In [146]:
```
for dataset in datasets:
    # Mapping Sex to binary
    dataset['Sex'] = dataset['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

    # Fill Embarked missing data to Mode
    dataset['Embarked'] = dataset['Embarked'].fillna(dataset['Embarked'].mode().i

    # Mapping Embarked
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).ast

    # New feature Family size
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1

    # Name Length
    dataset['Name_length'] = dataset['Name'].apply(len)

    # Is Alone?
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1

    # Map Family size to group
    dataset.loc[ dataset['FamilySize'] <= 4, 'FamilySize'] = 0
    dataset.loc[ dataset['FamilySize'] > 4, 'Fare'] = 1
    dataset['FamilySize'] = dataset['FamilySize'].astype(int)

    # Fill Fare missing data to Mean
    dataset['Fare'] = dataset['Fare'].fillna(dataset['Fare'].mean())

    # Extract Title from Name, replace french title to uniform title, replace rar
    dataset['Title'] = dataset['Name'].apply(get_title)
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col'
                                                 'Don', 'Dr', 'Major', 'Rev', 'Si
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

    # Mapping titles
    title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

    # Get avg std and null count of age
    age_avg = dataset['Age'].mean()
    age_std = dataset['Age'].std()
    age_null_count = dataset['Age'].isnull().sum()

    # Generate random age within 95% confidence interval
    age_null_random_list = np.random.randint(age_avg - 1.96*age_std/(age_null_cou
    dataset['Age'][np.isnan(dataset['Age'])] = age_null_random_list
    dataset['Age'] = dataset['Age'].astype(int)

    # Map Age to group
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
```

```
    dataset.loc[ dataset['Age'] > 64, 'Age'] = 4

    # normalize Fare and name_length
    transf = dataset.Fare.reshape(-1,1)
    scaler = preprocessing.StandardScaler().fit(transf)
    dataset['Fare'] = scaler.transform(transf)
    transf = dataset.Name_length.reshape(-1,1)
    scaler = preprocessing.StandardScaler().fit(transf)
    dataset['Name_length'] = scaler.transform(transf)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:49: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:59: FutureWarn
ing: reshape is deprecated and will raise in a subsequent release. Please use .
values.reshape(...) instead
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:62: FutureWarn
ing: reshape is deprecated and will raise in a subsequent release. Please use .
values.reshape(...) instead
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:429: Dat
aConversionWarning: Data with input dtype int64 was converted to float64 by Sta
ndardScaler.
  warnings.warn(msg, _DataConversionWarning)
```

In [147]:
```
drop_elements = ['Cabin','Ticket','PassengerId','Name','SibSp','Parch']
training_data = training_data.drop(drop_elements, axis = 1)
test_data = test_data.drop(drop_elements, axis=1)
```

In [148]:
```
test_data.head()
```

Out[148]:

	Pclass	Sex	Age	Fare	Embarked	FamilySize	Name_length	IsAlone	Title
0	3	1	2	-0.463384	2	0	-1.153019	1	1
1	3	0	2	-0.479915	0	0	0.453521	0	3
2	2	1	3	-0.426337	2	0	-0.249340	1	1
3	3	1	1	-0.446771	0	0	-1.153019	1	1
4	3	0	1	-0.374504	0	0	1.658426	0	3

In [149]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 9 columns):
Pclass         418 non-null int64
Sex            418 non-null int32
Age            418 non-null int32
Fare           418 non-null float64
Embarked       418 non-null int32
FamilySize     418 non-null int32
Name_length    418 non-null float64
IsAlone        418 non-null int64
Title          418 non-null int64
dtypes: float64(2), int32(4), int64(3)
memory usage: 22.9 KB
```

In [154]:
```
train=training_data.values
X = train[0::, 1::]
Y = train[0::, 0]
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,random_state=1)
```

In [155]:
```
Data_pca=PCA(n_components=2).fit(X_train)
X_train_pca=Data_pca.transform(X_train)
X_test_pca=Data_pca.transform(X_test)
```

In [153]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(3)
fitted_knn=knn.fit(X_train,Y_train)
fitted_knn.score(X_test,Y_test)
```

Out[153]: 0.81165919282511212

In [165]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
fitted_dtc=dtc.fit(X_train,Y_train)
fitted_dtc.score(X_test,Y_test)
```

Out[165]: 0.7488789237668162

In [169]:
```
from sklearn.svm import SVC
svc=SVC(probability=True)
fitted_svc=svc.fit(X_train,Y_train)
fitted_svc.score(X_test,Y_test)
```

Out[169]: 0.80269058295964124

In [171]:
```
from sklearn.linear_model import LogisticRegression
log=LogisticRegression()
fitted_log=log.fit(X_train,Y_train)
fitted_log.score(X_test,Y_test)
```

Out[171]: 0.80269058295964124

In [180]:
```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientB
rfc=RandomForestClassifier()
fitted_rfc=rfc.fit(X_train,Y_train)
fitted_rfc.score(X_test,Y_test)
```

Out[180]: 0.81614349775784756

In [181]:
```
abc=AdaBoostClassifier()
fitted_abc=abc.fit(X_train,Y_train)
fitted_abc.score(X_test,Y_test)
```

Out[181]: 0.76681614349775784

In [182]:
```
gbc=GradientBoostingClassifier()
fitted_gbc=gbc.fit(X_train,Y_train)
fitted_gbc.score(X_test,Y_test)
```

Out[182]: 0.79372197309417036

In [183]:
```
test=test_data.values
prediction=fitted_rfc.predict(test)
prediction=prediction.astype(int)
prediction
```

Out[183]:
```
array([0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 1])
```

In [184]:
```
pdf=pd.DataFrame(prediction,columns=['Survived'])
source_test_data = pd.read_csv('test.csv')
pdf=pd.concat([source_test_data['PassengerId'],pdf],axis=1)
pdf.to_csv('submission.csv',index=False)
```

In [ ]:
```

```