In [ ]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import Pipeline,make_pipeline
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.feature_selection import SelectKBest
from sklearn import cross_validation, metrics
from sklearn.grid_search import GridSearchCV, RandomizedSearchCV
import warnings

warnings.filterwarnings('ignore')
```

In [ ]:

```
print ("a")
```

In [ ]:

```
train = pd.read_csv('train.csv',dtype={"Age": np.float64})
test = pd.read_csv('test.csv',dtype={"Age": np.float64})
PassengerId=test['PassengerId']
all_data = pd.concat([train, test], ignore_index = True)
```

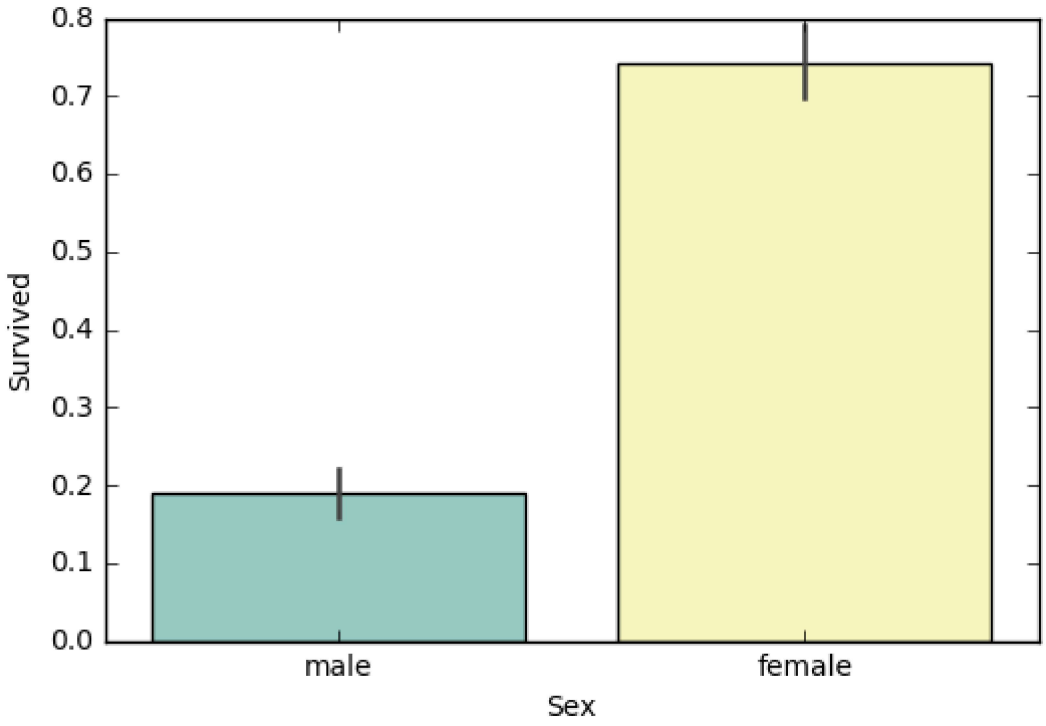
In [ ]:

```
                                    #1.数据可视化
```

In [120]:

```
#    1)Sex Feature：女性幸存率远高于男性
sns.barplot(x="Sex", y="Survived", data=train, palette='Set3')
print("Percentage of females who survived:%.2f" % (train["Survived"][train["Sex"] == 'female'].value
print("Percentage of males who survived:%.2f" % (train["Survived"][train["Sex"] == 'male'].value_cou
```

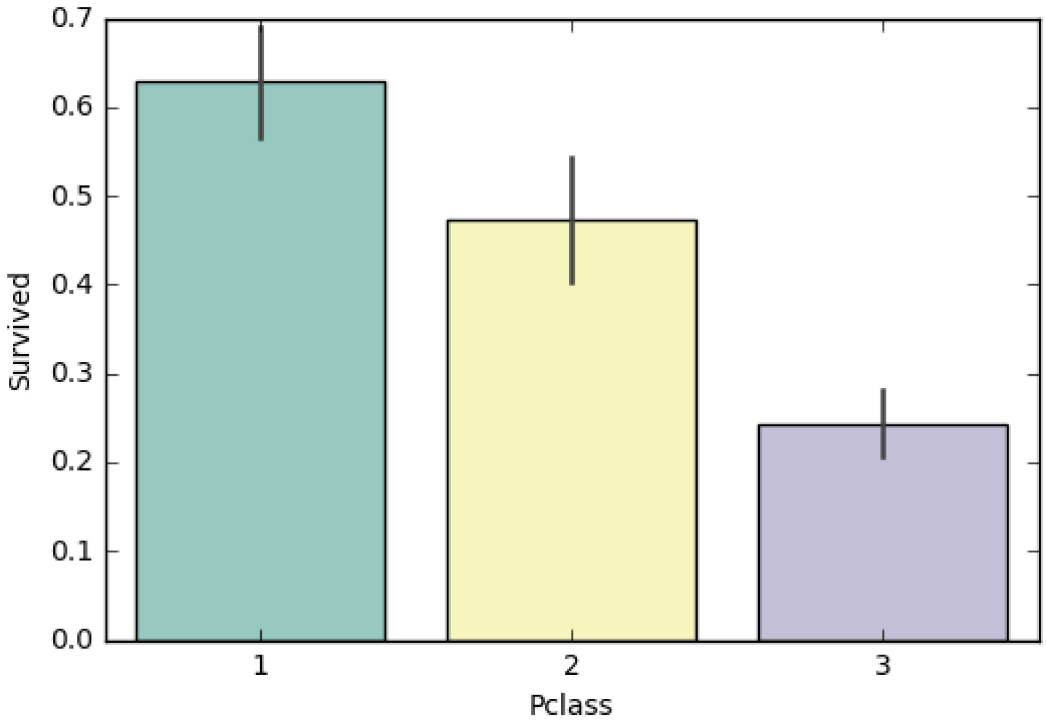
```
Percentage of females who survived:74.20
Percentage of males who survived:18.89
```



In [121]:

```
#    2)Pclass Feature：乘客社会等级越高，幸存率越高
sns.barplot(x="Pclass", y="Survived", data=train, palette='Set3')
print("Percentage of Pclass = 1 who survived:%.2f" % (train["Survived"][train["Pclass"] == 1].value_
print("Percentage of Pclass = 2 who survived:%.2f" % (train["Survived"][train["Pclass"] == 2].value_
print("Percentage of Pclass = 3 who survived:%.2f" % (train["Survived"][train["Pclass"] == 3].value_
```

```
Percentage of Pclass = 1 who survived:62.96
Percentage of Pclass = 2 who survived:47.28
Percentage of Pclass = 3 who survived:24.24
```

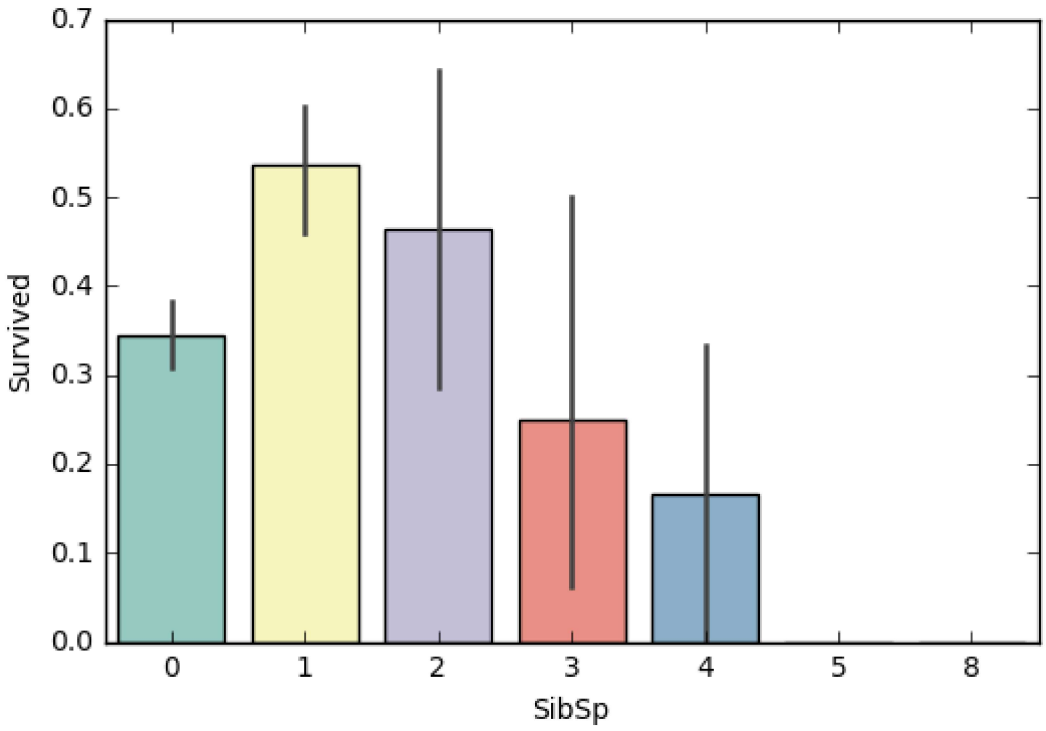


In [122]:

```
#    3)SibSp Feature：配偶及兄弟姐妹数适中的乘客幸存率更高
sns.barplot(x="SibSp", y="Survived", data=train, palette='Set3')
```

Out[122]:

<matplotlib.axes._subplots.AxesSubplot at 0x299aa3d5978>

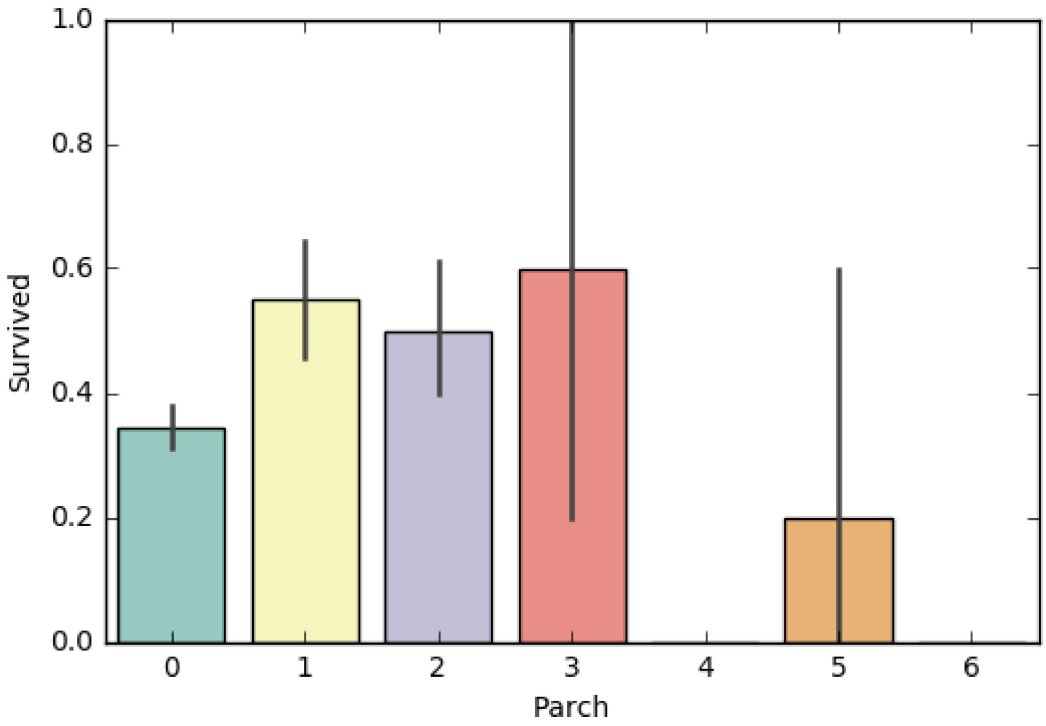


In [123]:

```
#    4)Parch Feature：父母与子女数适中的乘客幸存率更高
sns.barplot(x="Parch", y="Survived", data=train, palette='Set3')
```

Out[123]:

<matplotlib.axes._subplots.AxesSubplot at 0x299a9c777f0>

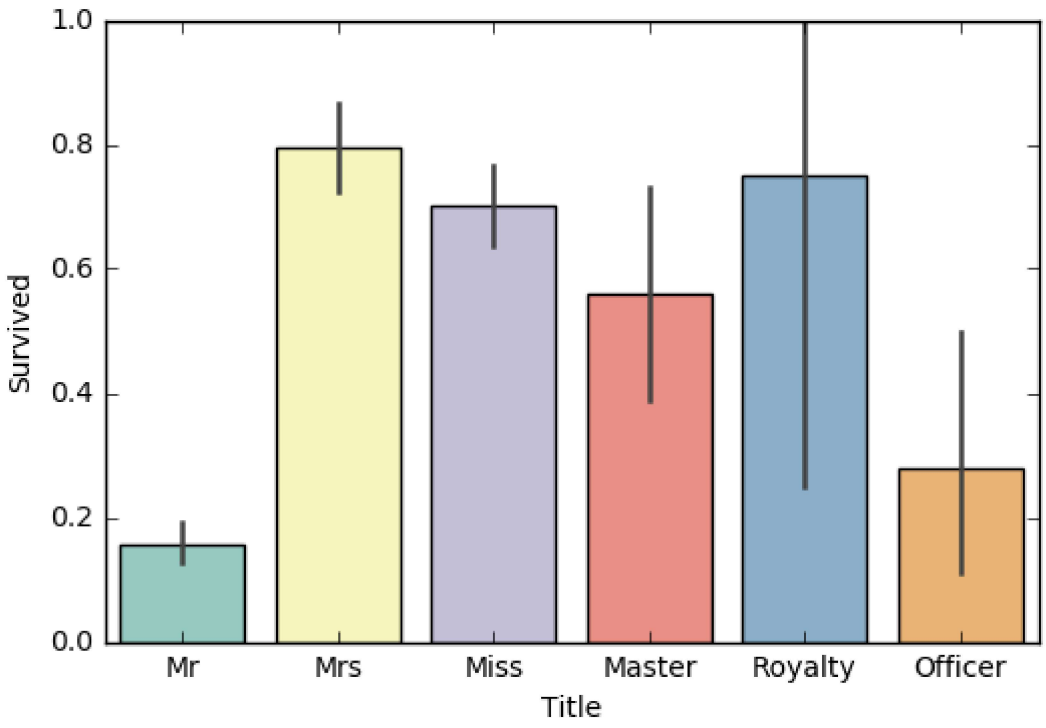


In [124]:

```
#7)Title Feature(New)：不同称呼的乘客幸存率不同
all_data['Title'] = all_data['Name'].apply(lambda x:x.split(',')[1].split('.')[0].strip())
Title_Dict = {}
Title_Dict.update(dict.fromkeys(['Capt', 'Col', 'Major', 'Dr', 'Rev'], 'Officer'))
Title_Dict.update(dict.fromkeys(['Don', 'Sir', 'the Countess', 'Dona', 'Lady'], 'Royalty'))
Title_Dict.update(dict.fromkeys(['Mme', 'Ms', 'Mrs'], 'Mrs'))
Title_Dict.update(dict.fromkeys(['Mlle', 'Miss'], 'Miss'))
Title_Dict.update(dict.fromkeys(['Mr'], 'Mr'))
Title_Dict.update(dict.fromkeys(['Master','Jonkheer'], 'Master'))
all_data['Title'] = all_data['Title'].map(Title_Dict)
sns.barplot(x="Title", y="Survived", data=all_data, palette='Set3')
```

Out[124]:

<matplotlib.axes._subplots.AxesSubplot at 0x299ab8d36d8>

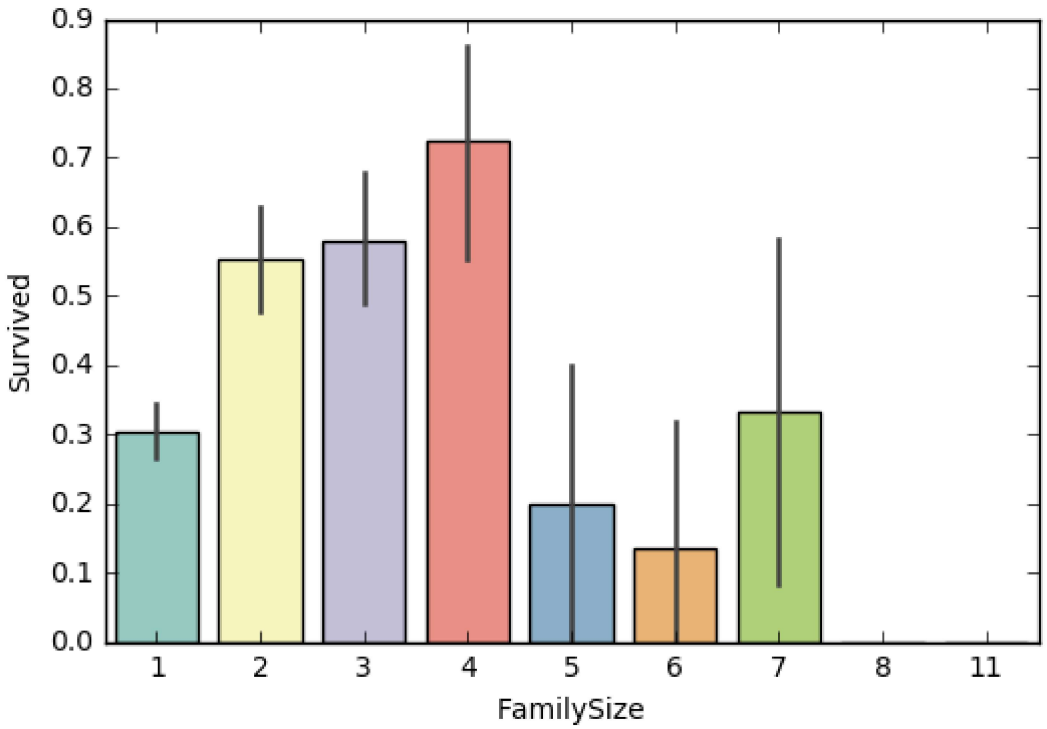


In [125]:

```
#8)FamilyLabel Feature(New)：家庭人数为2到4的乘客幸存率较高
#新增FamilyLabel特征，先计算FamilySize=Parch+SibSp+1，然后把FamilySize分为三类。
all_data['FamilySize']=all_data['SibSp']+all_data['Parch']+1
sns.barplot(x="FamilySize", y="Survived", data=all_data, palette='Set3')
```

Out[125]:

<matplotlib.axes._subplots.AxesSubplot at 0x299ab92a3c8>

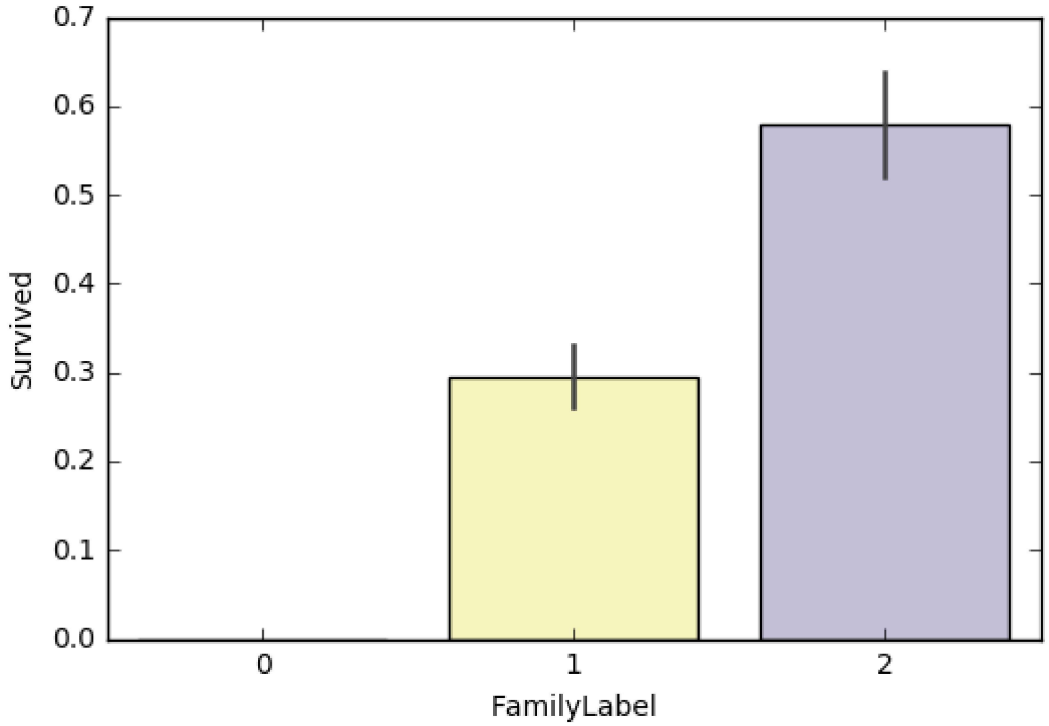


In [126]:

```
def Fam_label(s):
    if (s >= 2) & (s <= 4):
        return 2
    elif ((s > 4) & (s <= 7)) | (s == 1):
        return 1
    elif (s > 7):
        return 0
all_data['FamilyLabel']=all_data['FamilySize'].apply(Fam_label)
sns.barplot(x="FamilyLabel", y="Survived", data=all_data, palette='Set3')
```

Out[126]:

<matplotlib.axes._subplots.AxesSubplot at 0x299ab832d68>



In [127]:

```
all_data.head()
```

Out[127]:

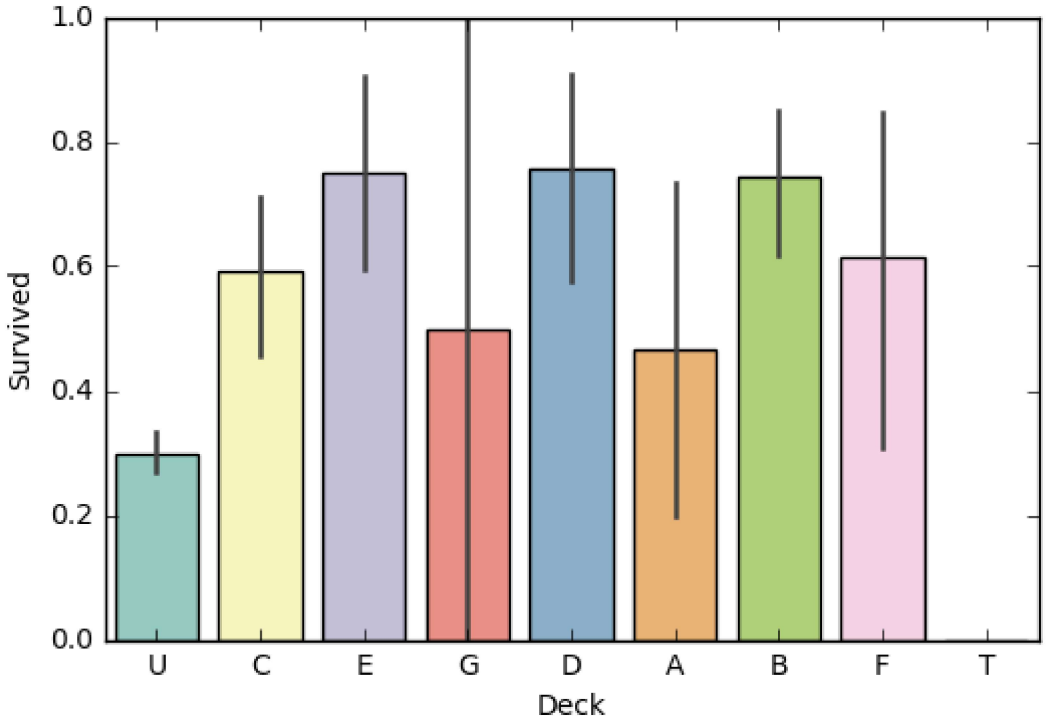
	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibS
0	22.0	NaN	S	7.2500	Braund, Mr. Owen Harris	0	1	3	male	1
1	38.0	C85	C	71.2833	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	2	1	female	1
2	26.0	NaN	S	7.9250	Heikkinen, Miss. Laina	0	3	3	female	0
3	35.0	C123	S	53.1000	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	4	1	female	1
4	35.0	NaN	S	8.0500	Allen, Mr. William Henry	0	5	3	male	0

In [128]:

```
#9)Deck Feature(New)：不同甲板的乘客幸存率不同
#新增Deck特征，先把Cabin空缺值填充为'Unknown'，再提取Cabin中的首字母构成乘客的甲板号。
all_data['Cabin'] = all_data['Cabin'].fillna('Unknown')
all_data['Deck']=all_data['Cabin'].str.get(0)
sns.barplot(x="Deck", y="Survived", data=all_data, palette='Set3')
```

Out[128]:

<matplotlib.axes._subplots.AxesSubplot at 0x299acb4d940>



In [129]:

```
all_data.head()
```

Out[129]:

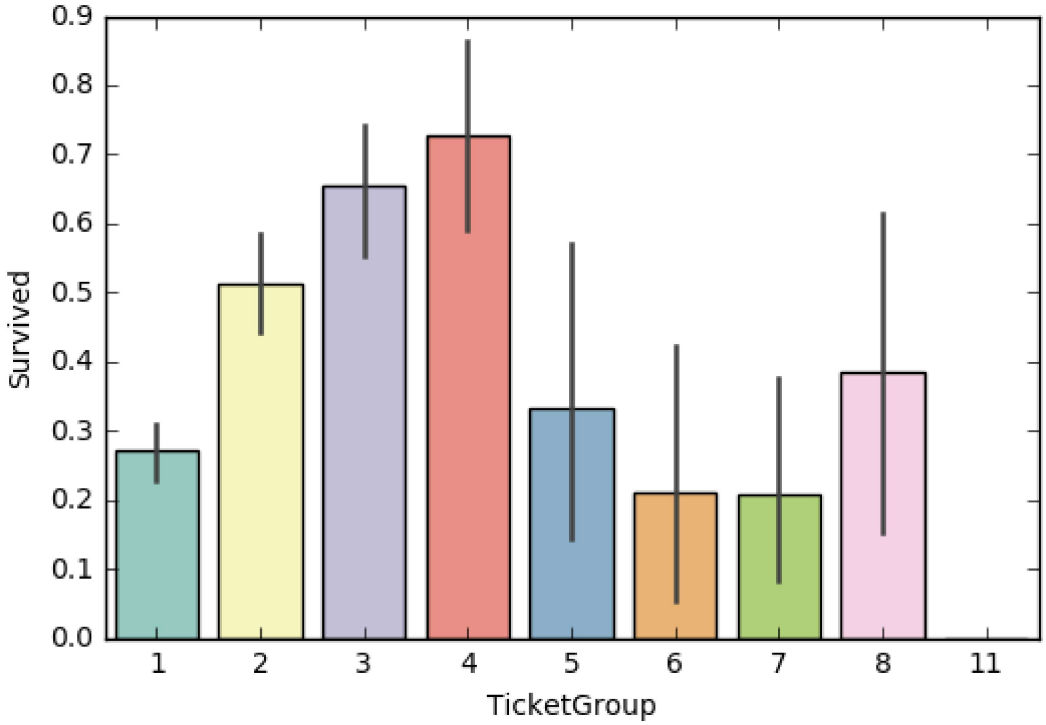
	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	Si
0	22.0	Unknown	S	7.2500	Braund, Mr. Owen Harris	0	1	3	male	1
1	38.0	C85	C	71.2833	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	2	1	female	1
2	26.0	Unknown	S	7.9250	Heikkinen, Miss. Laina	0	3	3	female	0
3	35.0	C123	S	53.1000	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	4	1	female	1
4	35.0	Unknown	S	8.0500	Allen, Mr. William Henry	0	5	3	male	0

In [130]:

```
#10)TicketGroup Feature(New)：与2至4人共票号的乘客幸存率较高
#新增TicketGroup特征，统计每个乘客的共票号数。
Ticket_Count = dict(all_data['Ticket'].value_counts())
all_data['TicketGroup'] = all_data['Ticket'].apply(lambda x:Ticket_Count[x])
sns.barplot(x='TicketGroup', y='Survived', data=all_data, palette='Set3')
```

Out[130]:

<matplotlib.axes._subplots.AxesSubplot at 0x299a9c74358>



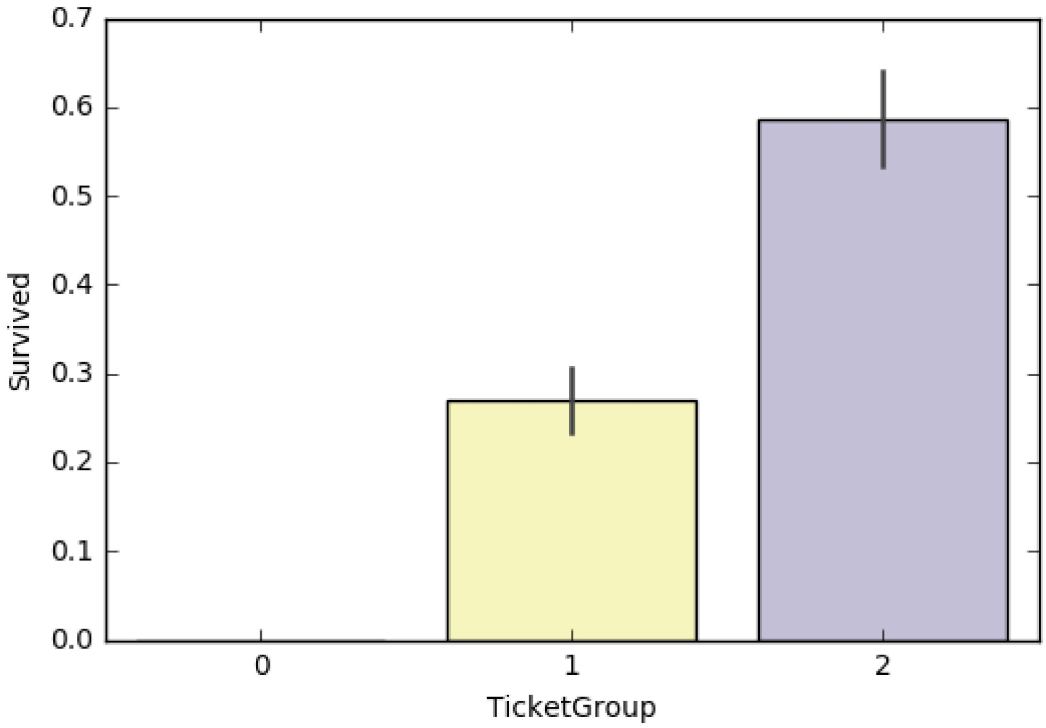
In [131]:

```
#按生存率把TicketGroup分为三类。
def Ticket_Label(s):
    if (s >= 2) & (s <= 4):
        return 2
    elif ((s > 4) & (s <= 8)) | (s == 1):
        return 1
    elif (s > 8):
        return 0

all_data['TicketGroup'] = all_data['TicketGroup'].apply(Ticket_Label)
sns.barplot(x='TicketGroup', y='Survived', data=all_data, palette='Set3')
```

Out[131]:

<matplotlib.axes._subplots.AxesSubplot at 0x299accc6d30>



In [132]:

```
all_data['TicketGroup'].values
```

Out[132]:

array([1, 2, 1, ..., 1, 1, 2], dtype=int64)

In [133]:

```
all_data.head()
```

Out[133]:

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	Si
0	22.0	Unknown	S	7.2500	Braund, Mr. Owen Harris	0	1	3	male	1
1	38.0	C85	C	71.2833	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	2	1	female	1
2	26.0	Unknown	S	7.9250	Heikkinen, Miss. Laina	0	3	3	female	0
3	35.0	C123	S	53.1000	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	4	1	female	1
4	35.0	Unknown	S	8.0500	Allen, Mr. William Henry	0	5	3	male	0

In [134]:

```
#处理缺失
```

In [135]:

```
#    1)Age Feature：Age缺失量为263，缺失量较大，用Sex, Title, Pclass三个特征构建随机森林模型，填充年
age_df = all_data[['Age', 'Pclass','Sex','Title']]
age_df=pd.get_dummies(age_df)
known_age = age_df[age_df.Age.notnull()].as_matrix()
unknown_age = age_df[age_df.Age.isnull()].as_matrix()
y = known_age[:, 0]
X = known_age[:, 1:]
rfr = RandomForestRegressor(random_state=0, n_estimators=100, n_jobs=-1)
rfr.fit(X, y)
predictedAges = rfr.predict(unknown_age[:, 1::])
all_data.loc[ (all_data.Age.isnull()), 'Age' ] = predictedAges
```

In [136]:

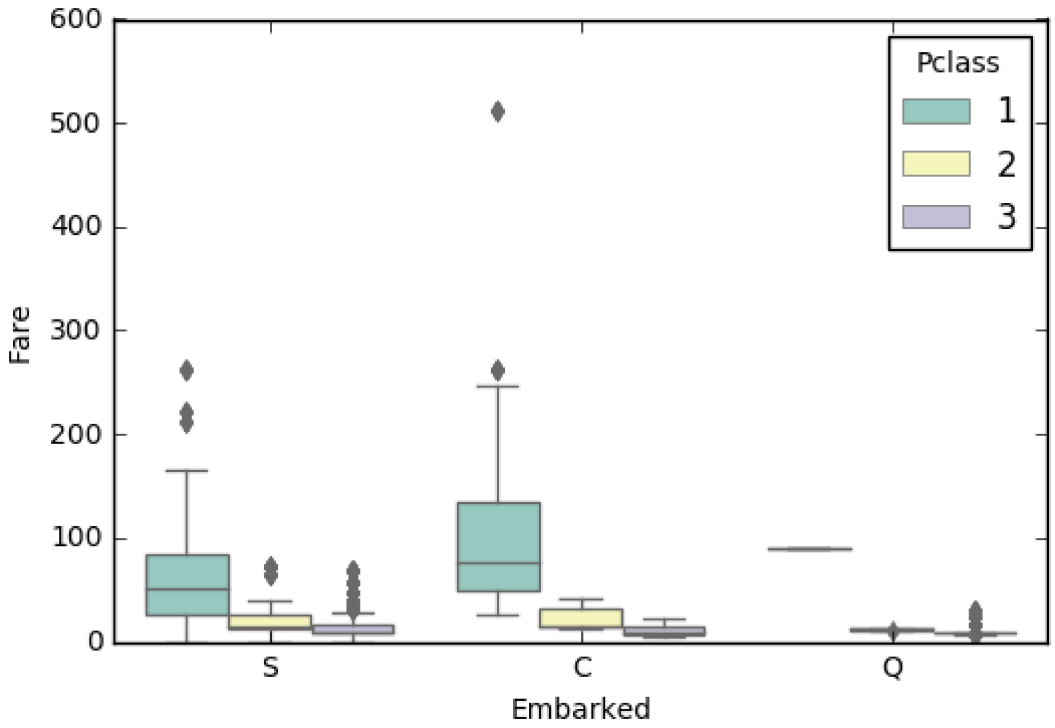
```
#    2)Embarked Feature：Embarked缺失量为2，
#缺失Embarked信息的乘客的Pclass均为1，且Fare均为80，因为Embarked为C且Pclass为1的乘客的Fare中位数为80
all_data[all_data['Embarked'].isnull()]
```

Out[136]:

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	S
61	38.0	B28	NaN	80.0	Icard, Miss. Amelie	0	62	1	female	0	1
829	62.0	B28	NaN	80.0	Stone, Mrs. George Nelson (Martha Evelyn)	0	830	1	female	0	1

In [137]:

```
sns.boxplot(x="Embarked", y="Fare", hue="Pclass",data=all_data, palette="Set3")
all_data['Embarked'] = all_data['Embarked'].fillna('C')
```



In [138]:

```
#    3)Fare Feature：Fare缺失量为1，缺失Fare信息的乘客的Embarked为S，Pclass为3，所以用Embarked为S，
all_data[all_data['Fare'].isnull()]
```

Out[138]:

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp
1043	60.5	Unknown	S	NaN	Storey, Mr. Thomas	0	1044	3	male	0

In [139]:

```
fare=all_data[(all_data['Embarked'] == "S") & (all_data['Pclass'] == 3)].Fare.median()
all_data['Fare']=all_data['Fare'].fillna(fare)
```

In [140]:

```
#同组识别
all_data['Surname']=all_data['Name'].apply(lambda x:x.split(',')[0].strip())
Surname_Count = dict(all_data['Surname'].value_counts())
all_data['FamilyGroup'] = all_data['Surname'].apply(lambda x:Surname_Count[x])
Female_Child_Group=all_data.loc[(all_data['FamilyGroup']>=2) & ((all_data['Age']<=12) | (all_data['S
Male_Adult_Group=all_data.loc[(all_data['FamilyGroup']>=2) & (all_data['Age']>12) & (all_data['Sex']
#发现绝大部分女性和儿童组的平均存活率都为1或0，即同组的女性和儿童要么全部幸存，要么全部遇难。
Female_Child=pd.DataFrame(Female_Child_Group.groupby('Surname')['Survived'].mean().value_counts())
Female_Child.columns=['GroupCount']
Female_Child
```

Out[140]:

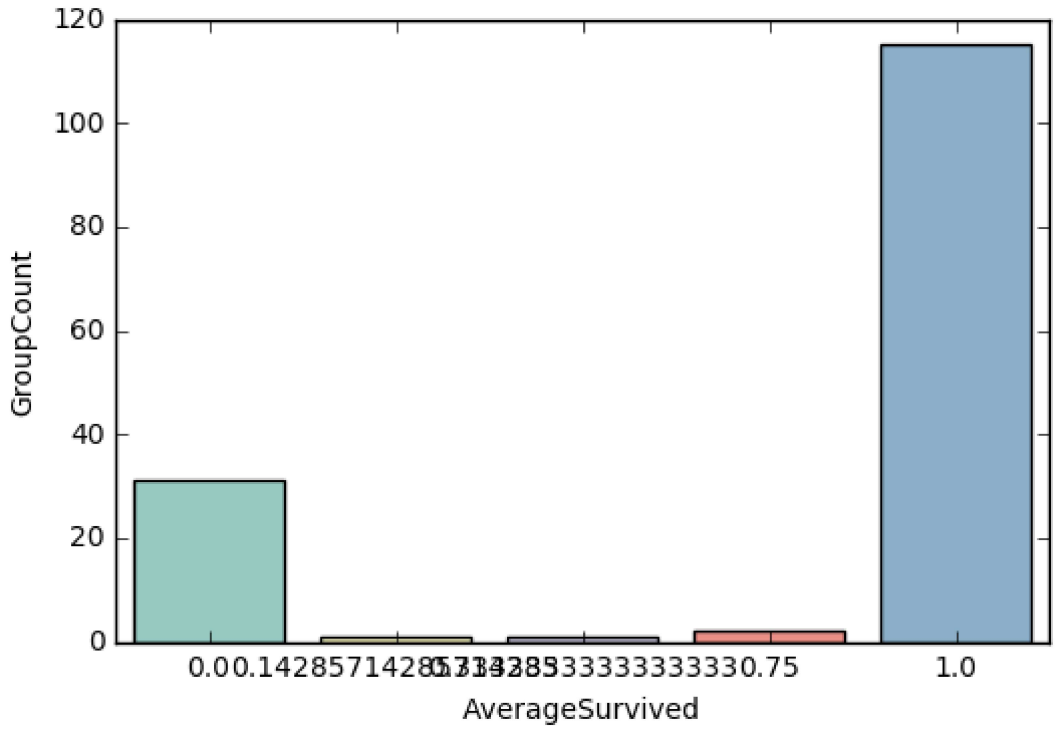
	GroupCount
1.000000	115
0.000000	31
0.750000	2
0.333333	1
0.142857	1

In [141]:

```
#绝大部分成年男性组的平均存活率也为1或0。
sns.barplot(x=Female_Child.index, y=Female_Child["GroupCount"], palette='Set3').set_xlabel('AverageS
```

Out[141]:

<matplotlib.text.Text at 0x299a9d32128>



In [142]:

```
Male_Adult=pd.DataFrame(Male_Adult_Group.groupby('Surname')['Survived'].mean().value_counts())
Male_Adult.columns=['GroupCount']
Male_Adult
```

Out[142]:

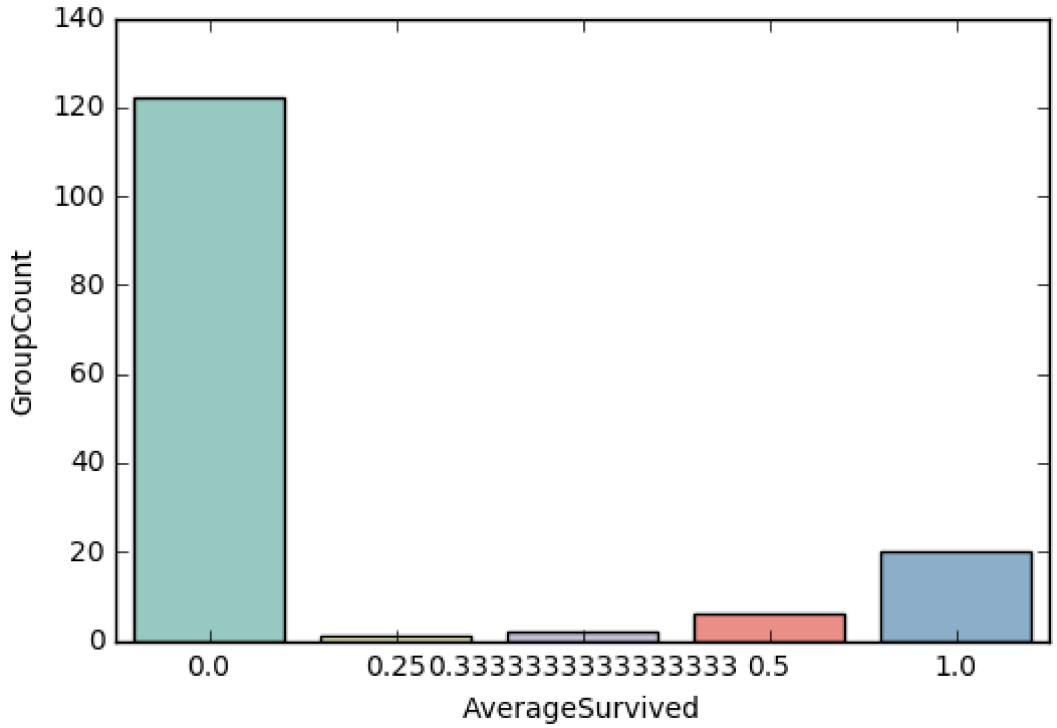
	GroupCount
0.000000	122
1.000000	20
0.500000	6
0.333333	2
0.250000	1

In [143]:

```
sns.barplot(x=Male_Adult.index, y=Male_Adult['GroupCount'], palette='Set3').set_xlabel('AverageSurvi
```

Out[143]:

<matplotlib.text.Text at 0x299a9cddd68>



In [144]:

```
#因为普遍规律是女性和儿童幸存率高，成年男性幸存较低，所以我们把不符合普遍规律的反常组选出来单独处理。
#把女性和儿童组中幸存率为0的组设置为遇难组，把成年男性组中存活率为1的设置为幸存组，
#推测处于遇难组的女性和儿童幸存的可能性较低，处于幸存组的成年男性幸存的可能性较高。
Female_Child_Group=Female_Child_Group.groupby('Surname')['Survived'].mean()
Dead_List=set(Female_Child_Group[Female_Child_Group.apply(lambda x:x==0)].index)
print(Dead_List)
Male_Adult_List=Male_Adult_Group.groupby('Surname')['Survived'].mean()
Survived_List=set(Male_Adult_List[Male_Adult_List.apply(lambda x:x==1)].index)
print(Survived_List)
```

```
{'Panula', 'Rice', 'Van Impe', 'Johnston', 'Olsson', 'Bourke', 'Skoog', 'Goodwin',
'Turpin', 'Ilmakangas', 'Zabour', 'Strom', 'Jussila', 'Ford', 'Lefebre', 'Lahtinen',
'Rosblom', 'Palsson', 'Attalah', 'Robins', 'Sage', 'Danbom', 'Canavan', 'Oreskovic',
'Arnold-Franchi', 'Caram', 'Barbara', 'Vander Planke', 'Boulos', 'Lobb', 'Cacic'}
{'Daly', 'Goldenberg', 'Nakid', 'Dick', 'Jonsson', 'Chambers', 'Frolicher-Stehli',
'Greenfield', 'Harder', 'Beane', 'Kimball', 'Bishop', 'Taylor', 'Beckwith', 'Cardez
a', 'Frauenthal', 'Bradley', 'Jussila', 'McCoy', 'Duff Gordon'}
```

In [145]:

```
#为了使处于这两种反常组中的样本能够被正确分类，对测试集中处于反常组中的样本的Age，Title，Sex进行惩罚
train=all_data.loc[all_data['Survived'].notnull()]
test=all_data.loc[all_data['Survived'].isnull()]
test.loc[(test['Surname'].apply(lambda x:x in Dead_List)),'Sex'] = 'male'
test.loc[(test['Surname'].apply(lambda x:x in Dead_List)),'Age'] = 60
test.loc[(test['Surname'].apply(lambda x:x in Dead_List)),'Title'] = 'Mr'
test.loc[(test['Surname'].apply(lambda x:x in Survived_List)),'Sex'] = 'female'
test.loc[(test['Surname'].apply(lambda x:x in Survived_List)),'Age'] = 5
test.loc[(test['Surname'].apply(lambda x:x in Survived_List)),'Title'] = 'Miss'
```

In [146]:

```
#3)特征转换：选取特征，转换为数值变量，划分训练集和测试集。
all_data=pd.concat([train, test])
all_data=all_data[['Survived','Pclass','Sex','Age','Fare','Embarked','Title','FamilyLabel','Deck','T
all_data=pd.get_dummies(all_data)
train=all_data[all_data['Survived'].notnull()]
test=all_data[all_data['Survived'].isnull()].drop('Survived',axis=1)
```

In [147]:

```
train['Deceased'] = train['Survived'].apply(lambda x: 1 - x)
dataset_Y=train[['Deceased', 'Survived']].values
train_Y=train[['Survived']].values
```

In [148]:

```
train =train.drop("Survived",axis=1).copy()
train =train.drop("Deceased",axis=1).copy()
```

In [149]:

```
dataset_X=train.values
train_X=dataset_X
```

In [150]:

```
test=test.values
```

In [151]:

```
test.shape
```

Out[151]:

(418, 25)

In [156]:

```
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import numpy as np

model = SVC(kernel='rbf', degree=3, gamma=1.7)
model.fit(train_X, train_Y)
print(model)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=1.7, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [157]:

```
pred_Y=model.predict(test)
pred_Y=pred_Y.astype(int)
predDf=pd.DataFrame({'PassengerId':PassengerId,'Survived':pred_Y})
predDf.head()
```

Out[157]:

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

In [158]:

```
predDf.to_csv("svm-test.csv",index=False)
```

In [36]:

```
x = tf.placeholder(tf.float32, shape=[None, 25])
y = tf.placeholder(tf.float32, shape=[None,2])
# 使用逻辑回归模型
#y = σ(wx + b)
weights = tf.Variable(tf.random_normal([25, 2]), name='weights')
bias = tf.Variable(tf.zeros([2]), name='bias')
y_pred = tf.nn.softmax(tf.matmul(x, weights) + bias)
# 定义交叉熵
cross_entropy = - tf.reduce_sum(y * tf.log(y_pred + 1e-10), reduction_indices=1)
#定义损失函数
cost = tf.reduce_mean(cross_entropy)
# 使用梯度下降优化算法最小化损失函数
lr = 0.001
train_op = tf.train.GradientDescentOptimizer(lr).minimize(cost)
```

In [38]:

```
#保存模型
saver = tf.train.Saver()
with tf.Session() as sess:
    print('start training')
    tf.global_variables_initializer().run()

    len_dataset=len(dataset_X)#891条训练数据
    total_epoch=50#总共训练50个epoch

    for epoch in range(total_epoch):
            total_loss = 0
            for i in range(len_dataset):
            # prepare feed data and run
                feed_dict = {x: [dataset_X[i]], y: [dataset_Y[i]]}
                var, loss = sess.run([train_op, cost], feed_dict=feed_dict)
                total_loss += loss
            # display loss per epoch
            model_pred=np.argmax(sess.run(y_pred, feed_dict={x: dataset_X}), 1)
            true=np.argmax(dataset_Y,1)
            correct_p = tf.equal(model_pred, true)
            #每一个epoch结束以后计算acc
            accuracy = tf.reduce_mean(tf.cast(correct_p, tf.float32))
            print('Epoch: %04d, acc=%.7f' % (epoch + 1,sess.run(accuracy)))
    #保存模型
    saver.save(sess, "./test.model")
    print("Train Complete")
```

```
start training
Epoch: 0001, acc=0.6285073
Epoch: 0002, acc=0.6599327
Epoch: 0003, acc=0.6722783
Epoch: 0004, acc=0.6745230
Epoch: 0005, acc=0.6980920
Epoch: 0006, acc=0.7070707
Epoch: 0007, acc=0.7081931
Epoch: 0008, acc=0.7138047
Epoch: 0009, acc=0.7250280
Epoch: 0010, acc=0.7340068
Epoch: 0011, acc=0.7362514
Epoch: 0012, acc=0.7441077
Epoch: 0013, acc=0.7542087
Epoch: 0014, acc=0.7631874
Epoch: 0015, acc=0.7643098
Epoch: 0016, acc=0.7508417
Epoch: 0017, acc=0.7519641
Epoch: 0018, acc=0.7553311
Epoch: 0019, acc=0.7542087
Epoch: 0020, acc=0.7553311
Epoch: 0021, acc=0.7519641
Epoch: 0022, acc=0.7564534
Epoch: 0023, acc=0.7564534
Epoch: 0024, acc=0.7586981
Epoch: 0025, acc=0.7564534
Epoch: 0026, acc=0.7564534
Epoch: 0027, acc=0.7609428
Epoch: 0028, acc=0.7586981
Epoch: 0029, acc=0.7957351
Epoch: 0030, acc=0.7968575
```

```
Epoch: 0031, acc=0.8035915
Epoch: 0032, acc=0.8035915
Epoch: 0033, acc=0.8035915
Epoch: 0034, acc=0.8035915
Epoch: 0035, acc=0.8058361
Epoch: 0036, acc=0.8069585
Epoch: 0037, acc=0.8035915
Epoch: 0038, acc=0.8013468
Epoch: 0039, acc=0.8024691
Epoch: 0040, acc=0.8002245
Epoch: 0041, acc=0.8013468
Epoch: 0042, acc=0.7991021
Epoch: 0043, acc=0.7979798
Epoch: 0044, acc=0.8002245
Epoch: 0045, acc=0.8013468
Epoch: 0046, acc=0.8013468
Epoch: 0047, acc=0.8013468
Epoch: 0048, acc=0.8024691
Epoch: 0049, acc=0.8013468
Epoch: 0050, acc=0.8013468
Train Complete
```

In [40]:

```
saver = tf.train.Saver()
with tf.Session() as sess:
    saver.restore(sess, "./test.model")
    # 测试模型
    predictions = np.argmax(sess.run(y_pred, feed_dict={x: test}), 1)
    # 保存结果
    submission = pd.DataFrame({
        "PassengerId": PassengerId,
        "Survived": predictions
    })
    submission.to_csv("titanic-submission-test.csv", index=False)
    print('result saved')
```

```
INFO:tensorflow:Restoring parameters from ./test.model
result saved
```

In [ ]:

```

```