Titanic_Survival

March 24, 2019

The goal of this exercise is to predict if the Titanic passengers in the test set survived. I have used Stacking to train my model. Stacking is an ensemble technique where predictions from multiple models are used to generate a second level model called meta-model. This second-layer algorithm is trained to optimally combine the model predictions to form a new set of predictions. There are many good resources online that explain this concept in detail.

In this kernel, I have tried multiple classification algorithms and used cross validation score to pick the 5 best models. These 5 models are called base models. The predictions from these base models serve as the input to the second-level model called the meta-model. I used Random Forest for fitting my meta-model. To predict the results, I used the base models to generate first-level predictions on test set. These first level predictions from the base models were used as the input to the meta model.

This kernel is a combination of multiple approaches that I have learned through various online courses and books.

First, we will import the libraries:

```
In [81]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
         from sklearn.ensemble import (RandomForestClassifier, AdaBoostClassifier,
                                       GradientBoostingClassifier, ExtraTreesClassifier)

In [82]: train = pd.read_csv('train.csv')
         test = pd.read_csv('test.csv')

In [83]: train.head()

Out[83]:    PassengerId  Survived  Pclass  \
         0            1         0       3
         1            2         1       1
         2            3         1       3
         3            4         1       1
         4            5         0       3

                                                      Name     Sex   Age  SibSp  \
         0                           Braund, Mr. Owen Harris    male  22.0      1
         1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
```

```
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

In [84]:
```
train.tail()
```

Out[84]:
```
     PassengerId  Survived  Pclass                                          Name  \
886          887         0       2                         Montvila, Rev. Juozas
887          888         1       1                  Graham, Miss. Margaret Edith
888          889         0       3      Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                         Behr, Mr. Karl Howell
890          891         0       3                           Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

In [85]:
```
train.describe()
```

Out[85]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
In [86]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

```
In [87]: len(train)
```

```
Out[87]: 891
```

```
In [88]: len(test)
```

```
Out[88]: 418
```

0.0.1 Missing Values

```
In [89]: print(train.isnull().sum())
```

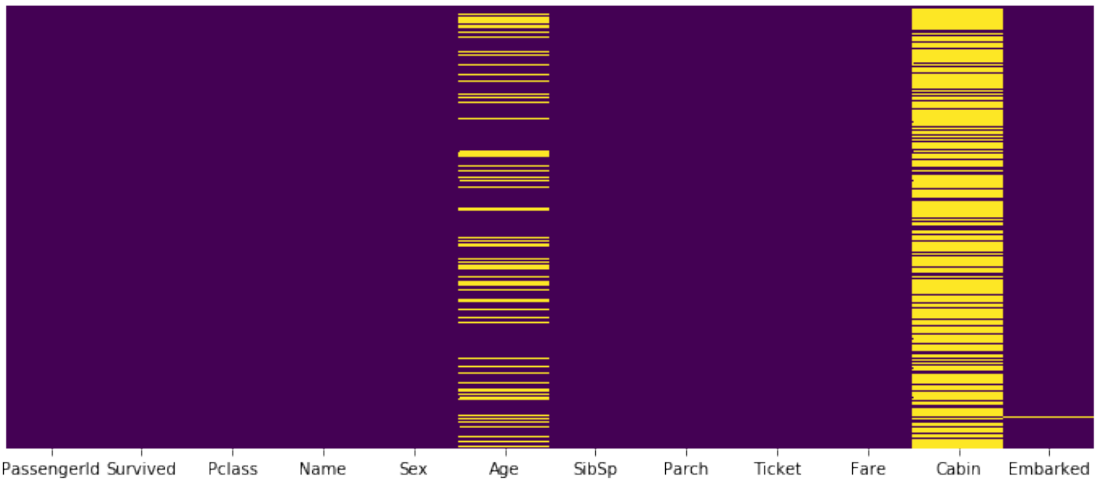
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
In [90]: print(test.isnull().sum())
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

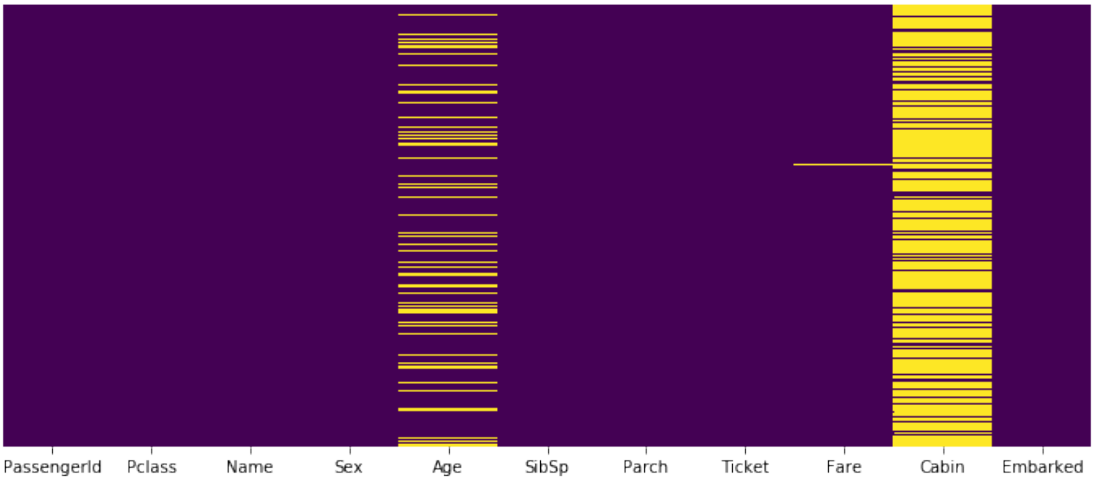
```
In [91]: plt.figure(figsize=(12,5))
         sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
Out[91]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e1f8940>
```




```
In [92]: plt.figure(figsize=(12,5))
         sns.heatmap(test.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

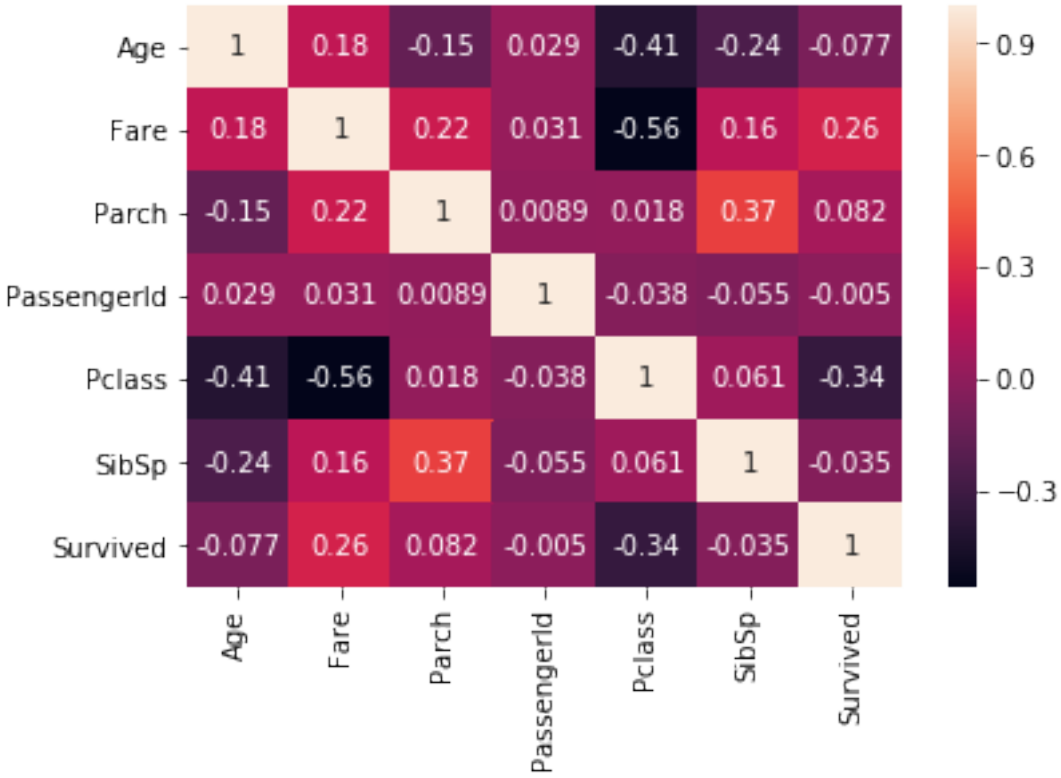
```
Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e249978>
```



0.0.2 Impute Missing Values

Age and Cabin columns contain many missing values. 77% of the observations for Cabin do not contain a value. ML algorithms will not be able to handle such a large number of missing values. We will drop the Cabin column.

```
In [93]: train.drop(['Cabin'], axis =1, inplace=True)
         test.drop(['Cabin'], axis = 1, inplace=True)

In [94]: combined = pd.concat([train,test])

In [95]: sns.heatmap(combined.corr(),annot=True)

Out[95]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e1f39e8>
```



There is a good correlation between Age and PClass. I will use PClass value to impute Age, where it is misisng.

```
In [96]: combined.groupby('Pclass').mean()['Age']
```

```
Out[96]: Pclass
         1    39.159930
         2    29.506705
         3    24.816367
         Name: Age, dtype: float64
```

```
In [97]: def setAge(cols):
             age = cols[0]
             pclass = cols[1]
             if pd.isnull(age):
                 if pclass == 1:
                     return 39
                 elif pclass == 2:
                     return 30
                 else:
                     return 25
             else:
                 return age
```

```
In [98]: train['Age'] = train[['Age','Pclass']].apply(setAge,axis=1)
         test['Age'] = test[['Age','Pclass']].apply(setAge,axis=1)

         combined['Embarked'].value_counts()

Out[98]: S    914
         C    270
         Q    123
         Name: Embarked, dtype: int64
```

As Southampton is the most common value for port emabrked, I will replace the misisng values with 'S'

```
In [99]: train['Embarked'] = train['Embarked'].replace(np.NaN, 'S')
         test['Embarked'] = test['Embarked'].replace(np.NaN, 'S')
```

There is one observation in the Train data set that is misisng Fare information. I will set the value based on the mean of 3rd class fare.

```
In [100]: combined.groupby('Pclass').mean()['Fare']

Out[100]: Pclass
          1    87.508992
          2    21.179196
          3    13.302889
          Name: Fare, dtype: float64

In [101]: test[test['Fare'].isnull()]

Out[101]:      PassengerId  Pclass                Name   Sex   Age  SibSp  Parch Ticket  \
          152         1044       3  Storey, Mr. Thomas  male  60.5      0      0   3701

               Fare Embarked
          152   NaN        S

In [102]: test["Fare"].fillna(13.30, inplace=True)
          print(train.isnull().sum())

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

```
In [103]: print(test.isnull().sum())

PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

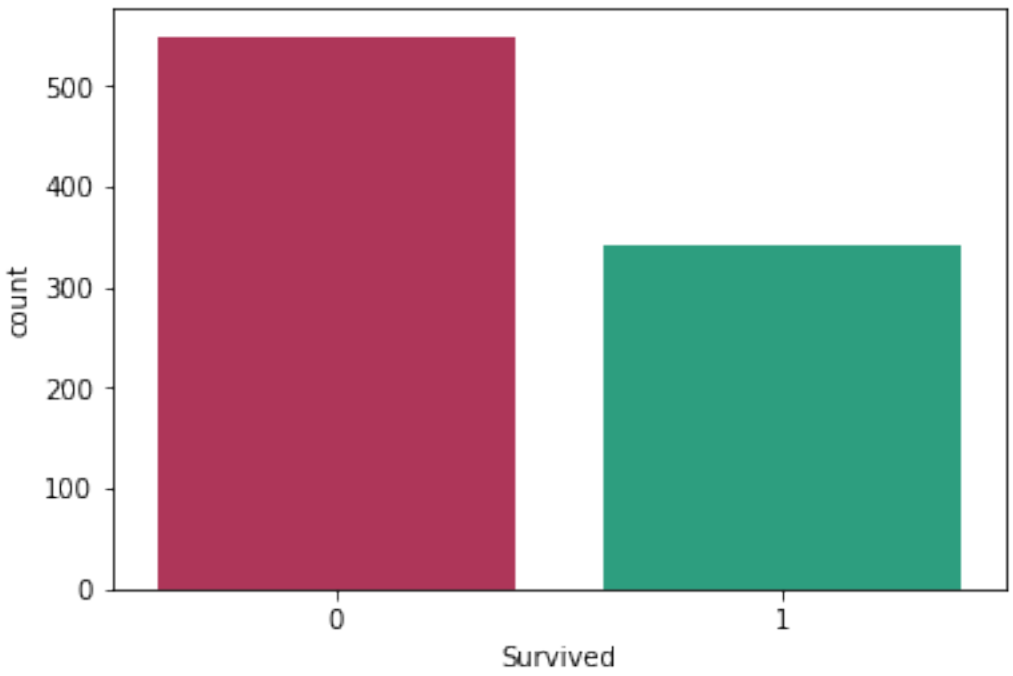
The above steps resolve all the missing values. We no longer have any missing values.

0.1 EDA & Feature Engineering

This is my favorite phase of any data science project. I feel it is important to analyze all variables in the data set. Most of my analysis is limited to the target variable 'Survived' but I have included some analysis on relationsips between other columns.

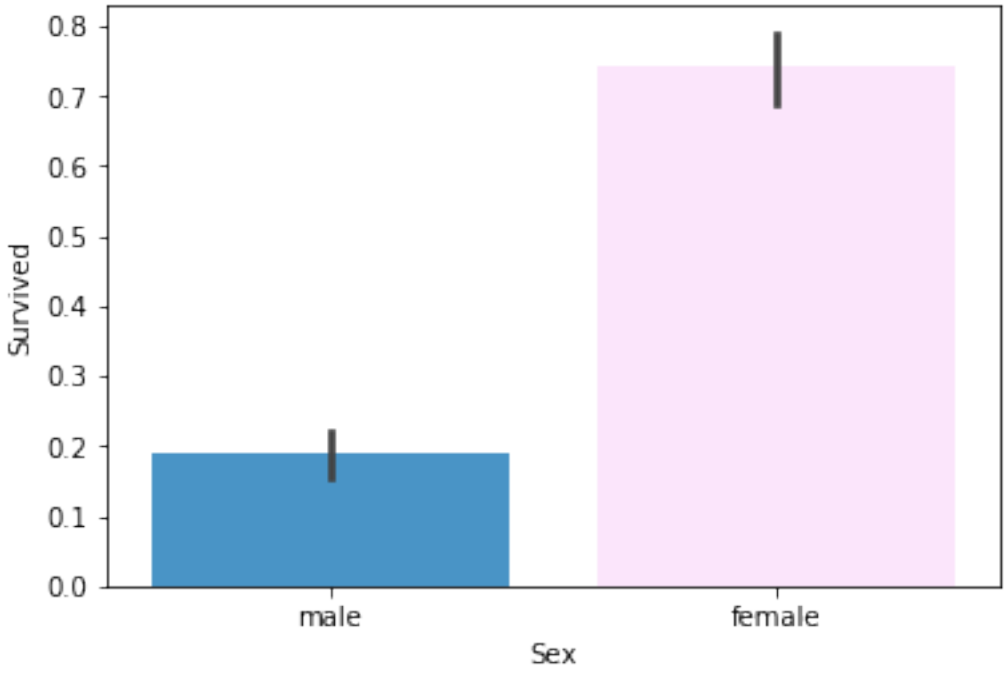
0.1.1 Sex

```
In [104]: sns.countplot(train['Survived'],palette= {1: "#1ab188", 0: "#c22250"})

Out[104]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e1f3780>
```



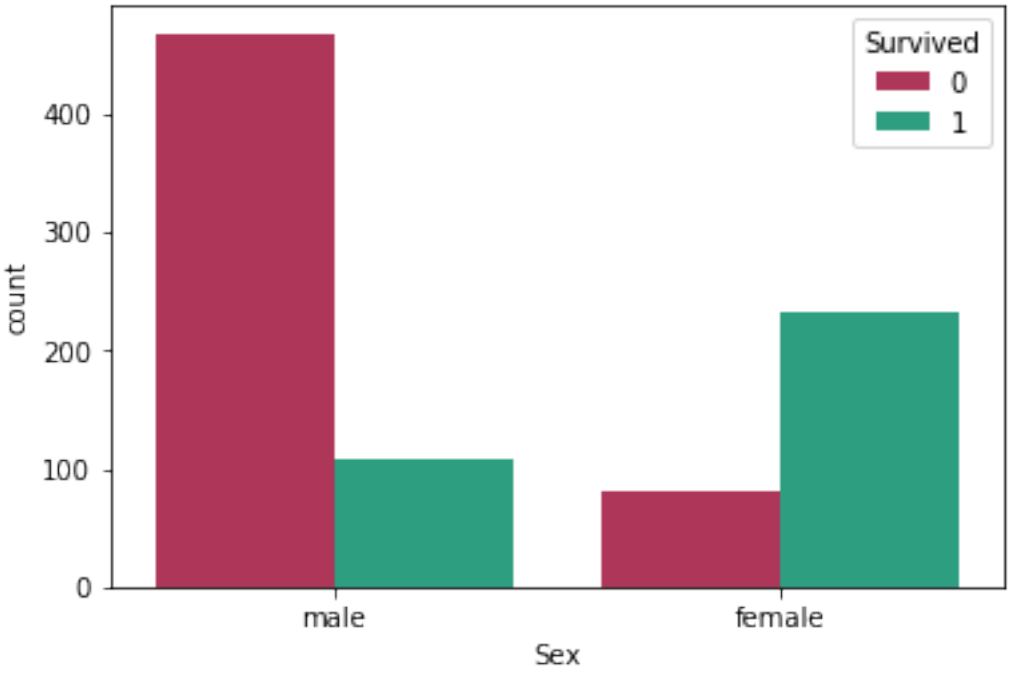
In [105]:
```
sns.barplot(data=train,x='Sex',y='Survived',palette= {'male': "#3498db", 'female': "
```

Out[105]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2683e400240>
```



In [106]:
```
#Sex
sns.countplot(data=train,x='Sex',hue='Survived',palette= {1: "#1ab188", 0: "#c22250"
```

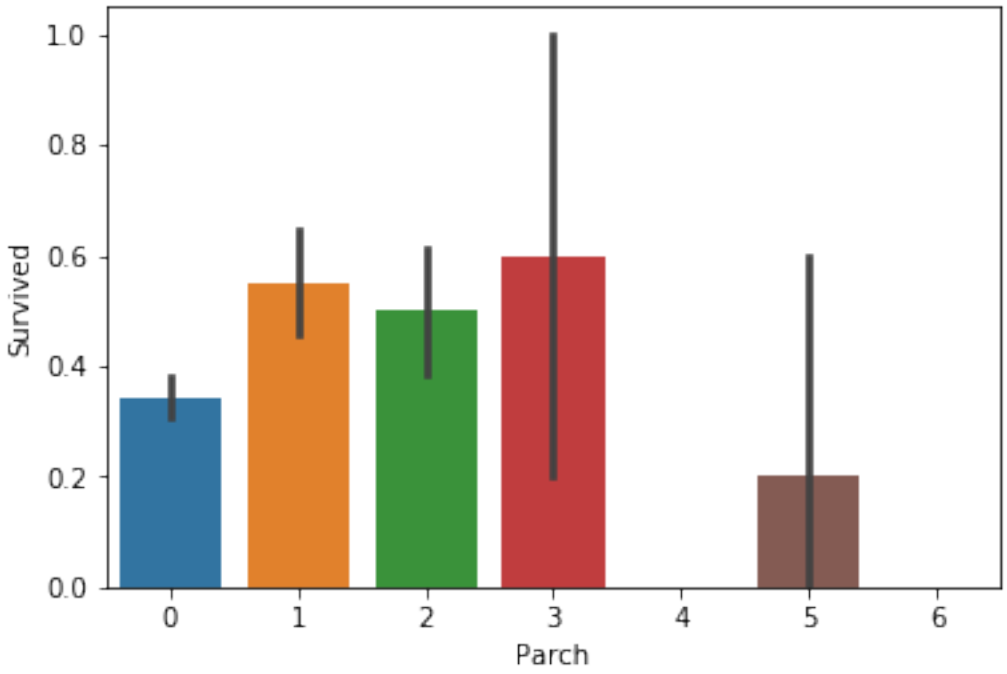
Out[106]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2683e44bb38>
```



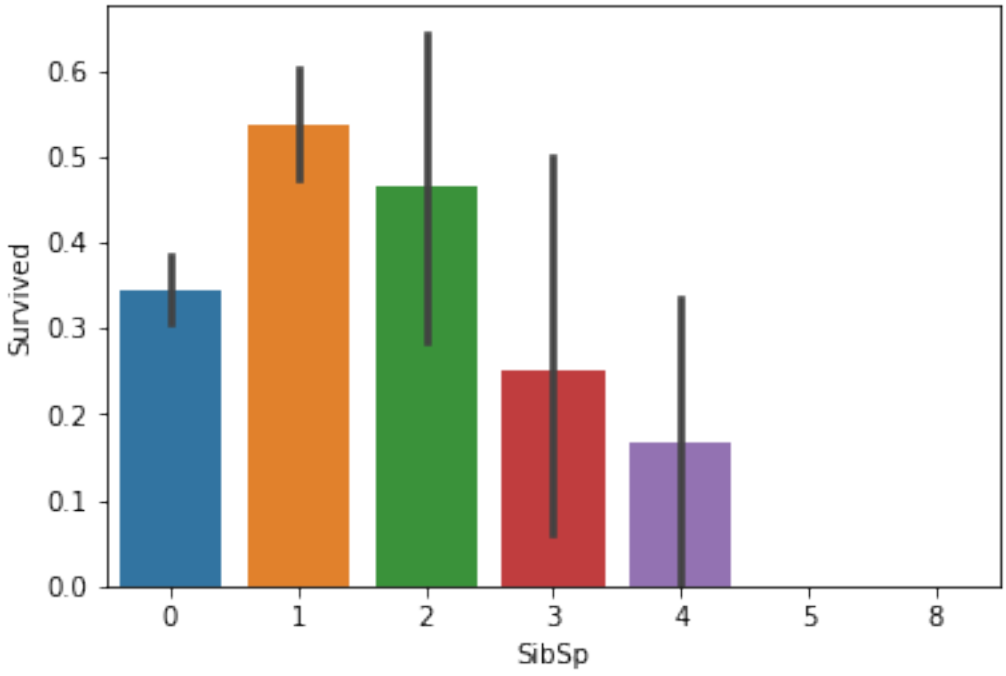
0.1.2 Parch, SibSp and Family Size

```
In [107]: sns.barplot(data=train,x='Parch',y='Survived')
```

```
Out[107]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e3a5ac8>
```

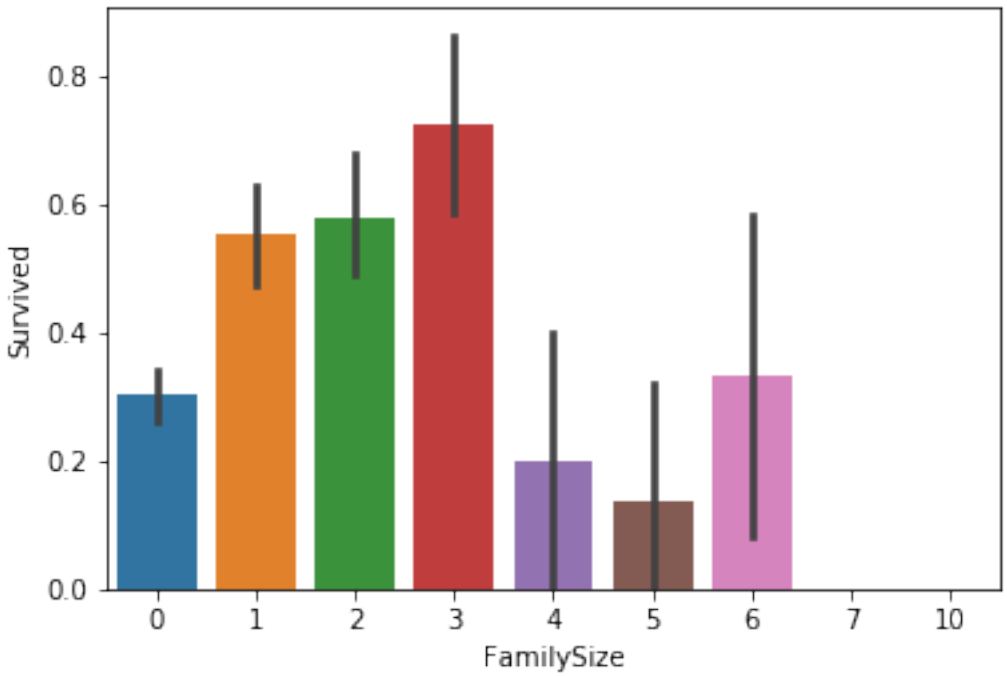


```
In [108]: sns.barplot(data=train,x='SibSp',y='Survived')

Out[108]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e49a710>
```



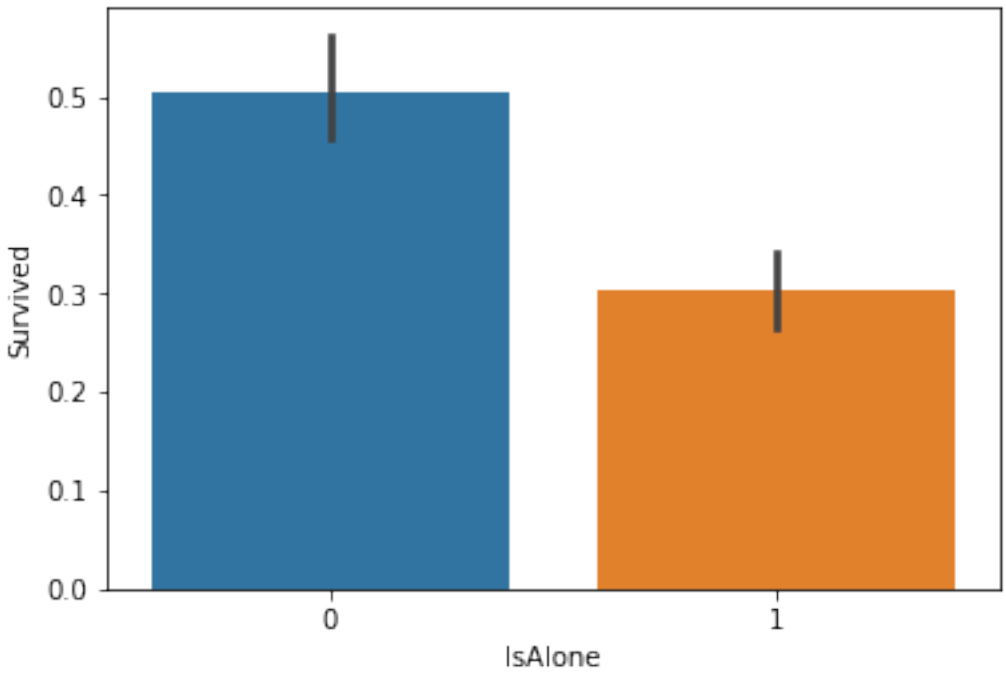
As the SibSp and Parch columns are not very different, I will combine them into a single column called Family Size

```
In [109]: train['FamilySize'] = train['SibSp'] + train ['Parch']
          test['FamilySize'] = test['SibSp'] + test ['Parch']

          sns.barplot(data=train,x='FamilySize',y='Survived')

Out[109]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e5915f8>
```



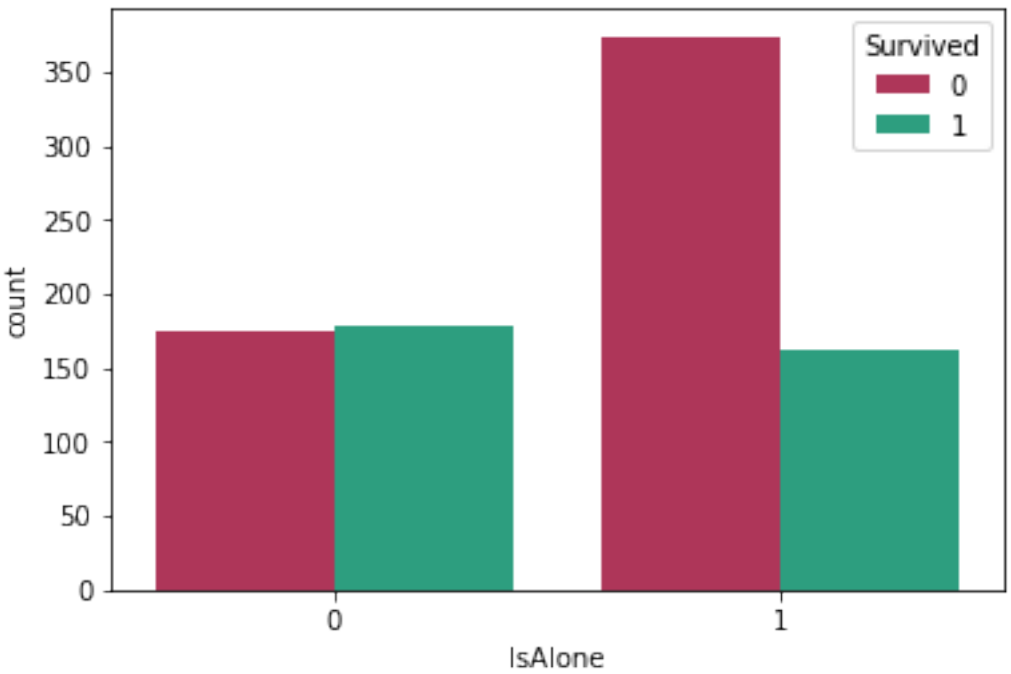
Survival probability is low for single travelers. So I will create a column to identify the solo travellers.

```
In [110]: def isAlone(cols):
              if (cols[0]==0) & (cols[1]==0):
                  return 1
              else:
                  return 0
          train['IsAlone'] = train[['SibSp','Parch']].apply(isAlone,axis=1)
          test['IsAlone'] = test[['SibSp','Parch']].apply(isAlone,axis=1)

          sns.barplot(data=train,x='IsAlone',y='Survived')

Out[110]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e617c50>
```



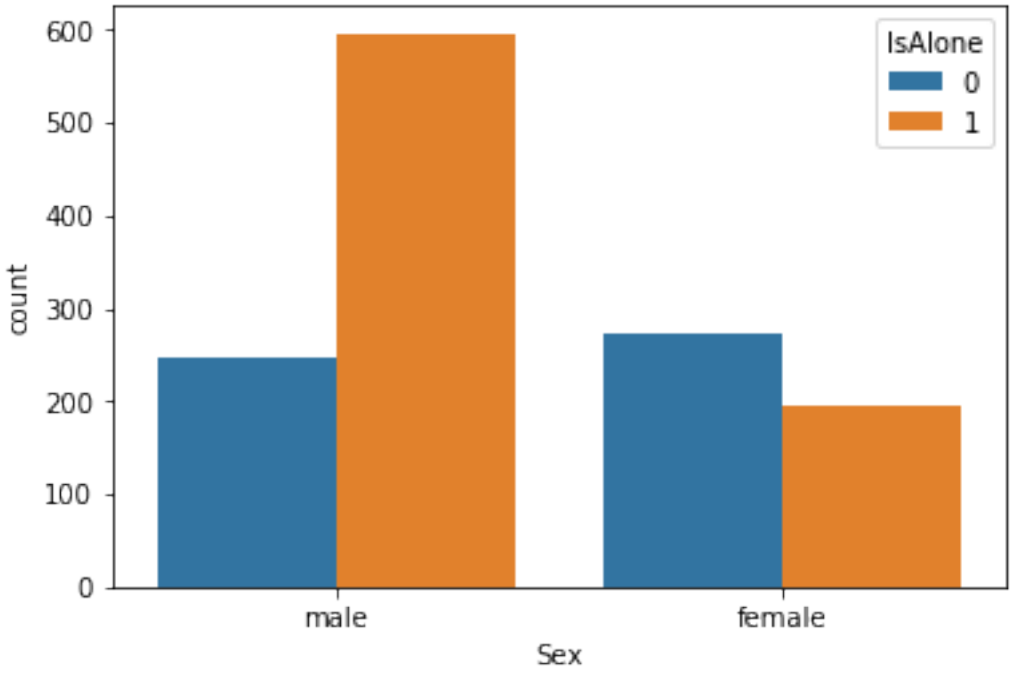

```
In [111]: sns.countplot(data=train,x='IsAlone',hue='Survived',palette= {1: "#1ab188", 0: "#c22
```

```
Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e6447b8>
```



In [112]:
```
#sns.countplot(data=train,x='Sex',hue='IsAlone')
combined = pd.concat([train,test])
sns.countplot(data=combined,x='Sex',hue='IsAlone')
```

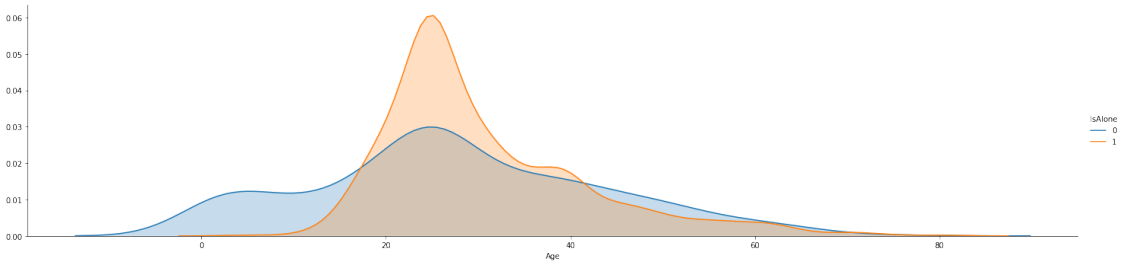
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x2683e6bcb70>



0.1.3 Age

In [113]:
```
f = sns.FacetGrid(combined,hue='IsAlone',size=5,aspect=4)
f.map(sns.kdeplot,'Age',shade= True)
f.add_legend()
```

Out[113]: <seaborn.axisgrid.FacetGrid at 0x2683e696208>



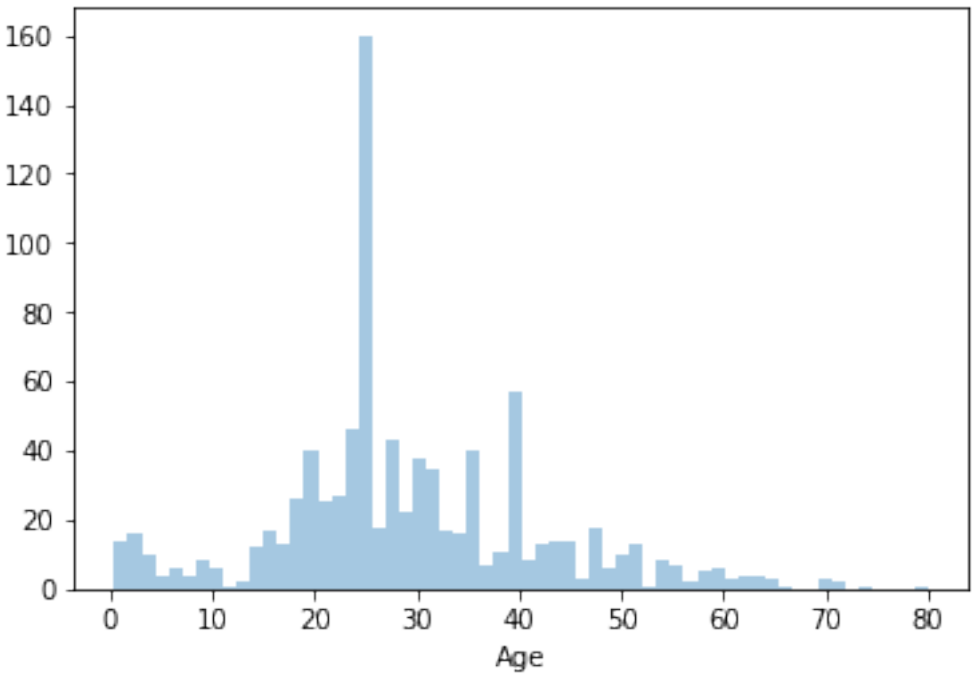
```
In [114]: combined[(combined['IsAlone'] == True)].sort_values(['Age']).head()
```

```
Out[114]:       Age Embarked  FamilySize     Fare  IsAlone  \
          777   5.0        S           0  12.4750        1
          731  11.0        C           0  18.7875        1
          120  12.0        S           0  15.7500        1
          780  13.0        C           0   7.2292        1
          14   14.0        S           0   7.8542        1

                                              Name  Parch  PassengerId  Pclass     Sex  \
          777         Emanuel, Miss. Virginia Ethel      0          778       3  female
          731              Hassan, Mr. Houssein G N      0          732       3    male
          120                  Watt, Miss. Bertha J      0         1012       2  female
          780                  Ayoub, Miss. Banoura      0          781       3  female
          14   Vestrom, Miss. Hulda Amanda Adolfina      0           15       3  female

               SibSp  Survived      Ticket
          777      0       1.0      364516
          731      0       0.0        2699
          120      0       NaN  C.A. 33595
          780      0       1.0        2687
          14       0       0.0      350406
```

The 2 plots above may not be of much significance to this challenge but I find them interesting. We observe that most female travelers had a family member travelling with them. Most of the travelers sailing alone were in the age group of 17-40. The youngest solo traveler was just 5 years and she survived. This piqued my curiosity and I found that she was travelling with a nurse-maid. More information about Virginia Ethel: https://www.encyclopedia-titanica.org/titanic-survivor/virginia-ethel-emanuel.html. The second youngest solo traveler was not that fortunate. He was travelling with a relative. That's it for now, lets continue with the challenge.

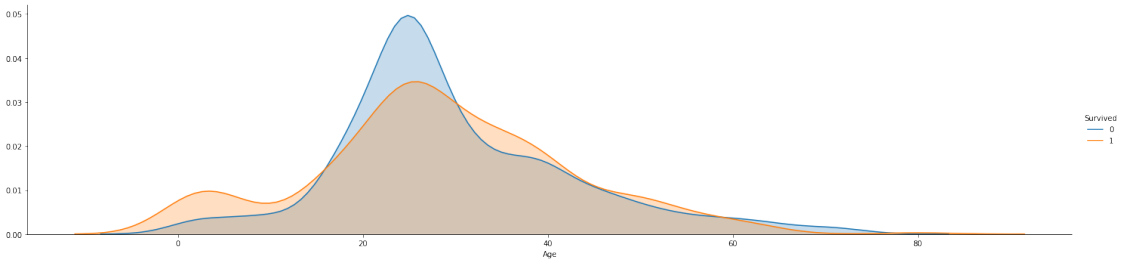
```
In [115]: sns.distplot(train['Age'].dropna(),kde=False,bins=60)
```

```
Out[115]: <matplotlib.axes._subplots.AxesSubplot at 0x2683f947240>
```




```
In [116]: f = sns.FacetGrid(train,hue='Survived',size=5,aspect=4)
          f.map(sns.kdeplot,'Age',shade= True)
          f.add_legend()
```

```
Out[116]: <seaborn.axisgrid.FacetGrid at 0x2683f9d66a0>
```

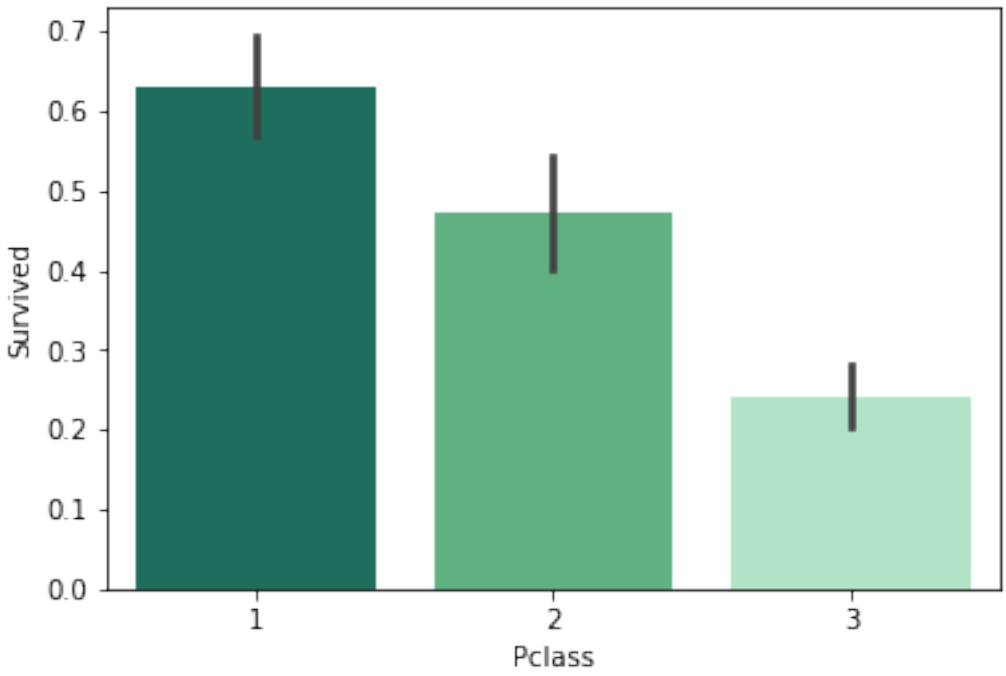


0.1.4 P Class and Fare

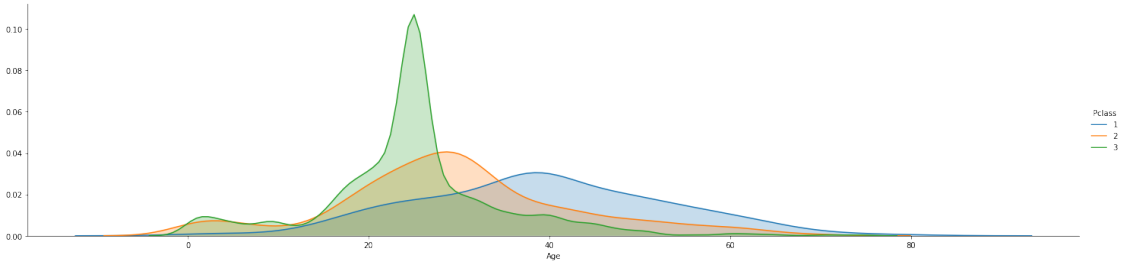
The chance of survival increased with class. Around 60% of first class passengers survived.Only 23% of third class passengers survived.

```
In [117]: sns.barplot(data=train,x='Pclass',y='Survived',palette= {1: "#117A65", 2: "#52BE80",3
```

```
Out[117]: <matplotlib.axes._subplots.AxesSubplot at 0x2683fa3b9e8>
```

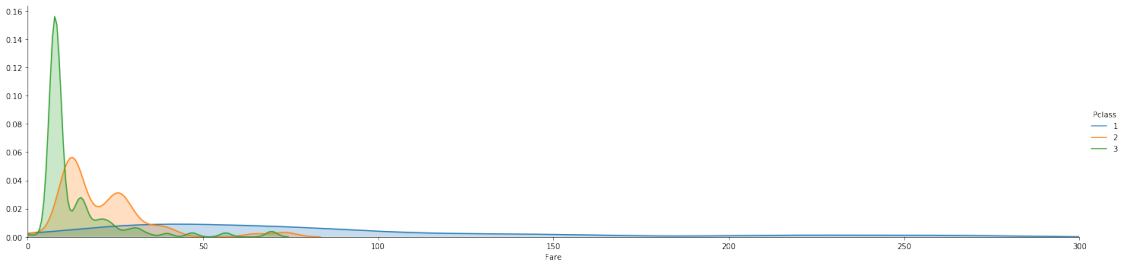



```
In [118]: f = sns.FacetGrid(combined,hue='Pclass',size=5,aspect=4)
          f.map(sns.kdeplot,'Age',shade= True)
          f.add_legend()

Out[118]: <seaborn.axisgrid.FacetGrid at 0x2683faabe48>
```



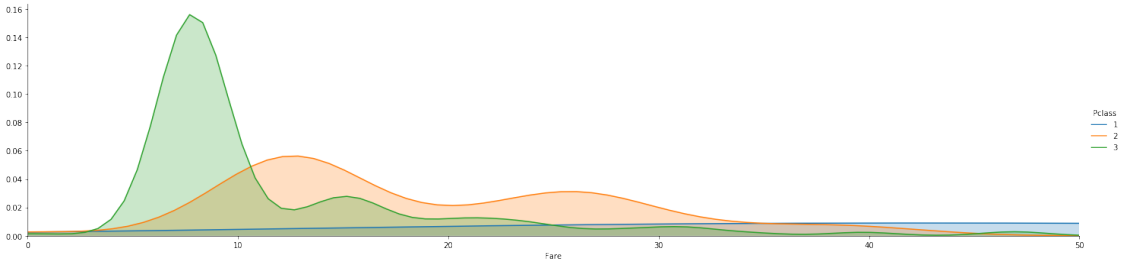
This above graph is interesting but expected.

```
In [119]: f = sns.FacetGrid(combined,hue='Pclass',size=5,aspect=4)
          plt.xlim(0, 300)
          f.map(sns.kdeplot,'Fare',shade= True)
          f.add_legend()

Out[119]: <seaborn.axisgrid.FacetGrid at 0x2683fb5db38>
```



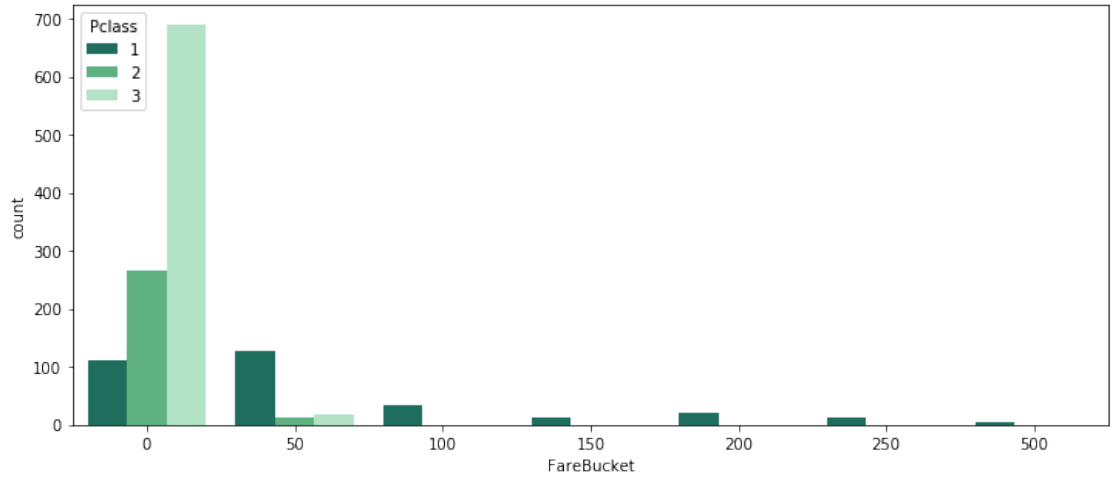
In [120]:
```
f = sns.FacetGrid(combined,hue='Pclass',size=5,aspect=4)
plt.xlim(0, 50)
f.map(sns.kdeplot,'Fare',shade= True)
f.add_legend()
```

Out[120]: <seaborn.axisgrid.FacetGrid at 0x26840620d68>



In [121]:
```
plt.figure(figsize=(12,5))
combined['FareBucket'] = (combined['Fare']/50).astype(int)*50
sns.countplot(data=combined,x='FareBucket',hue='Pclass',palette= {1: "#117A65", 2: "#
```

Out[121]: <matplotlib.axes._subplots.AxesSubplot at 0x26840b07da0>



```
In [122]: combined[combined['Pclass']==1].sort_values('Fare').head()

Out[122]:       Age Embarked  FamilySize  Fare  IsAlone  \
          822  38.0        S           0   0.0        1
          815  39.0        S           0   0.0        1
          263  40.0        S           0   0.0        1
          266  39.0        S           0   0.0        1
          633  39.0        S           0   0.0        1

                                              Name  Parch  PassengerId  Pclass   Sex  \
          822        Reuchlin, Jonkheer. John George      0          823       1  male
          815                     Fry, Mr. Richard      0          816       1  male
          263                 Harrison, Mr. William      0          264       1  male
          266  Chisholm, Mr. Roderick Robert Crispin      0         1158       1  male
          633          Parr, Mr. William Henry Marsh      0          634       1  male

               SibSp  Survived  Ticket  FareBucket
          822      0       0.0   19972           0
          815      0       0.0  112058           0
          263      0       0.0  112059           0
          266      0       NaN  112051           0
          633      0       0.0  112052           0
```

I see that a few first class passengers did not pay for their cross atlantic sail. This is interesting. I looked it up and found that many passengers were given complimentary tickets. So let's find the lowest revenue first class ticket.

```
In [123]: combined[(combined['Pclass']==1)&(combined['Fare']>0)].sort_values('Fare').head()

Out[123]:       Age Embarked  FamilySize     Fare  IsAlone  \
          872  33.0        S           0   5.0000        1
          662  47.0        S           0  25.5875        1
          77   55.0        S           2  25.7000        0
          205  39.0        C           0  25.7417        1
          168  39.0        S           0  25.9250        1

                                                  Name  Parch  PassengerId  \
          872                        Carlsson, Mr. Frans Olof      0          873
          662                       Colley, Mr. Edward Pomeroy      0          663
          77   Cornell, Mrs. Robert Clifford (Malvina Helen L...      0          969
          205                       Omont, Mr. Alfred Fernand      0         1097
          168                             Baumann, Mr. John D      0          169

               Pclass     Sex  SibSp  Survived      Ticket  FareBucket
          872       1    male      0       0.0         695           0
          662       1    male      0       0.0        5727           0
```

```
77        1  female      2       NaN       11770           0
205       1    male      0       NaN  F.C. 12998           0
168       1    male      0       0.0    PC 17318           0
```

Frans Olof paid just 5 pounds for his first class ticket. The encyclopedia-titanica article about him says that his company bought his ticket. I wish I knew his travel agent. Many first class passengers paid around 25 pounds which is considerably cheaper than the most expensive 3rd class ticket!

I am curious about the most expensive 3rd class ticket. Let's have a look.

```
In [124]: combined[combined['Pclass']==3].sort_values('Fare',ascending=False).head(5)
```

```
Out[124]:       Age Embarked  FamilySize   Fare  IsAlone                        Name  \
          159  25.0        S          10  69.55        0  Sage, Master. Thomas Henry
          201  25.0        S          10  69.55        0         Sage, Mr. Frederick
          846  25.0        S          10  69.55        0    Sage, Mr. Douglas Bullen
          792  25.0        S          10  69.55        0     Sage, Miss. Stella Anna
          188  25.0        S          10  69.55        0              Sage, Miss. Ada

               Parch  PassengerId  Pclass     Sex  SibSp  Survived    Ticket  FareBucket
          159      2          160       3    male      8       0.0  CA. 2343          50
          201      2          202       3    male      8       0.0  CA. 2343          50
          846      2          847       3    male      8       0.0  CA. 2343          50
          792      2          793       3  female      8       0.0  CA. 2343          50
          188      2         1080       3  female      8       NaN  CA. 2343          50
```

The highest 3rd class fares are associated with Ticket No CA. 2343. 11 passengers traveled on this ticket and 7 of them died. We do not know the fate of 4 who are the test set. This is unfortunate but reveals a very impotant charateristic about our data. I had ignored the 'Ticket' column till now but I should not ignore it. There is a good chance that they are from the same family. I hope someone from the Sage family survived.

```
In [125]: combined[combined['Ticket']=='CA. 2343']
```

```
Out[125]:       Age Embarked  FamilySize   Fare  IsAlone  \
          159  25.0        S          10  69.55        0
          180  25.0        S          10  69.55        0
          201  25.0        S          10  69.55        0
          324  25.0        S          10  69.55        0
          792  25.0        S          10  69.55        0
          846  25.0        S          10  69.55        0
          863  25.0        S          10  69.55        0
          188  25.0        S          10  69.55        0
          342  25.0        S          10  69.55        0
          360  14.5        S          10  69.55        0
          365  25.0        S          10  69.55        0

                                        Name  Parch  PassengerId  Pclass     Sex  \
          159         Sage, Master. Thomas Henry      2          160       3    male
```

```
180       Sage, Miss. Constance Gladys      2          181       3  female
201                Sage, Mr. Frederick      2          202       3    male
324            Sage, Mr. George John Jr     2          325       3    male
792             Sage, Miss. Stella Anna     2          793       3  female
846            Sage, Mr. Douglas Bullen     2          847       3    male
863  Sage, Miss. Dorothy Edith "Dolly"      2          864       3  female
188                    Sage, Miss. Ada      2         1080       3  female
342               Sage, Mr. John George     9         1234       3    male
360         Sage, Master. William Henry     2         1252       3    male
365      Sage, Mrs. John (Annie Bullen)     9         1257       3  female

     SibSp  Survived    Ticket  FareBucket
159      8       0.0  CA. 2343          50
180      8       0.0  CA. 2343          50
201      8       0.0  CA. 2343          50
324      8       0.0  CA. 2343          50
792      8       0.0  CA. 2343          50
846      8       0.0  CA. 2343          50
863      8       0.0  CA. 2343          50
188      8       NaN  CA. 2343          50
342      1       NaN  CA. 2343          50
360      8       NaN  CA. 2343          50
365      1       NaN  CA. 2343          50
```

```
In [126]: tst_sageFamily = test[test['Ticket']=='CA. 2343']
          tst_sageFamily
```

```
Out[126]:      PassengerId  Pclass                                Name     Sex   Age  SibSp  \
          188         1080       3                 Sage, Miss. Ada  female  25.0      8
          342         1234       3           Sage, Mr. John George    male  25.0      1
          360         1252       3     Sage, Master. William Henry    male  14.5      8
          365         1257       3  Sage, Mrs. John (Annie Bullen)  female  25.0      1

               Parch    Ticket   Fare Embarked  FamilySize  IsAlone
          188      2  CA. 2343  69.55        S          10        0
          342      9  CA. 2343  69.55        S          10        0
          360      2  CA. 2343  69.55        S          10        0
          365      9  CA. 2343  69.55        S          10        0
```

I did some researching online for more information about their expensive 3rd class fare but I was not able to find anything concrete. There is a mention of the family changing plans (to sail aboard the Titanic instead of Philadelphia) due to the coal strike. Maybe they ended up buying these expensive tickets due to the late change of plans.

0.2 Ticket Number

There are 40 groups with more than 2 passengers in the trainset. I will create a column called TicSurvProb that will contain the probability of survival of passengers who have these ticket numbers.

```
In [127]: ticCount = train.groupby('Ticket')['Sex'].count()
          ticSurN = train.groupby('Ticket')['Survived'].sum()
          ticCount = pd.DataFrame(ticCount)
          ticSurN = pd.DataFrame(ticSurN)
          ticSur = ticCount.join(ticSurN)
          ticSur['TicSurvProb'] = ticSur['Survived']*(100) /(ticSur['Sex'])

          ticSur.rename(index=str, columns={"Sex": "PassengerCount", "Survived": "PassengersSu
          ticSur.reset_index(level=0, inplace=True)
          len(ticSur)

Out[127]: 681

In [128]: ticSur.head()

Out[128]:    Ticket  PassengerCount  PassengersSurvived  TicSurvProb
          0  110152               3                   3   100.000000
          1  110413               3                   2    66.666667
          2  110465               2                   0     0.000000
          3  110564               1                   1   100.000000
          4  110813               1                   1   100.000000

In [129]: ticSur = ticSur[ticSur['PassengerCount'] > 2]
          len(ticSur)

Out[129]: 40

In [130]: train = pd.merge(train, ticSur, on=['Ticket', 'Ticket'],how='left')
          train['TicSurvProb'] = train['TicSurvProb'].replace(np.NaN, 40)

In [131]: len(train)

Out[131]: 891

In [132]: test = pd.merge(test, ticSur, on=['Ticket', 'Ticket'],how='left')
          print(test.isnull().sum())

PassengerId             0
Pclass                  0
Name                    0
Sex                     0
Age                     0
SibSp                   0
Parch                   0
Ticket                  0
Fare                    0
Embarked                0
FamilySize              0
IsAlone                 0
```

```
PassengerCount        391
PassengersSurvived    391
TicSurvProb           391
dtype: int64
```

```
In [133]: test['TicSurvProb'] = test['TicSurvProb'].replace(np.NaN, 40)
          len(test)
```

```
Out[133]: 418
```

```
In [134]: train.drop(['PassengerCount','PassengersSurvived','Ticket'],axis=1,inplace=True)
          test.drop(['PassengerCount','PassengersSurvived','Ticket'],axis=1,inplace=True)
```

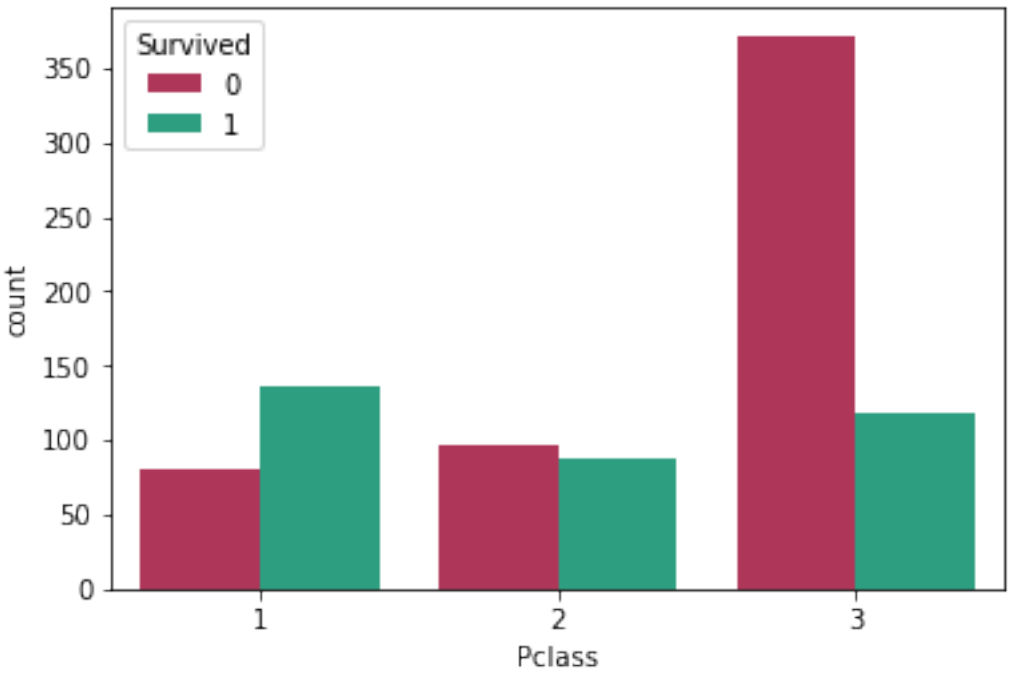
```
In [135]: train.head()
```

```
Out[135]:    PassengerId  Survived  Pclass  \
          0            1         0       3
          1            2         1       1
          2            3         1       3
          3            4         1       1
          4            5         0       3

                                                Name     Sex   Age  SibSp  \
          0                            Braund, Mr. Owen Harris    male  22.0      1
          1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
          2                             Heikkinen, Miss. Laina  female  26.0      0
          3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
          4                           Allen, Mr. William Henry    male  35.0      0

             Parch     Fare Embarked  FamilySize  IsAlone  TicSurvProb
          0      0   7.2500        S           1        0         40.0
          1      0  71.2833        C           1        0         40.0
          2      0   7.9250        S           0        1         40.0
          3      0  53.1000        S           1        0         40.0
          4      0   8.0500        S           0        1         40.0
```

0.3 End Ticket Number

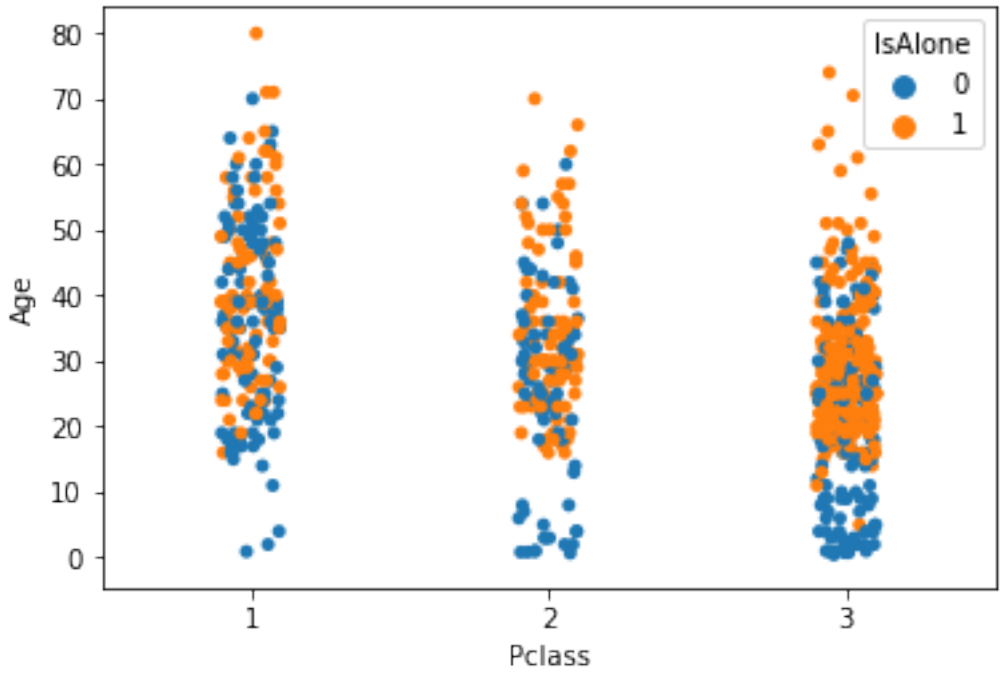
```
In [136]: sns.countplot(data=train,x='Pclass',hue='Survived',palette= {1: "#1ab188", 0: "#c222
```

```
Out[136]: <matplotlib.axes._subplots.AxesSubplot at 0x26840cf11d0>
```




```
In [137]: sns.stripplot(x='Pclass',y='Age',data=train,hue='IsAlone',jitter=True)
```

```
Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x26840bd6a20>
```



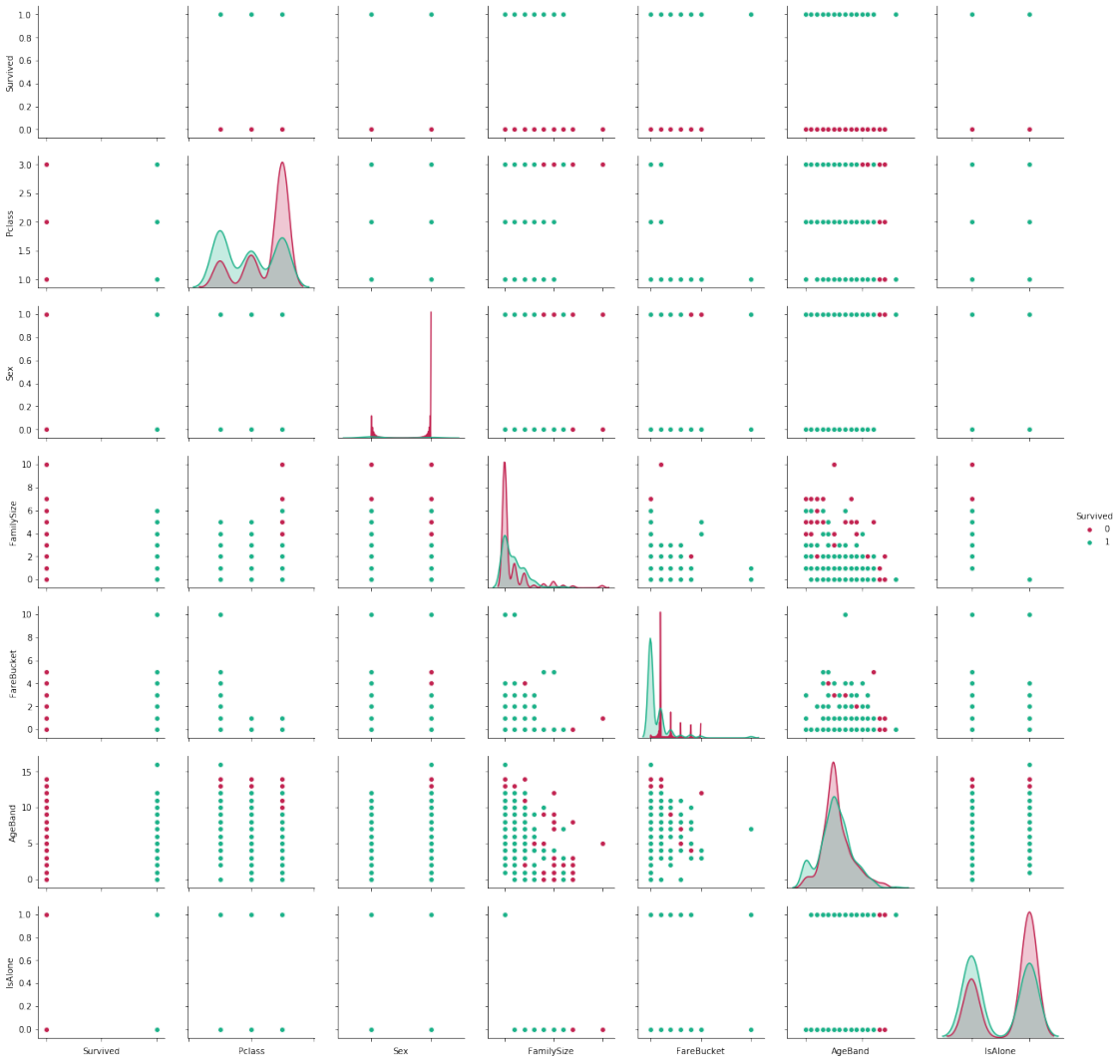
0.4 Categorical Columns

```
In [138]: train = pd.get_dummies(train, columns=['Embarked'])
          test = pd.get_dummies(test, columns=['Embarked'])

In [139]: train['Sex'] = train['Sex'].map( {'female': 0, 'male': 1} ).astype(int)
          test['Sex'] = test['Sex'].map( {'female': 0, 'male': 1} ).astype(int)
```

0.5 Binning

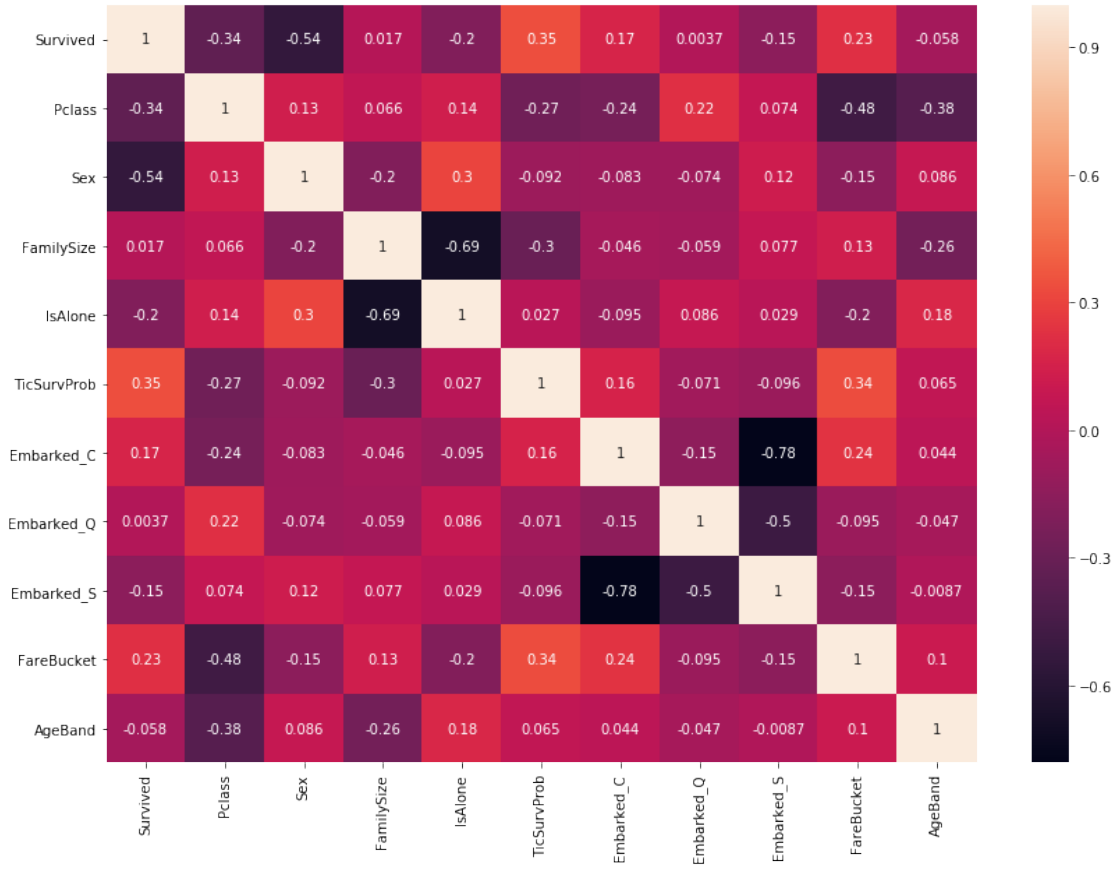
Here, I split the Fare and Age columns to bins based on value. I can also create quantile bins using the pd.qcut function but I found better results based on value.

```
In [140]: train['FareBucket'] = (train['Fare']/50).astype(int)
          test['FareBucket'] = (test['Fare']/50).astype(int)

In [141]: train['AgeBand'] = (train['Age']/5).astype(int)
          test['AgeBand'] = (test['Age']/5).astype(int)

In [142]: train.drop(['Age','Fare','PassengerId','Name','SibSp','Parch'], axis =1, inplace=Tru
          test.drop(['Age','Fare','Name','SibSp','Parch'], axis = 1, inplace=True)

In [143]: p = sns.pairplot(train[['Survived', 'Pclass', 'Sex', 'FamilySize', 'FareBucket','Age
          p.set(xticklabels=[])

Out[143]: <seaborn.axisgrid.PairGrid at 0x26840c3aa90>
```




```
In [144]: plt.figure(figsize=(14,10))
          sns.heatmap(train.corr(),annot=True)
```

```
Out[144]: <matplotlib.axes._subplots.AxesSubplot at 0x2684390cf60>
```



The column 'Sex' has the highest correlation with Survived followed by 'TicSurvProb'.

```
In [145]: pd.DataFrame(train.corr()['Survived']).abs().sort_values('Survived',ascending=False)
```

```
Out[145]:              Survived
          Survived     1.000000
          Sex          0.543351
          TicSurvProb  0.345087
          Pclass       0.338481
          FareBucket   0.225942
          IsAlone      0.203367
          Embarked_C   0.168240
          Embarked_S   0.149683
          AgeBand      0.057515
          FamilySize   0.016639
          Embarked_Q   0.003650
```

Insert x and y

```
In [146]: from sklearn.model_selection import train_test_split
          X = train.drop(['Survived'],axis=1)
          y = train['Survived']
```

0.6 Stacking

I will use the entire train data to perform Cross Validation. I may get better results with KNN and SVC if I scale the data but I have skipped that step.

```
In [147]: from sklearn.model_selection import KFold
          from sklearn.ensemble import ExtraTreesClassifier
          from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.svm import SVC
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.naive_bayes import MultinomialNB
          from sklearn.model_selection import cross_val_score
          import xgboost as xgb
          import warnings
          warnings.filterwarnings('ignore')
          num_of_estimators = 500
          rfClass = RandomForestClassifier(n_estimators=200,max_depth=3,
                                           min_samples_leaf= 1,max_features=5,min_samples_split
          logClass = LogisticRegression(penalty='l1',C=21.544346900318832)
          svcClass = SVC(gamma=0.001,C=10)
          knnClass = KNeighborsClassifier(n_neighbors=9)
          xgbClass = xgb.XGBClassifier(n_estimators=100,colsample_bytree= 0.8, gamma=1, max_dep
          nbClass = MultinomialNB()
          adaClass = AdaBoostClassifier(n_estimators=20,learning_rate=0.2)
          extraTreesClass = ExtraTreesClassifier(n_estimators=50,bootstrap=False,criterion='ent
                                                 min_samples_split=10,max_depth=None)
          gradientBClass = GradientBoostingClassifier(n_estimators=20,max_depth=3,max_features=
```

0.7 Stacking

```
In [148]: #Build base models

In [149]: len(X) #Length before splitting:891 Source - Coursera Advanced ML

Out[149]: 891
```

I will divide the training data set into 3 subsets - s_train, s_valid and s_test. s_train will be used to train the base models. I will use these base models to make predictions on the s_valid dataset. I will then make a data frame of all the predictions and this will server as the training data to the meta model. s_test will be used to test the model.

Divide into train and a temporary test set

```
In [150]: X_s_train, X_s_test2, y_s_train, y_s_test2 = train_test_split(X, y, test_size=0.5)
```

Divide the temporary test set into validate and test

```
In [151]: X_s_valid, X_s_test, y_s_valid, y_s_test = train_test_split(X_s_test2, y_s_test2, tes
```

```
In [152]: print(len(X_s_train)) #length of Train: 534  (60%)
          print(len(X_s_valid)) #length of Validate: 178 (20%)
          print(len(X_s_test)) #length of Test: 179 (20%)
```

```
445
401
45
```

Train the base models with s_train

```
In [153]: rfClass.fit(X_s_train,y_s_train)
          adaClass.fit(X_s_train,y_s_train)
          extraTreesClass.fit(X_s_train,y_s_train)
          logClass.fit(X_s_train,y_s_train)
          xgbClass.fit(X_s_train,y_s_train)
          svcClass.fit(X_s_train,y_s_train)
          knnClass.fit(X_s_train,y_s_train)
          gradientBClass.fit(X_s_train,y_s_train)

          #predict these models on validate data
          vld_rfPred = rfClass.predict(X_s_valid)
          vld_adaPred = adaClass.predict(X_s_valid)
          vld_extPred = extraTreesClass.predict(X_s_valid)
          vld_logPred = logClass.predict(X_s_valid)
          vld_xgbPred = xgbClass.predict(X_s_valid)
          vld_svcPred = svcClass.predict(X_s_valid)
          vld_knnPred = knnClass.predict(X_s_valid)
          vld_gbPred =  gradientBClass.predict(X_s_valid)
```

```
In [154]: base_predictions_train = pd.DataFrame( {
              'RandomForest': vld_rfPred,
              'AdaptiveBoost': vld_adaPred,
              'ExtraTrees': vld_extPred,
              'Log': vld_logPred,
              'XGB': vld_xgbPred,
              'SVC': vld_svcPred,
              'KNN': vld_knnPred,
              'GB' : vld_gbPred,
              'Y': y_s_valid,
              })
          base_predictions_train.head()
```

Out[154]:

	RandomForest	AdaptiveBoost	ExtraTrees	Log	XGB	SVC	KNN	GB	Y
151	1	1	1	1	1	1	1	1	1
887	1	1	1	1	1	1	0	1	1
383	1	1	1	1	1	1	1	1	1
66	1	1	1	1	1	1	0	1	1
52	1	1	1	1	1	1	1	1	1

In [155]:
```
#Analyze Stack Result Begin
```

In [156]:
```
def generateMatchScore(row):
    score =0
    if(row['AdaptiveBoost'] == row['Y']):
        score = score + 1
    if(row['ExtraTrees'] == row['Y']):
        score = score + 1
    if(row['KNN'] == row['Y']):
        score = score + 1
    if(row['Log'] == row['Y']):
        score = score + 1
    if(row['RandomForest'] == row['Y']):
        score = score + 1
    if(row['SVC'] == row['Y']):
        score = score + 1
    if(row['XGB'] == row['Y']):
        score = score + 1
    return score
```

In [157]:
```
base_predictions_train['Score'] = base_predictions_train.apply(generateMatchScore, ax
```

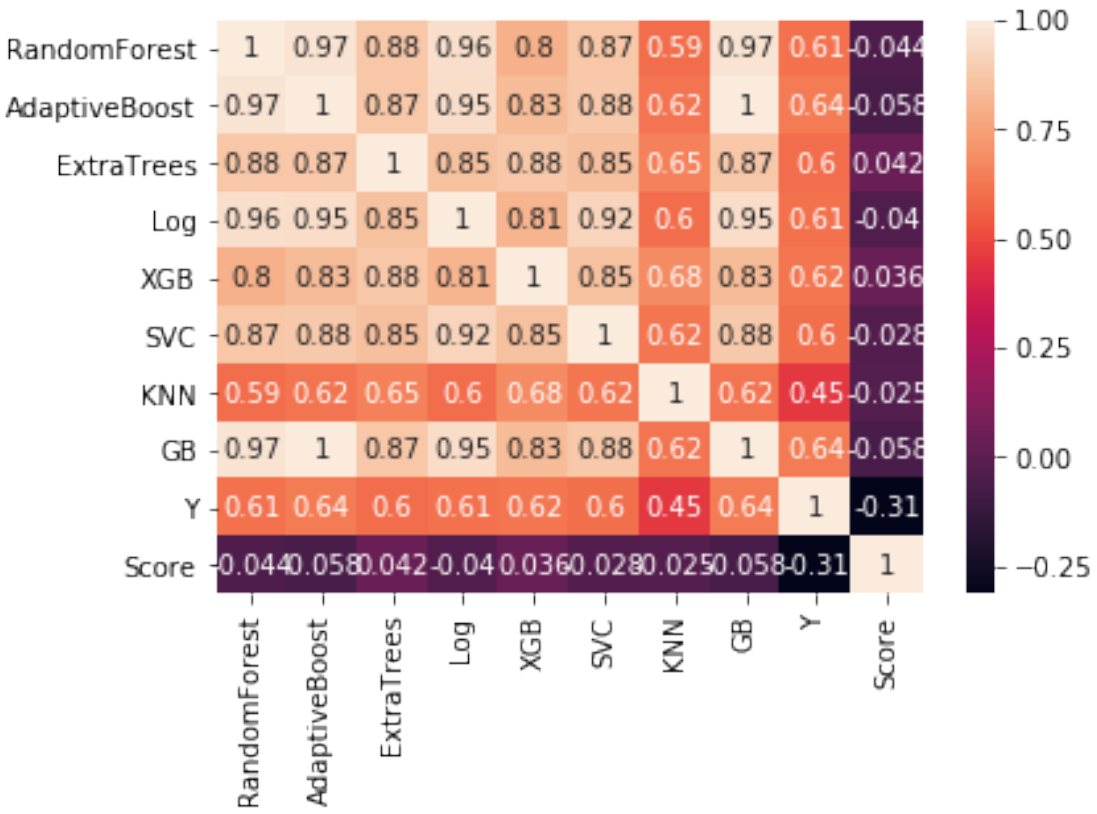
In [158]:
```
base_predictions_train[base_predictions_train['Score'] ==1]
```

Out[158]:

	RandomForest	AdaptiveBoost	ExtraTrees	Log	XGB	SVC	KNN	GB	Y	Score
507	0	0	0	0	0	0	1	0	1	1
512	0	0	0	0	0	0	1	0	1	1
599	0	0	0	0	0	0	1	0	1	1
423	1	1	1	1	1	1	0	1	0	1
114	1	1	1	1	0	1	1	1	0	1
224	0	0	0	0	0	0	1	0	1	1
772	1	1	1	1	1	0	1	1	0	1
248	0	0	0	0	0	0	1	0	1	1
182	1	1	0	1	1	1	1	1	0	1
854	1	1	1	1	1	1	0	1	0	1
38	1	1	1	1	1	1	0	1	0	1
617	1	1	1	1	0	1	1	1	0	1
690	0	0	0	0	0	0	1	0	1	1
447	0	0	0	0	0	0	1	0	1	1
18	1	1	1	1	0	1	1	1	0	1

In [159]:
```
sns.heatmap(base_predictions_train.corr(),annot=True)
```

Out[159]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x268444a0c50>
```




```
In [160]: #Concatenate all predictions on Validate
          stacked_valid_predictions = np.column_stack((vld_rfPred, vld_adaPred, vld_extPred,vl
                                                       vld_svcPred,vld_knnPred,vld_gbPred))
```

Create meta model

```
In [161]: meta_model = xgb.XGBClassifier(n_estimators=90,colsample_bytree=0.8, gamma=5, max_de
                                          min_child_weight=10, subsample=0.6)
```

Fit meta model on Validate subset

```
In [162]: meta_model.fit(stacked_valid_predictions,y_s_valid)
```

```
Out[162]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                 colsample_bytree=0.8, gamma=5, learning_rate=0.1, max_delta_step=0,
                 max_depth=3, min_child_weight=10, missing=None, n_estimators=90,
                 n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
                 reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                 silent=True, subsample=0.6)
```

```
In [163]: feature_importances = pd.DataFrame(meta_model.feature_importances_,index = ['vld_rfP
                                                       'vld_svcPred','vld_knnPred','vld_gbPred
          feature_importances
```

```
Out[163]:              importance
          vld_xgbPred    0.384615
          vld_adaPred    0.307692
          vld_svcPred    0.115385
          vld_gbPred     0.115385
          vld_extPred    0.076923
          vld_rfPred     0.000000
          vld_logPred    0.000000
          vld_knnPred    0.000000
```

Use Base Models to predict on s_test set

```
In [164]: tst_rfPred = rfClass.predict(X_s_test)
          tst_adaPred = adaClass.predict(X_s_test)
          tst_extPred = extraTreesClass.predict(X_s_test)
          tst_logPred = logClass.predict(X_s_test)
          tst_xgbPred = xgbClass.predict(X_s_test)
          tst_svcPred = svcClass.predict(X_s_test)
          tst_knnPred = knnClass.predict(X_s_test)
          tst_gbPred = gradientBClass.predict(X_s_test)

          #Concatenate base model predictions on Test
          stacked_test_predictions = np.column_stack((tst_rfPred, tst_adaPred, tst_extPred,tst_
                                                      tst_svcPred,tst_knnPred,tst_gbPred))
```

Use the predictions from the above step as input to the meta model

```
In [165]: #Predict Test predictions using meta model
          s_test_pred = meta_model.predict(stacked_test_predictions)

In [166]: from sklearn.metrics import confusion_matrix,classification_report
          print(classification_report(y_s_test,s_test_pred))

                    precision    recall  f1-score   support

                 0       0.87      0.93      0.90        28
                 1       0.87      0.76      0.81        17

avg / total       0.87      0.87      0.86        45
```

Predictions on Test Data for submission

```
In [167]: X_test = test.drop(['PassengerId'],axis=1)

In [168]: #Use the base models to make predictions on test set
          t_rfPred = rfClass.predict(X_test)
          t_adaPred = adaClass.predict(X_test)
```

```
t_extPred = extraTreesClass.predict(X_test)
t_logPred = logClass.predict(X_test)
t_xgbPred = xgbClass.predict(X_test)
t_svcPred = svcClass.predict(X_test)
t_knnPred = knnClass.predict(X_test)
t_gbPred = gradientBClass.predict(X_test)

#Concatenate base model predictions on Test
stacked_t_predictions = np.column_stack((t_rfPred, t_adaPred, t_extPred,t_logPred,t_x
#Use the meta model to make predictions on test set
final_pred = meta_model.predict(stacked_t_predictions)
```

Before we submit, lets check if the 4 remaining passengers of the Sage family survived

```
In [169]: tst_sageFamily
```

```
Out[169]:      PassengerId  Pclass                                Name     Sex   Age  SibSp  \
          188         1080       3                 Sage, Miss. Ada  female  25.0      8
          342         1234       3           Sage, Mr. John George    male  25.0      1
          360         1252       3     Sage, Master. William Henry    male  14.5      8
          365         1257       3  Sage, Mrs. John (Annie Bullen)  female  25.0      1

               Parch    Ticket   Fare Embarked  FamilySize  IsAlone
          188      2  CA. 2343  69.55        S          10        0
          342      9  CA. 2343  69.55        S          10        0
          360      2  CA. 2343  69.55        S          10        0
          365      9  CA. 2343  69.55        S          10        0
```

```
In [170]: submission = pd.DataFrame({
                "PassengerId": test["PassengerId"],
                "Survived": final_pred
            })
```

```
In [171]: submission.head()
```

```
Out[171]:    PassengerId  Survived
          0          892         0
          1          893         1
          2          894         0
          3          895         0
          4          896         1
```

```
In [172]: tst_sageFamily.head()
```

```
Out[172]:      PassengerId  Pclass                                Name     Sex   Age  SibSp  \
          188         1080       3                 Sage, Miss. Ada  female  25.0      8
          342         1234       3           Sage, Mr. John George    male  25.0      1
          360         1252       3     Sage, Master. William Henry    male  14.5      8
          365         1257       3  Sage, Mrs. John (Annie Bullen)  female  25.0      1
```

```
     Parch    Ticket   Fare Embarked  FamilySize  IsAlone
188      2  CA. 2343  69.55        S          10        0
342      9  CA. 2343  69.55        S          10        0
360      2  CA. 2343  69.55        S          10        0
365      9  CA. 2343  69.55        S          10        0
```

In [173]:
```
#submission['PassengerId'] in tst_sageFamily['PassengerId']
tst_sageFamily.merge(submission,how='left',on='PassengerId')
```

Out[173]:
```
   PassengerId  Pclass                                Name     Sex   Age  SibSp  \
0         1080       3                 Sage, Miss. Ada  female  25.0      8
1         1234       3           Sage, Mr. John George    male  25.0      1
2         1252       3     Sage, Master. William Henry    male  14.5      8
3         1257       3  Sage, Mrs. John (Annie Bullen)  female  25.0      1

   Parch    Ticket   Fare Embarked  FamilySize  IsAlone  Survived
0      2  CA. 2343  69.55        S          10        0         0
1      9  CA. 2343  69.55        S          10        0         0
2      2  CA. 2343  69.55        S          10        0         0
3      9  CA. 2343  69.55        S          10        0         0
```

In [174]:
```
submission.to_csv('titanic_output.csv', index=False)
```