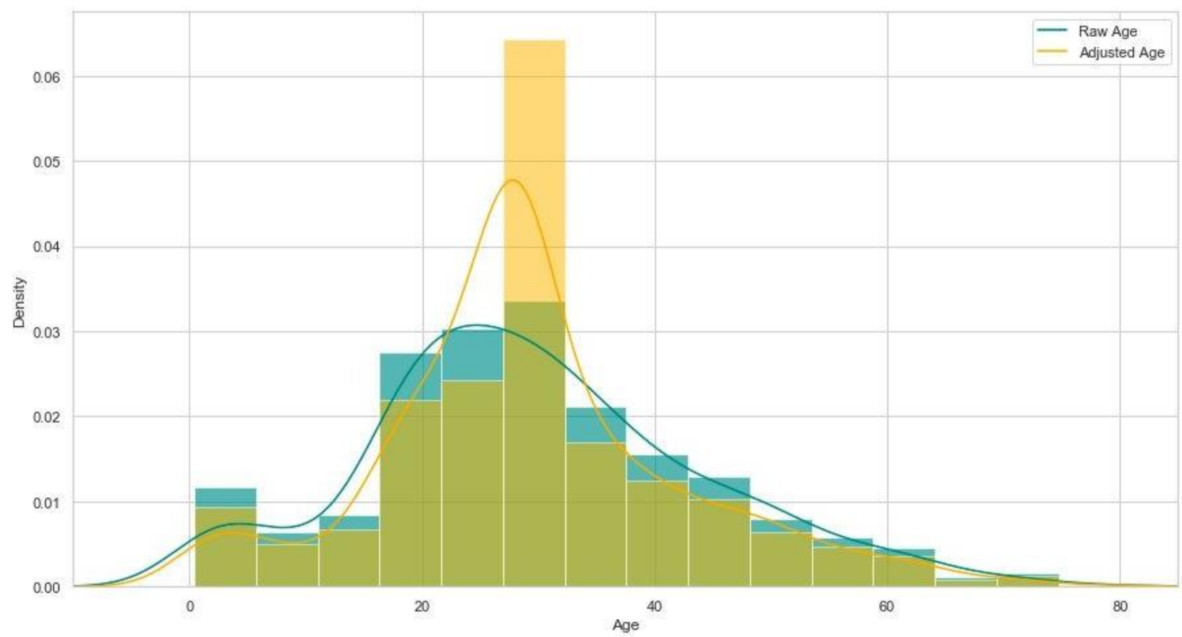
preview adjusted train data

```
In [19]: # preview adjusted train data
         train_data.head()
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
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	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

```
In [20]: plt.figure(figsize=(15,8))
         ax = train_df["Age"].hist(bins=15, density=True, stacked=True, color='teal', alpha=0.6)
         train_df["Age"].plot(kind='density', color='teal')
         ax = train_data["Age"].hist(bins=15, density=True, stacked=True, color='orange', alpha=0.5)
         train_data["Age"].plot(kind='density', color='orange')
         ax.legend(['Raw Age', 'Adjusted Age'])
         ax.set(xlabel='Age')
         plt.xlim(-10,85)
         plt.show()
```



Create categorical variable for traveling alone

```
In [21]: ## Create categorical variable for traveling alone
         train_data['TravelAlone']=np.where((train_data["SibSp"]+train_data["Parch"])>0, 0, 1)
         train_data.drop('SibSp', axis=1, inplace=True)
         train_data.drop('Parch', axis=1, inplace=True)
```

I'll also create categorical variables for Passenger Class ("Pclass"), Gender ("Sex"), and Port Embarked ("Embarked").

```
In [22]: #create categorical variables and drop some variables
         training=pd.get_dummies(train_data, columns=["Pclass","Embarked","Sex"])
         training.drop('Sex_female', axis=1, inplace=True)
         training.drop('PassengerId', axis=1, inplace=True)
         training.drop('Name', axis=1, inplace=True)
         training.drop('Ticket', axis=1, inplace=True)

         final_train = training
         final_train.head()
```

Out[22]:

	Survived	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S	Sex_male
0	0	22.0	7.2500	0	0	0	1	0	0	1	1
1	1	38.0	71.2833	0	1	0	0	1	0	0	0
2	1	26.0	7.9250	1	0	0	1	0	0	1	0
3	1	35.0	53.1000	0	1	0	0	0	0	1	0
4	0	35.0	8.0500	1	0	0	1	0	0	1	1

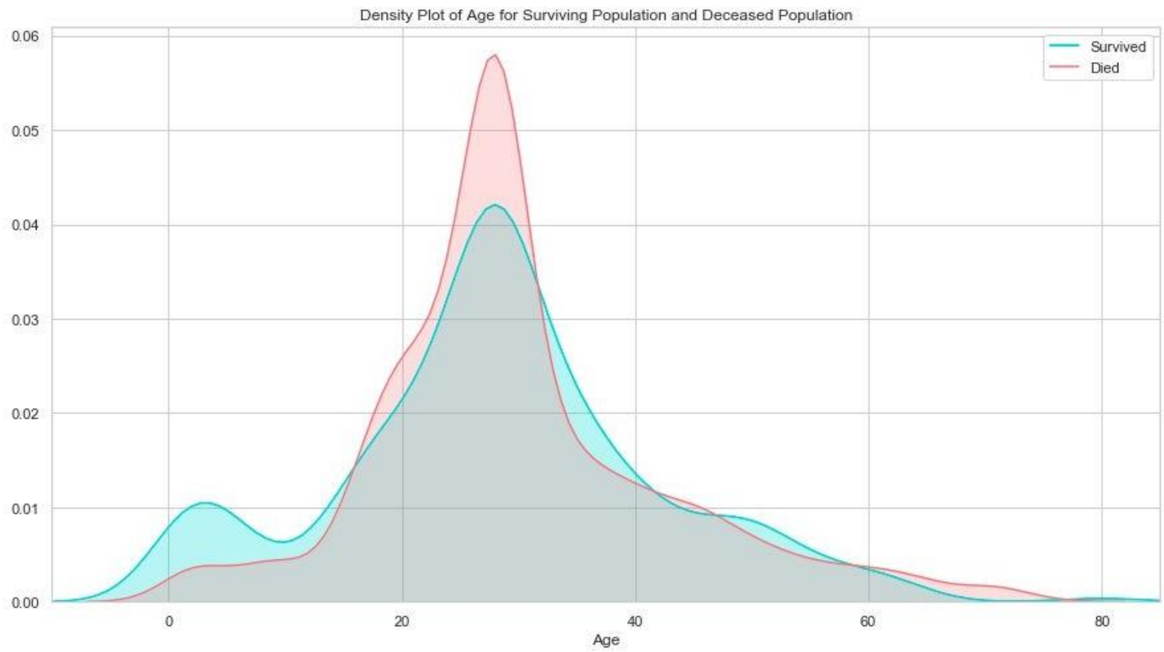
```
In [24]: test_data = test_df.copy()
         test_data["Age"].fillna(train_df["Age"].median(skipna=True), inplace=True)
         test_data["Fare"].fillna(train_df["Fare"].median(skipna=True), inplace=True)
         test_data.drop('Cabin', axis=1, inplace=True)

         test_data['TravelAlone']=np.where((test_data["SibSp"]+test_data["Parch"])>0, 0, 1)

         test_data.drop('SibSp', axis=1, inplace=True)
         test_data.drop('Parch', axis=1, inplace=True)

         testing = pd.get_dummies(test_data, columns=["Pclass","Embarked","Sex"])
         testing.drop('Sex_female', axis=1, inplace=True)
         testing.drop('PassengerId', axis=1, inplace=True)
         testing.drop('Name', axis=1, inplace=True)
         testing.drop('Ticket', axis=1, inplace=True)

         final_test = testing
         final_test.head()
```

Out[24]:

	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S	Sex_male
0	34.5	7.8292	1	0	0	1	0	1	0	1
1	47.0	7.0000	0	0	0	1	0	0	1	0
2	62.0	9.6875	1	0	1	0	0	1	0	1
3	27.0	8.6625	1	0	0	1	0	0	1	1
4	22.0	12.2875	0	0	0	1	0	0	1	0

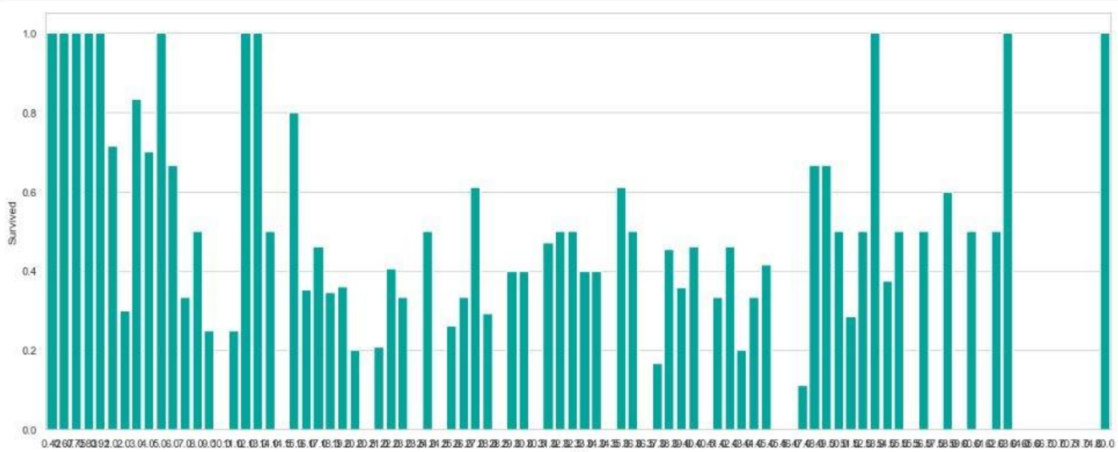
Exploration of Age:-

```
In [25]: plt.figure(figsize=(15,8))
         ax = sns.kdeplot(final_train["Age"][final_train.Survived == 1], color="darkturquoise", shade=True)
         sns.kdeplot(final_train["Age"][final_train.Survived == 0], color="lightcoral", shade=True)
         plt.legend(['Survived', 'Died'])
         plt.title('Density Plot of Age for Surviving Population and Deceased Population')
         ax.set(xlabel='Age')
         plt.xlim(-10,85)
         plt.show()
```



The age distribution for survivors and deceased is actually very similar. One notable difference is that, of the survivors, a larger proportion were children. The passengers evidently made an attempt to save children by giving them a place on the life rafts.

```
In [26]: plt.figure(figsize=(20,8))
         avg_survival_byage = final_train[["Age", "Survived"]].groupby(['Age'], as_index=False).mean()
         g = sns.barplot(x='Age', y='Survived', data=avg_survival_byage, color="LightSeaGreen")
         plt.show()
```

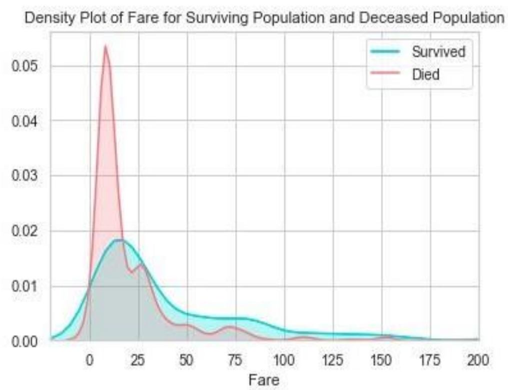


Considering the survival rate of passengers under 16, I'll also include another categorical variable in my dataset: "Minor"

```
In [27]: final_train['IsMinor']=np.where(final_train['Age']<=16, 1, 0)

         final_test['IsMinor']=np.where(final_test['Age']<=16, 1, 0)
```

Exploration of Fare

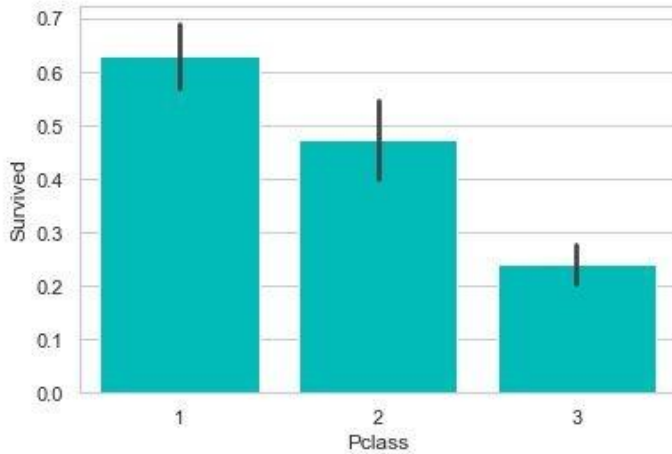
```
In [28]: ax = sns.kdeplot(final_train["Fare"][final_train.Survived == 1], color="darkturquoise", shade=True)
         sns.kdeplot(final_train["Fare"][final_train.Survived == 0], color="lightcoral", shade=True)
         plt.legend(['Survived', 'Died'])
         plt.title('Density Plot of Fare for Surviving Population and Deceased Population')
         ax.set(xlabel='Fare')
         plt.xlim(-20,200)
         plt.show()
```



As the distributions are clearly different for the fares of survivors vs. deceased, it's likely that this would be a significant predictor in our final model. Passengers who paid lower fare appear to have been less likely to survive. This is probably strongly correlated with Passenger Class, which we'll look at next.

Exploration of Passenger Class

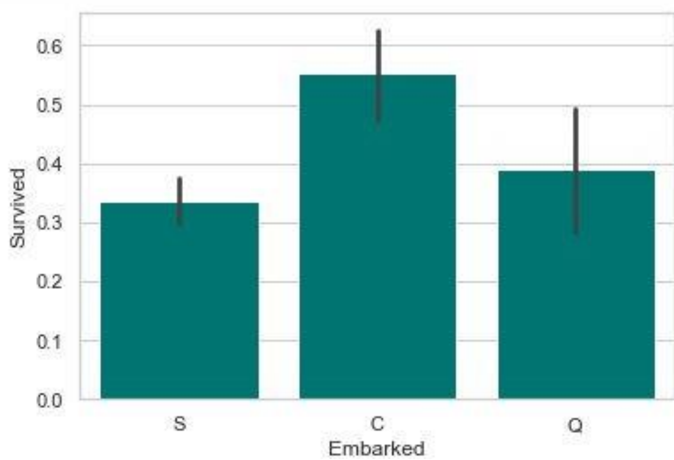
```
In [29]: sns.barplot('Pclass', 'Survived', data=train_df, color="darkturquoise")
         plt.show()
```



Unsurprisingly, being a first class passenger was safest.

Exploration of Embarked Port:-

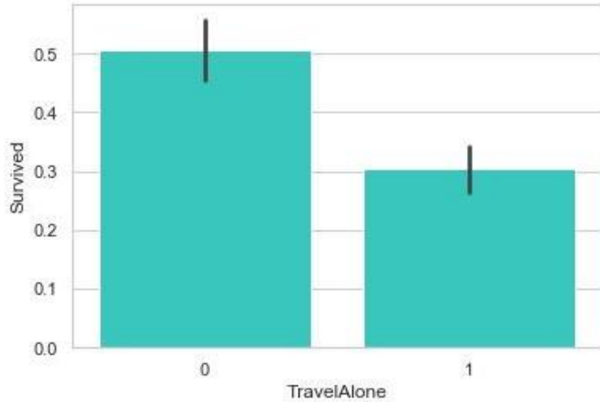
```
In [30]: sns.barplot('Embarked', 'Survived', data=train_df, color="teal")
         plt.show()
```



Passengers who boarded in Cherbourg, France, appear to have the highest survival rate. Passengers who boarded in Southampton were marginally less likely to survive than those who boarded in Queenstown. This is probably related to passenger class, or maybe even the order of room assignments (e.g. maybe earlier passengers were more likely to have rooms closer to deck). It's also worth noting the size of the whiskers in these plots. Because the number of passengers who boarded at Southampton was highest, the confidence around the survival rate is the highest. The whisker of the Queenstown plot includes the Southhampton average, as well as the lower bound of its whisker. It's possible that Queenstown passengers were equally, or even more, ill-fated than their Southampton counterparts.

Exploration of Traveling Alone vs. With Family:-

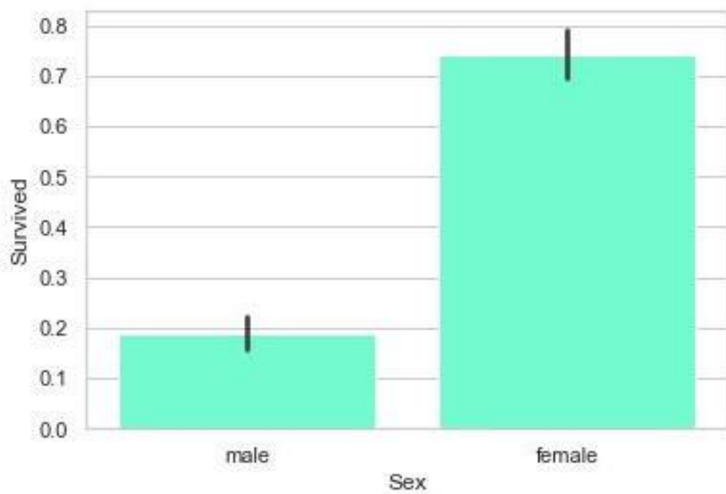
```
In [31]: sns.barplot('TravelAlone', 'Survived', data=final_train, color="mediumturquoise")
         plt.show()
```



Individuals traveling without family were more likely to die in the disaster than those with family aboard. Given the era, it's likely that individuals traveling alone were likely male.

Exploration of Gender Variable:-

```
In [32]: sns.barplot('Sex', 'Survived', data=train_df, color="aquamarine")
         plt.show()
```



This is a very obvious difference. Clearly being female greatly increased your chances of survival.

Logistic Regression and Results

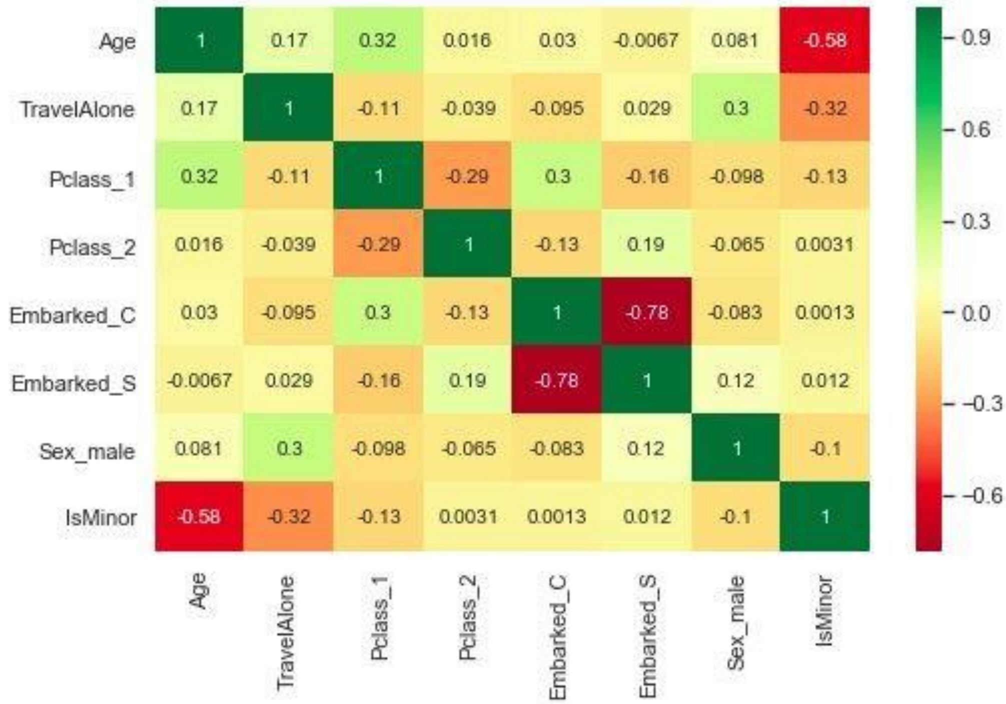
In [33]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
```

```
/usr/local/Cellar/python/3.7.2_2/Frameworks/Python.framework/Versions/3.7/lib/python3.7/importlib/_bootstrap.py:21
9: RuntimeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
/usr/local/Cellar/python/3.7.2_2/Frameworks/Python.framework/Versions/3.7/lib/python3.7/importlib/_bootstrap.py:21
9: RuntimeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
```

In [34]:
```
cols = ["Age","Fare","TravelAlone","Pclass_1","Pclass_2","Embarked_C","Embarked_S","Sex_male","IsMinor"]
X = final_train[cols]
y = final_train['Survived']
```

In [36]:
```
from sklearn.feature_selection import RFECV
# Create the RFE object and compute a cross-validated score.
# The "accuracy" scoring is proportional to the number of correct classifications
rfecv = RFECV(estimator=LogisticRegression(), step=1, cv=10, scoring='accuracy')
rfecv.fit(X, y)

print("Optimal number of features: %d" % rfecv.n_features_)
print('Selected features: %s' % list(X.columns[rfecv.support_]))

# Plot number of features VS. cross-validation scores
plt.figure(figsize=(10,6))
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```

In [37]:
```
Selected_features = ['Age', 'TravelAlone', 'Pclass_1', 'Pclass_2', 'Embarked_C',
                     'Embarked_S', 'Sex_male', 'IsMinor']
X = final_train[Selected_features]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



Review of model evaluation procedures

In [39]:
```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, precision_score, recall_score
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc, log_loss

# create X (features) and y (response)
X = final_train[Selected_features]
y = final_train['Survived']

# use train/test split with different random_state values
# we can change the random_state values that changes the accuracy scores
# the scores change a lot, this is why testing scores is a high-variance estimate

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [40]:
```
# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
```

In [41]:
```
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test, y_pred))
```

Train/Test split results:
LogisticRegression accuracy is 0.782

In [42]:
```
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred_proba))
```

```
LogisticRegression log_loss is 0.504
```

In [43]:
```
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

```
LogisticRegression auc is 0.838
```

In [64]:
```
final_test['Survived'] = log_clf.predict(final_test[Selected_features])
final_test['PassengerId'] = test_df['PassengerId']

submission = final_test[['PassengerId','Survived']]

submission.to_csv("submission.csv", index=False)

submission.tail()
```

Out[64]:

	PassengerId	Survived
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

Conclusion

In this project we formulated the task of Survival of Passengers in Titanic Dataset.

Findings from EDA - If you were on "the Titanic", your chances to survive would be the highest if you are a young female (or a child), have enough money to buy high fared tickets to get into a 1st class cabin, travelling in small family and getting aboard at the Port of Cherbourg.