In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt;
import seaborn as sns
%matplotlib inline
```

In [2]:
```python
train = pd.read_csv('../input/train.csv')
```

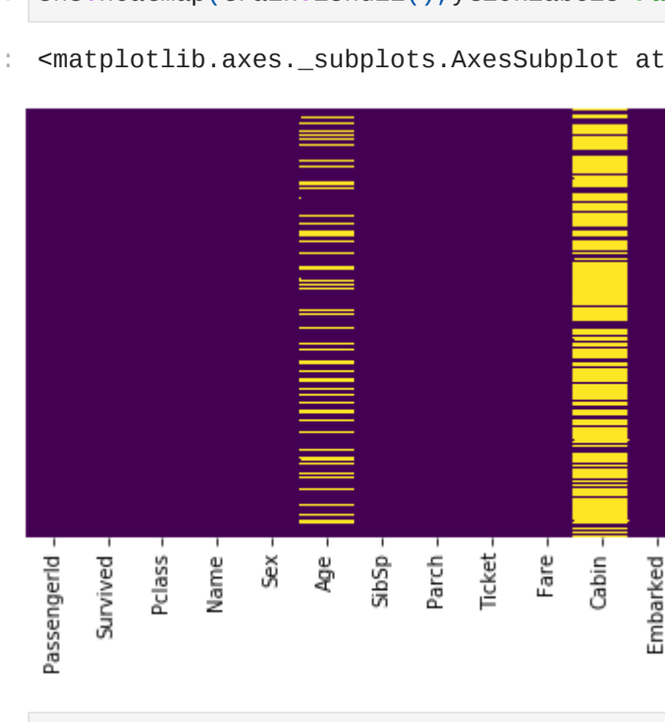
In [3]:
```python
train.tail()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [4]:
```python
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd795f337b8>



In [5]:
```python
train.isnull().sum().sort_values(ascending=False)
```

Out[5]:
```
Cabin          687
Age            177
Embarked         2
Fare             0
Ticket           0
Parch            0
SibSp            0
Sex              0
Name             0
Pclass           0
Survived         0
PassengerId      0
dtype: int64
```

Roughly 20 percent of the Age data is missing. The proportion of Age missing is likely small enough for reasonable replacement with some form of imputation. Looking at the Cabin column, it looks like we are just missing too much of that data to do something useful with at a basic level. We'll probably drop this later, or change it to another feature like "Cabin Known: 1 or 0"

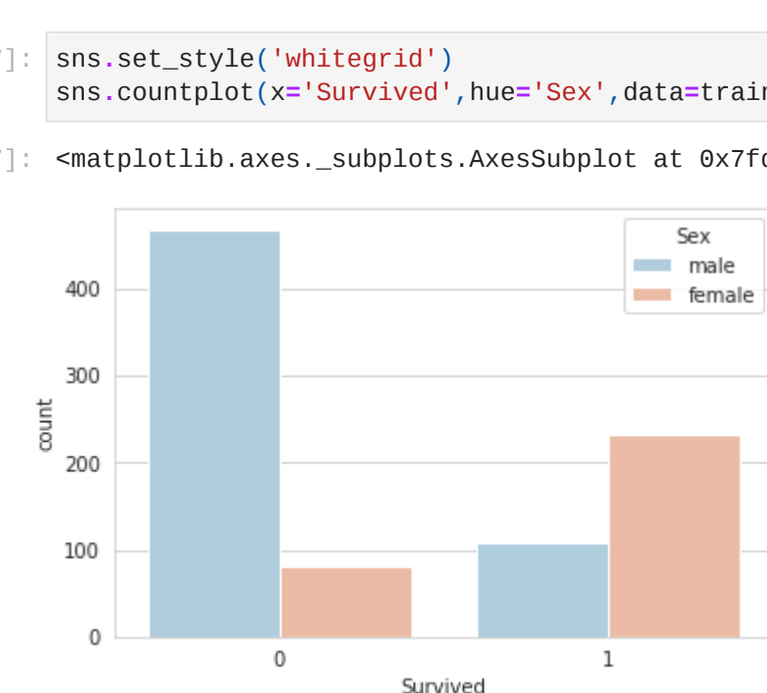
Let's continue on by visualizing some more of the data! Check out the video for full explanations over these plots, this code is just to serve as reference.

In [6]:
```python
sns.set_style('whitegrid')
sns.countplot(x='Survived',data=train,palette='RdBu_r')
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793dd8860>

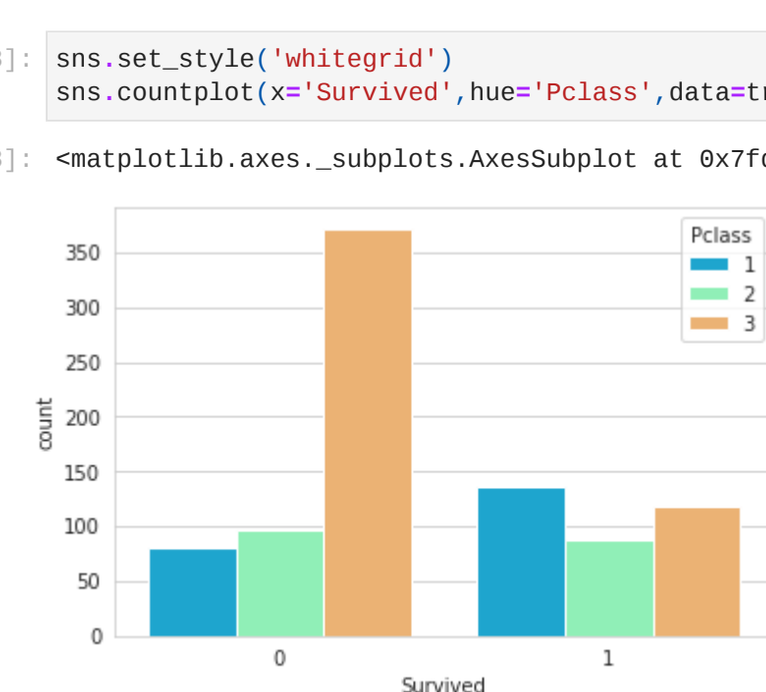
In [7]:
```python
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Sex',data=train,palette='RdBu_r')
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793dc4b38>



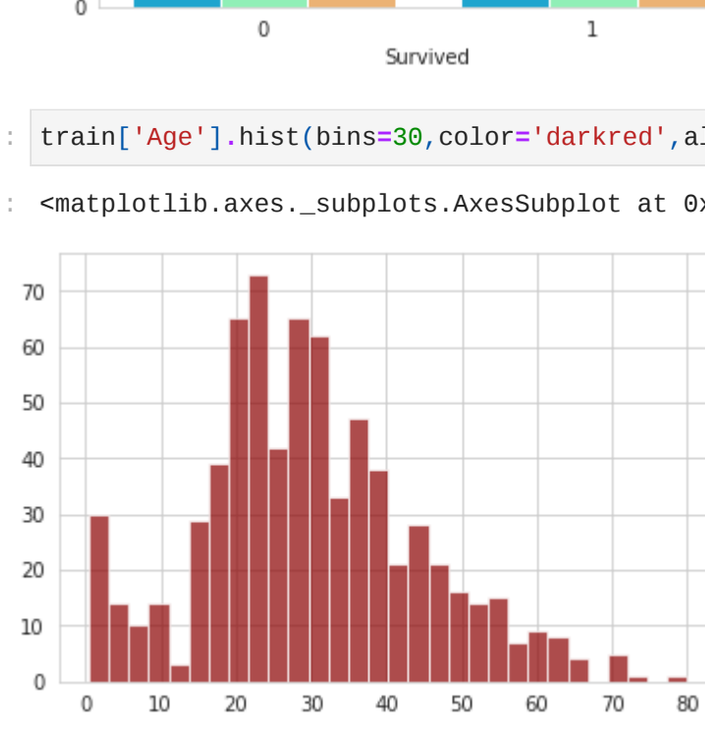
In [8]:
```python
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Pclass',data=train,palette='rainbow')
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793d234a8>



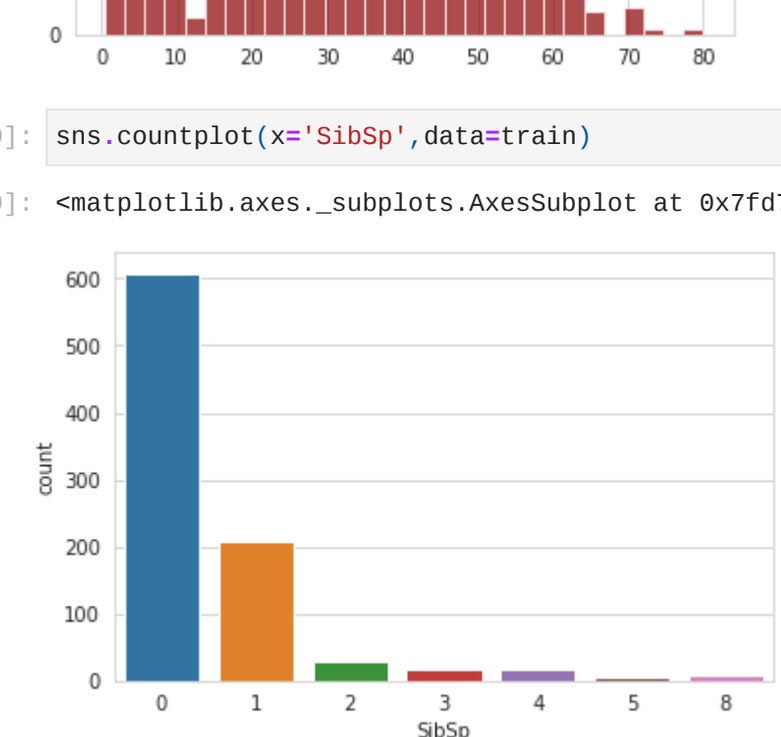
In [9]:
```python
train['Age'].hist(bins=30,color='darkred',alpha=0.7)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793c8b128>



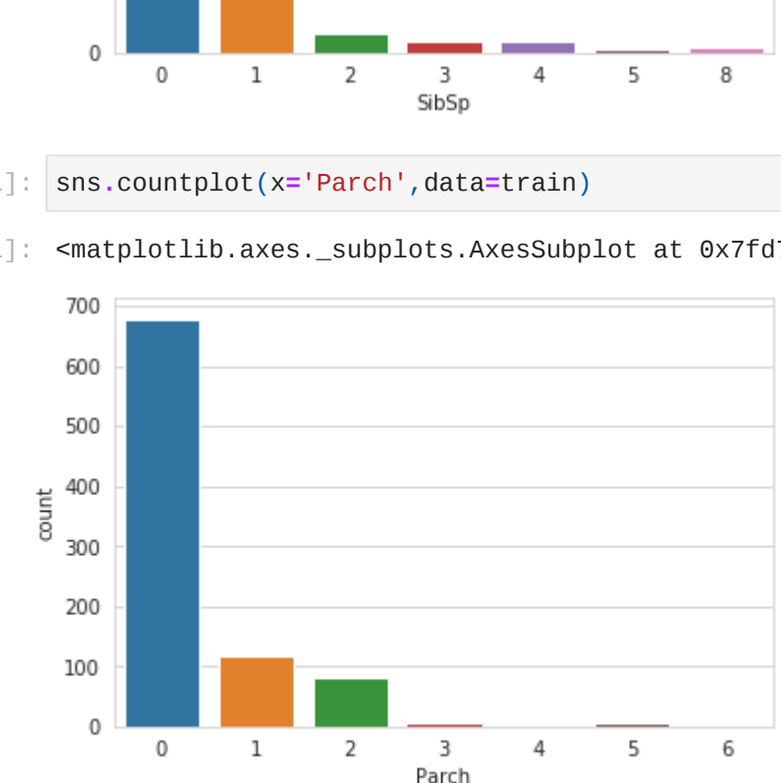
In [10]:
```python
sns.countplot(x='SibSp',data=train)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793cee3c8>



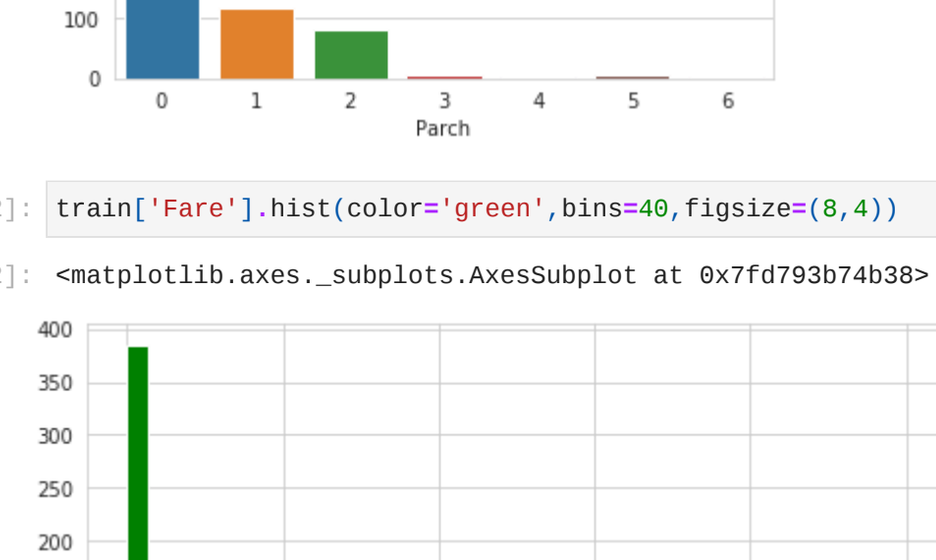
In [11]:
```python
sns.countplot(x='Parch',data=train)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793b95668>



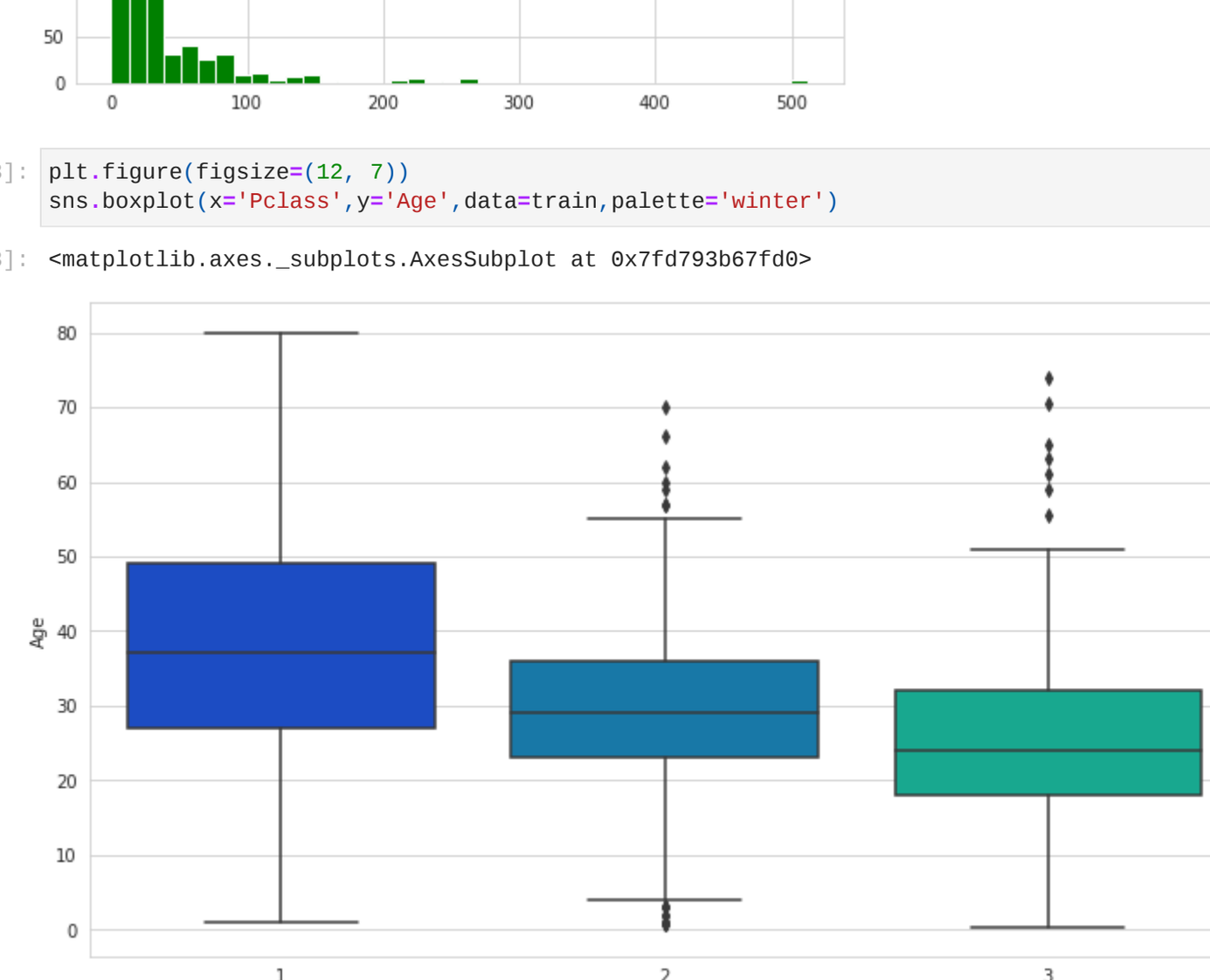
In [12]:
```python
train['Fare'].hist(color='green',bins=40,figsize=(8,4))
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793b74b38>



In [13]:
```python
plt.figure(figsize=(12, 7))
sns.boxplot(x='Pclass',y='Age',data=train,palette='winter')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793b67fd0>



We can see the wealthier passengers in the higher classes tend to be older, which makes sense. We'll use these average age values to impute based on Pclass for Age.

In [14]:
```python
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2:
            return 29

        else:
            return 24

    else:
        return Age
```

Now apply that function!

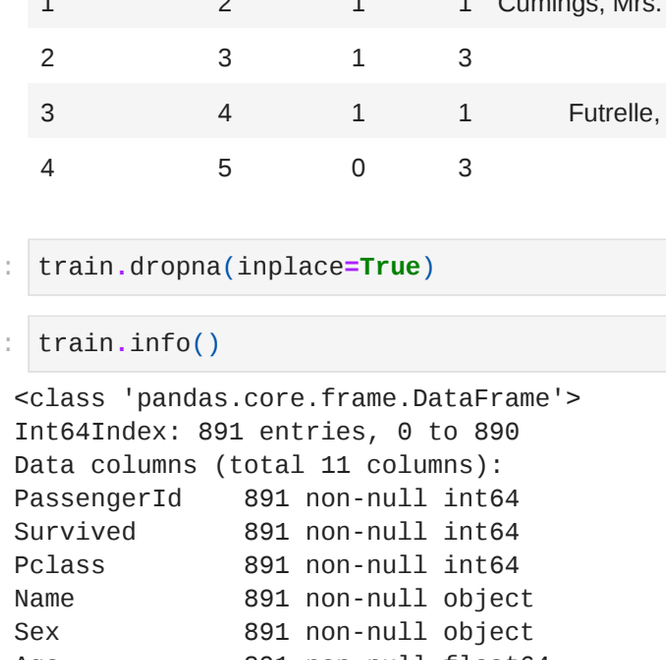
In [15]:
```python
train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
```

In [16]:
```python
train['Embarked'] = train['Embarked'].fillna('S')
```

Now let's check that heat map again!

In [17]:
```python
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd793a13f28>



Great! Let's go ahead and drop the Cabin column and the row in Embarked that is NaN.

We will sum of family member

In [18]:
```python
train.drop('Cabin',axis=1,inplace=True)
```

In [19]:
```python
train.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]:
```python
train.dropna(inplace=True)
```

In [21]:
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 0 to 890
Data columns (total 11 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            891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Embarked       891 non-null object
dtypes: float64(2), int64(5), object(4)
memory usage: 83.5+ KB
```

In [22]:
```python
sex = pd.get_dummies(train['Sex'],drop_first=True)
embark = pd.get_dummies(train['Embarked'],drop_first=True)
```

In [23]:
```python
train.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace=True)
```

In [24]:
```python
train = pd.concat([train,sex,embark],axis=1)
```

In [25]:
```python
train.head()
```

Out[25]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	0	3	22.0	1	0	7.2500	1	0	1
1	2	1	1	38.0	1	0	71.2833	0	0	0
2	3	1	3	26.0	0	0	7.9250	0	0	1
3	4	1	1	35.0	1	0	53.1000	0	0	1
4	5	0	3	35.0	0	0	8.0500	1	0	1

In [26]:
```python
from sklearn.model_selection import train_test_split
```

In [27]:
```python
X_train, X_test, y_train, y_test = train_test_split(train.drop(['Survived'],axis=1),
                                                    train['Survived'], test_size=0.10,
                                                    random_state=101)
```

In [28]:
```python
from sklearn.linear_model import LogisticRegression
```

In [29]:
```python
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[29]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [30]:
```python
predictions = logmodel.predict(X_test)
X_test.head()
```

Out[30]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	male	Q	S
331	332	1	45.5	0	0	28.500	1	0	1
700	701	1	18.0	1	0	227.525	0	0	0
748	749	1	19.0	1	0	53.100	1	0	1
751	752	3	6.0	0	1	12.475	1	0	1
481	482	2	29.0	0	0	0.000	1	0	1

In [31]:
```python
predictions
```

Out[31]:
```
array([0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1])
```

Evaluation

We can check precision,recall,f1-score using classification report!

In [32]:
```python
from sklearn.metrics import classification_report,confusion_matrix
```

In [33]:
```python
print(confusion_matrix(y_test,predictions))
```

```
[[46  5]
 [13 26]]
```

In [34]:
```python
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.78      0.90      0.84        51
           1       0.84      0.67      0.74        39

    accuracy                           0.80        90
   macro avg       0.81      0.78      0.79        90
weighted avg       0.81      0.80      0.80        90
```

In [35]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [36]:
```python
dt_model=DecisionTreeClassifier()
dt_model.fit(X_train,y_train)
```

Out[36]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [37]:
```python
dt_pred = dt_model.predict(X_test)
```

In [38]:
```python
print(confusion_matrix(y_test,dt_pred))
```

```
[[42  9]
 [14 25]]
```

In [39]:
```python
print(classification_report(y_test,dt_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.82      0.79        51
           1       0.74      0.64      0.68        39

    accuracy                           0.74        90
   macro avg       0.74      0.73      0.73        90
weighted avg       0.74      0.74      0.74        90
```

In [40]:
```python
from sklearn.ensemble import RandomForestClassifier
```

In [41]:
```python
rf= RandomForestClassifier(n_estimators=500)
rf.fit(X_train,y_train)
```

Out[41]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=500,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [42]:
```python
rf_pre=rf.predict(X_test)
```

In [43]:
```python
print(confusion_matrix(y_test,rf_pre))
```

```
[[48  3]
 [15 24]]
```

In [44]:
```python
print(classification_report(y_test,rf_pre))
```

```
              precision    recall  f1-score   support

           0       0.76      0.94      0.84        51
           1       0.89      0.62      0.73        39

    accuracy                           0.80        90
   macro avg       0.83      0.78      0.78        90
weighted avg       0.82      0.80      0.79        90
```

In [45]:
```python
from xgboost import XGBClassifier
xgboost = XGBClassifier(n_estimators=1000)
xgboost.fit(X_train,y_train)
```

Out[45]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [46]:
```python
xg_pred = xgboost.predict(X_test)
```

In [47]:
```python
print(confusion_matrix(y_test,xg_pred))
```

```
[[44  7]
 [16 23]]
```

In [48]:
```python
print(classification_report(y_test,xg_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.86      0.79        51
           1       0.77      0.59      0.67        39

    accuracy                           0.74        90
   macro avg       0.75      0.73      0.73        90
weighted avg       0.75      0.74      0.74        90
```

In [49]:
```python
import keras
from keras.layers import Dense
from keras.models import Sequential
```

```
Using TensorFlow backend.
```

In [50]:
```python
ann  = Sequential()
ann.add(Dense(units= 32,init= 'uniform', activation = 'relu', input_dim=9))
ann.add(Dense(units= 32,init= 'uniform', activation = 'relu'))
ann.add(Dense(units= 1,init= 'uniform', activation = 'sigmoid'))
ann.compile(optimizer='adam',
            loss='mean_squared_error',
            metrics=['accuracy'])
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=32, activation="relu", input_dim=9, kernel_initializer="uniform")`

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=32, activation="relu", kernel_initializer="uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:4: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=1, activation="sigmoid", kernel_initializer="uniform")`
  after removing the cwd from sys.path.
```

In [51]:
```python
ann.fit(X_train,y_train, batch_size=32, nb_epoch=300,verbose= 0)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  """Entry point for launching an IPython kernel.
```

Out[51]: <keras.callbacks.callbacks.History at 0x7fd782815e10>

In [52]:
```python
ann_pred = ann.predict(X_test)
ann_pred = [ 1 if y>=0.5 else 0 for y in ann_pred]
print(ann_pred)
```

```
[0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0,
1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0]
```

In [53]:
```python
print(confusion_matrix(y_test,ann_pred))
```

```
[[46  5]
 [13 26]]
```

In [54]:
```python
print(classification_report(y_test,ann_pred))
```

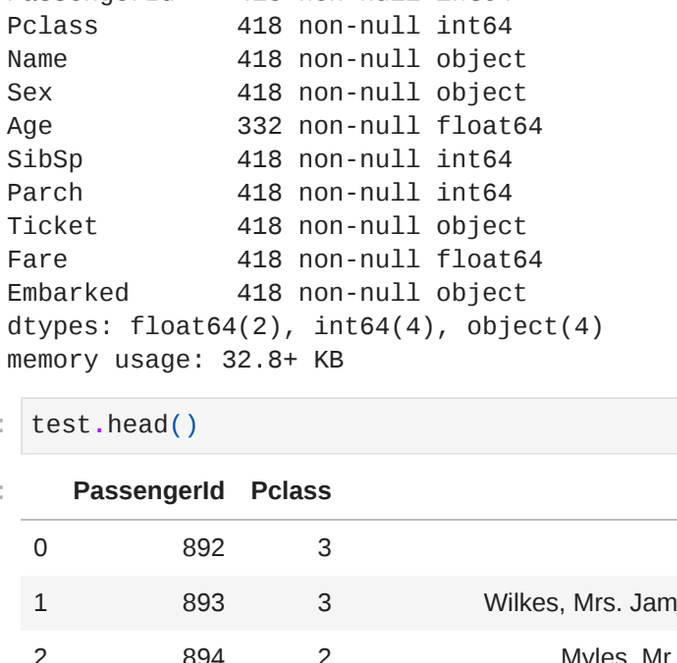
```
              precision    recall  f1-score   support

           0       0.78      0.90      0.84        51
           1       0.84      0.67      0.74        39

    accuracy                           0.80        90
   macro avg       0.81      0.78      0.79        90
weighted avg       0.81      0.80      0.80        90
```

Now we will use test dataset

In [55]:
```python
test  = pd.read_csv('../input/test.csv')
```

In [56]:
```python
sns.heatmap(test.isnull())
```

Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd7543a8c18>



In [57]:
```python
test.drop('Cabin',axis=1,inplace=True)
```

In [58]:
```python
test['Fare'].fillna(test['Fare'].median(), inplace=True)
```

In [59]:
```python
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 10 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           418 non-null float64
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(4)
memory usage: 32.8+ KB
```

In [60]:
```python
test.head()
```

Out[60]:

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

In [61]:
```python
test['Age'] = test[['Age','Pclass']].apply(impute_age,axis=1)
```

In [62]:
```python
sex_test = pd.get_dummies(test['Sex'],drop_first=True)
embark_test= pd.get_dummies(test['Embarked'],drop_first=True)
```

In [63]:
```python
test.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace=True)
```

In [64]:
```python
test = pd.concat([test,sex_test,embark_test],axis=1)
```

In [65]:
```python
test.head()
```

Out[65]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	892	3	34.5	0	0	7.8292	1	1	0
1	893	3	47.0	1	0	7.0000	0	0	1
2	894	2	62.0	0	0	9.6875	1	1	0
3	895	3	27.0	0	0	8.6625	1	0	1
4	896	3	22.0	1	1	12.2875	0	0	1

In [66]:
```python
train.head()
```

Out[66]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	0	3	22.0	1	0	7.2500	1	0	1
1	2	1	1	38.0	1	0	71.2833	0	0	0
2	3	1	3	26.0	0	0	7.9250	0	0	1
3	4	1	1	35.0	1	0	53.1000	0	0	1
4	5	0	3	35.0	0	0	8.0500	1	0	1

In [67]:
```python
ann.fit(train.drop(['Survived'],axis=1),train['Survived'] , nb_epoch=300,verbose= 0)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  """Entry point for launching an IPython kernel.
```

Out[67]: <keras.callbacks.callbacks.History at 0x7fd74e528e10>

In [68]:
```python
test_prediction = ann.predict(test)
test_prediction = [ 1 if y>=0.5 else 0 for y in test_prediction]
```

In [69]:
```python
test_pred = pd.DataFrame(test_prediction, columns= ['Survived'])
```

In [70]:
```python
new_test = pd.concat([test, test_pred], axis=1, join='inner')
```

In [71]:
```python
new_test.head()
```

Out[71]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	male	Q	S	Survived
0	892	3	34.5	0	0	7.8292	1	1	0	0
1	893	3	47.0	1	0	7.0000	0	0	1	0
2	894	2	62.0	0	0	9.6875	1	1	0	0
3	895	3	27.0	0	0	8.6625	1	0	1	0
4	896	3	22.0	1	1	12.2875	0	0	1	1

In [72]:
```python
df= new_test[['PassengerId' ,'Survived']]
```

In [73]:
```python
df.head()
```

Out[73]:

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1