Titanic

September 8, 2017

In [1]:
```
% matplotlib inline
import os
import numpy as np
import random as rnd
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.nonparametric.kde import KDEUnivariate
from statsmodels.nonparametric import smoothers_lowess
from pandas import Series, DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
```

```
/Users/disheng/miniconda3/lib/python3.5/site-packages/statsmodels/compat/pandas.py:56: FutureWar
  from pandas.core import datetools
```

In [2]:
```
titanTrain = pd.read_csv("/Users/disheng/Desktop/Machine Learning/hw1/train.csv")
titanTest = pd.read_csv("/Users/disheng/Desktop/Machine Learning/hw1/test.csv")
```

In [3]:
```
print(titanTrain.columns.values)
```

```
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

In [4]:
```
#Replace the missing age with age median for a good estimate
titanTrain['Age'].fillna(titanTrain['Age'].median(), inplace=True)
```

```
In [5]: titanTrain.describe()
```

```
Out[5]:        PassengerId    Survived      Pclass         Age       SibSp  \
        count   891.000000  891.000000  891.000000  891.000000  891.000000
        mean    446.000000    0.383838    2.308642   29.361582    0.523008
        std     257.353842    0.486592    0.836071   13.019697    1.102743
        min       1.000000    0.000000    1.000000    0.420000    0.000000
        25%     223.500000    0.000000    2.000000   22.000000    0.000000
        50%     446.000000    0.000000    3.000000   28.000000    0.000000
        75%     668.500000    1.000000    3.000000   35.000000    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
```

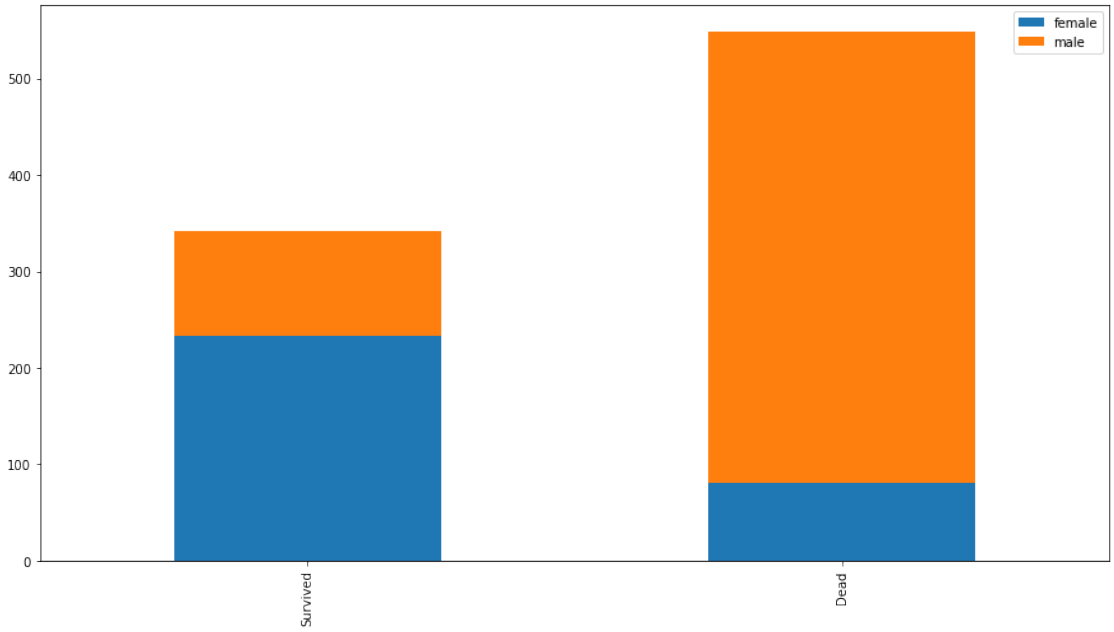
```
In [6]: titanTrain.info()
        #Most of cabins are missing value. We will not use it as part of our analysis.
```

```
<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            891 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 [7]: survived = titanTrain[titanTrain['Survived']==1]['Sex'].value_counts()
        dead = titanTrain[titanTrain['Survived']==0]['Sex'].value_counts()
        df = pd.DataFrame([survived ,dead])
        df.index = ['Survived','Dead']
```

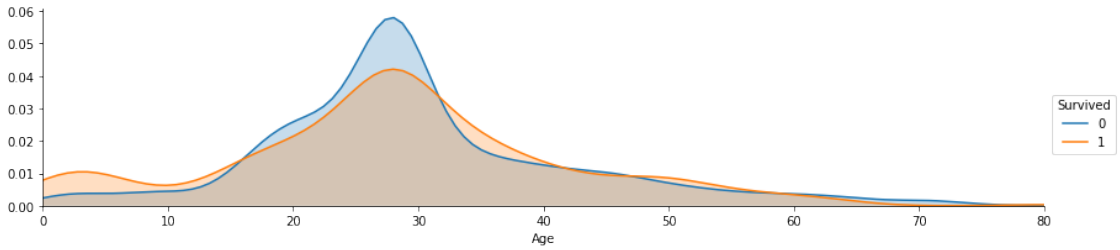
```
df.plot(kind='bar',stacked=True, figsize=(15,8))
#Looks like gender plays a very important factor on survival rate. Women are much more l
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x10b716630>

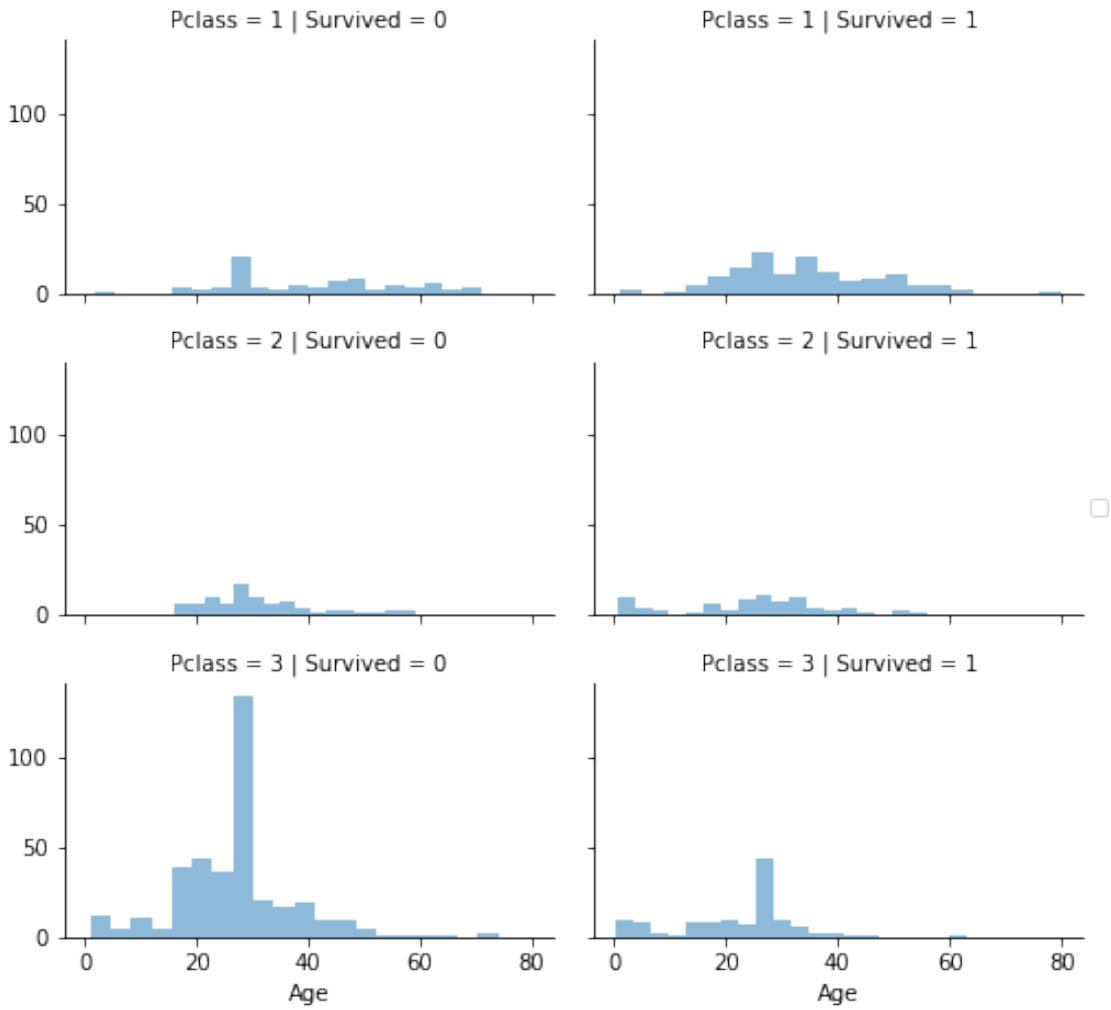


In [8]:
```
ageRate = sns.FacetGrid(titanTrain, hue="Survived",aspect=4)
ageRate.map(sns.kdeplot,'Age',shade= True)
ageRate.set(xlim=(0, titanTrain['Age'].max()))
ageRate.add_legend()
#I looks like youngers (especially age less than 12) are likely to survive more than eld
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0x10b80b7b8>

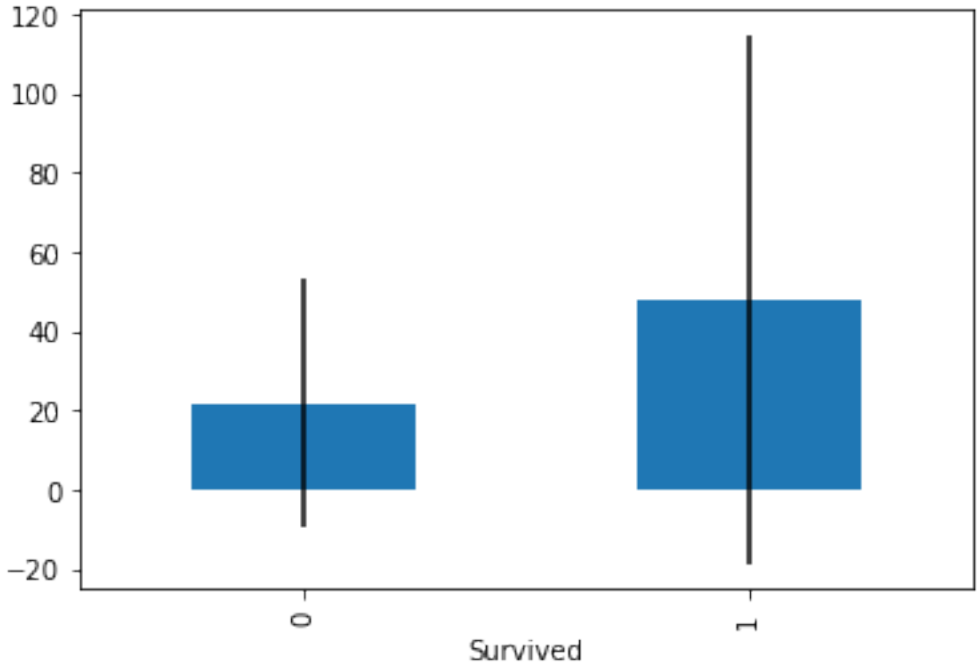


In [9]:
```
Pclassrate = sns.FacetGrid(titanTrain, col='Survived', row='Pclass', size=2.2, aspect=1.
Pclassrate.map(plt.hist, 'Age', alpha=.5, bins=20)
Pclassrate.add_legend();
#Looks like Class 1 has the most survival rate, and Class 3 has the lowest.
#It can be seen that ticket fare has a stong correlation with passenger class.
```



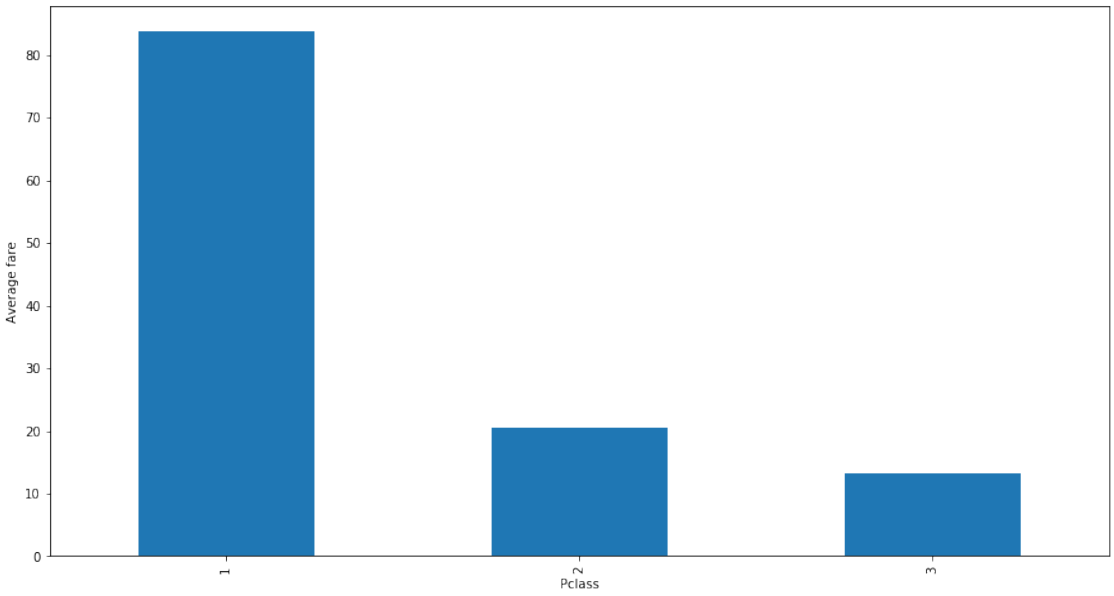
In [10]:
```
#Fare analysis. Fill NA fares value with median fare for better estimation
titanTest["Fare"].fillna(titanTest["Fare"].median(), inplace=True)
titanTrain['Fare'] = titanTrain['Fare'].astype(int)
titanTest['Fare']    = titanTest['Fare'].astype(int)

#Estimate survival based one fare rate.
fare_dead = titanTrain["Fare"][titanTrain["Survived"] == 0]
fare_survive     = titanTrain["Fare"][titanTrain["Survived"] == 1]
fareAvg = DataFrame([fare_dead.mean(), fare_survive.mean()])
fareSTD = DataFrame([fare_dead.std(), fare_survive.std()])

#Plot fare rate vs. survival.
fareAvg.index.names = fareSTD.index.names = ["Survived"]
fareAvg.plot(yerr = fareSTD,kind ='bar',legend=False)
#Based on the graph, people with higher ticket price are more likely to survive Titanic
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x10c1ca9e8>



In [11]:
```
PclassRate = plt.subplot()
PclassRate.set_ylabel('Average fare')
titanTrain.groupby('Pclass').mean()['Fare'].plot(kind='bar',figsize=(15,8), ax = Pclass
#It can be seen that ticket fare has a stong correlation with passenger class.
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x10c183160>

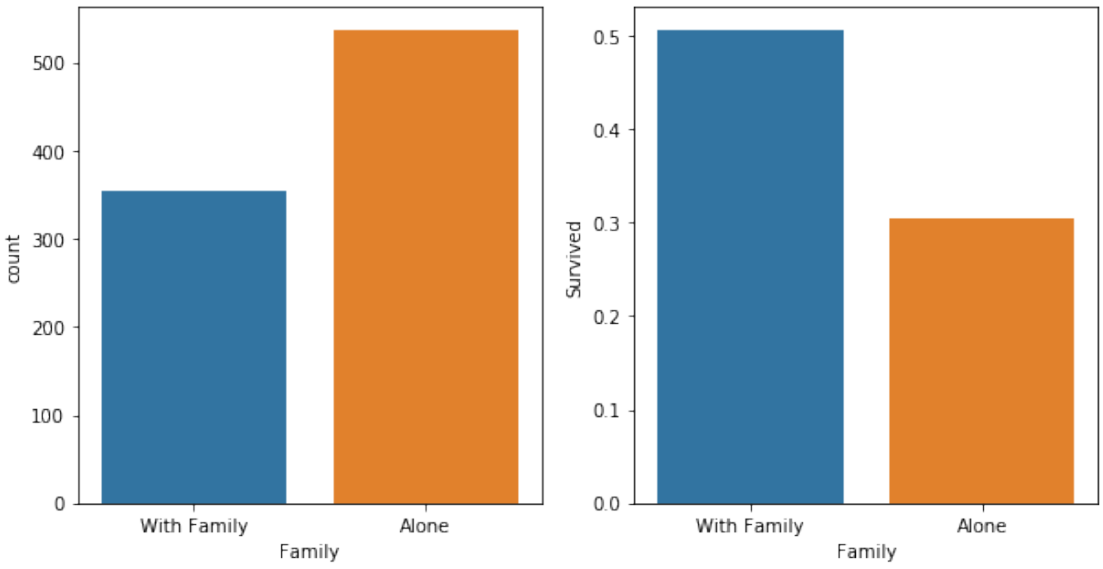


In [12]:
```
#Combining siblings and kids into a category called travel with Family then drop SibSp
titanTrain['Family'] = titanTrain["Parch"] + titanTrain["SibSp"]
titanTrain['Family'].loc[titanTrain['Family'] > 0] = 1
titanTrain['Family'].loc[titanTrain['Family'] == 0] = 0
titanTest['Family'] =  titanTest["Parch"] + titanTrain["SibSp"]
titanTest['Family'].loc[titanTest['Family'] > 0] = 1
titanTest['Family'].loc[titanTest['Family'] == 0] = 0

titanTrain = titanTrain.drop(['SibSp','Parch'], axis=1)
titanTest = titanTest.drop(['SibSp','Parch'], axis=1)
```

```
/Users/disheng/miniconda3/lib/python3.5/site-packages/pandas/core/indexing.py:179: SettingWithCo
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/stable/indexing.html#
  self._setitem_with_indexer(indexer, value)
```

In [13]:
```
fig, (pic1,pic2) = plt.subplots(1,2,sharex=True,figsize=(10,5))
sns.countplot(x='Family', data=titanTrain, order=[1,0], ax = pic1)

#plot survival rate vs. travel with or without family.
withFamily = titanTrain[["Family", "Survived"]].groupby(['Family'],as_index=False).mean
sns.barplot(x='Family', y='Survived', data=withFamily, order=[1,0], ax = pic2)

pic1.set_xticklabels(["With Family","Alone"], rotation=0)
#It can be inferred that those who travel with their families had a higher chanse of su
```

Out[13]:
```
[<matplotlib.text.Text at 0x10c893d68>, <matplotlib.text.Text at 0x10c83c080>]
```



In [14]:
```
#Fill the Na values with appropriate values
titanTest['Age'].fillna(titanTest['Age'].median(), inplace=True)
titanTrain["Embarked"] = titanTrain["Embarked"].fillna("S")
lb = LabelEncoder()
titanTrain['Embarked'] = lb.fit_transform(titanTrain['Embarked'])
titanTest['Embarked'] = lb.fit_transform(titanTest['Embarked'])
titanTrain['Sex'] = lb.fit_transform(titanTrain['Sex'])
titanTest['Sex'] = lb.fit_transform(titanTest['Sex'])
```

In [15]:
```
#Trim away irrelevant information would help to reduce the data size. It's better for c
titanTrain = titanTrain.drop(['PassengerId','Name','Ticket','Cabin'], axis=1)
titanTest = titanTest.drop(['Name','Ticket','Cabin'], axis=1)
X_train = titanTrain.drop("Survived", axis=1)
Y_train = titanTrain["Survived"]
X_test  = titanTest.drop("PassengerId", axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

Out[15]:
```
((891, 6), (891,), (418, 6))
```

In [16]:
```
#Logistic rate
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
```

Out[16]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [17]:
```
#The prediction rate for LogisticRegression
Y_pred = logreg.predict(X_test)

logreg.score(X_train, Y_train)
```

Out[17]:
```
0.7991021324354658
```

In [18]:
```
# Getting Correlation Coefficient for each feature using Logistic Regression
coeff_df = DataFrame(titanTrain.columns.delete(0))
coeff_df.columns = ['Features']
coeff_df["Coefficient Estimate"] = pd.Series(logreg.coef_[0])

coeff_df
```

Out[18]:

	Features	Coefficient Estimate
0	Pclass	-0.892844
1	Sex	-2.369238
2	Age	-0.022871
3	Fare	0.001934
4	Embarked	-0.194229
5	Family	0.034551