In [1]:
```
# This is solution for Kaggle tutorial
# Titanic: Machine Learning from Disaster
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# machine learning
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
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
```

In [2]:
```
training_data=pd.read_csv("train.csv")
test_data=pd.read_csv("test.csv")
gender_submission=pd.read_csv("gender_submission.csv")
```

In [3]:
```
training_data.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 [4]:
```
training_data
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1	5	347077	31.3875	NaN	S
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.8958	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
861	862	0	2	Giles, Mr. Frederick Edward	male	21.0	1	0	28134	11.5000	NaN	S
862	863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0	0	17466	25.9292	D17	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	NaN	S
864	865	0	2	Gill, Mr. John William	male	24.0	0	0	233866	13.0000	NaN	S
865	866	1	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.0000	NaN	S
866	867	1	2	Duran y More, Miss. Asuncion	female	27.0	1	0	SC/PARIS 2149	13.8583	NaN	C
867	868	0	1	Roebling, Mr. Washington Augustus II	male	31.0	0	0	PC 17590	50.4958	A24	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
869	870	1	3	Johnson, Master. Harold Theodor	male	4.0	1	1	347742	11.1333	NaN	S
870	871	0	3	Balkic, Mr. Cerin	male	26.0	0	0	349248	7.8958	NaN	S
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	NaN	S
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
876	877	0	3	Gustafsson, Mr. Alfred Ossian	male	20.0	0	0	7534	9.8458	NaN	S
877	878	0	3	Petroff, Mr. Nedelio	male	19.0	0	0	349212	7.8958	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [5]:
```
training_data[['Pclass','Survived']].groupby(['Pclass'],as_index=False).mean().sort_values(by='Survived',ascending=False)
```

Out[5]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

In [6]:
```
training_data['Family']=training_data['SibSp']+training_data['Parch']
test_data['Family']=test_data['SibSp']+test_data['Parch']
```

In [7]:
```
training_data[['Family','Survived']].groupby(['Family'],as_index=False).mean().sort_values(by='Survived',ascending=False)
```

Out[7]:

	Family	Survived
3	3	0.724138
2	2	0.578431
1	1	0.552795
6	6	0.333333
0	0	0.303538
4	4	0.200000
5	5	0.136364
7	7	0.000000
8	10	0.000000

In [8]:
```
training_data=training_data.drop(['Cabin','Ticket'],axis=1)
test_data=test_data.drop(['Cabin','Ticket'],axis=1)
```

In [9]:
```
training_data['Title']=training_data.Name.str.extract('([A-Za-z]+)\.',expand=False)
test_data['Title']=test_data.Name.str.extract('([A-Za-z]+)\.',expand=False)
```

In [10]:
```
training_data['Title'] = training_data['Title'].replace(['Capt','Col','Countess','Don','Dr','Jonkheer','Lady','Major','Rev','Sir'],'Rare')
test_data['Title'] = test_data['Title'].replace(['Capt','Col','Countess','Don','Dr','Jonkheer','Lady','Major','Rev','Sir'],'Rare')
```

In [11]:
```
training_data['Title'] = training_data['Title'].replace(['Mlle','Mme','Ms'],'Miss')
test_data['Title'] = test_data['Title'].replace(['Mlle','Mme','Ms'],'Miss')
```

In [12]:
```
title_mapping={"Mr":1,"Miss":2,"Master":3,"Mr":4,"Rare":5}
training_data['Title']=training_data['Title'].map(title_mapping)
training_data['Title']=training_data['Title'].fillna(0)
```

In [13]:
```
test_data['Title']=test_data['Title'].map(title_mapping)
test_data['Title']=test_data['Title'].fillna(0)
```

In [14]:
```
training_data=training_data.drop(['Name'],axis=1)
test_data=test_data.drop(['Name'],axis=1)
```

In [15]:
```
Sex_mapping={"male":0,"female":1}
training_data['Sex']=training_data['Sex'].map(Sex_mapping)
test_data['Sex']=test_data['Sex'].map(Sex_mapping)
```

In [16]:
```
training_data['IsAlone']=0
test_data['IsAlone']=0
training_data.loc[training_data['Family']==0,'IsAlone']=1
test_data.loc[test_data['Family']==0,'IsAlone']=1
```

In [17]:
```
training_data=training_data.drop(['SibSp','Parch','Family'],axis=1)
```

In [18]:
```
test_data=test_data.drop(['SibSp','Parch','Family'],axis=1)
```

In [19]:
```
freq_port=training_data.Embarked.dropna().mode()[0]
training_data['Embarked']=training_data['Embarked'].fillna(freq_port)
test_data['Embarked']=test_data['Embarked'].fillna(freq_port)
```

In [20]:
```
port_mapping={"S":0,"C":1,"Q":2}
training_data['Embarked']=training_data['Embarked'].map(port_mapping).astype(int)
```

In [21]:
```
test_data['Embarked']=test_data['Embarked'].map(port_mapping).astype(int)
```

In [22]:
```
test_data['Fare'].fillna(test_data['Fare'].dropna().median(),inplace=True)
```

In [23]:
```
training_data['FareBand'] = pd.qcut(training_data['Fare'], 4)
training_data.loc[training_data['Fare']<=7.91,'Fare']=0
training_data.loc[(training_data['Fare']>7.91) & (training_data['Fare']<=14.454),'Fare']=1
training_data.loc[(training_data['Fare']>14.454) & (training_data['Fare']<=31),'Fare']=2
training_data.loc[(training_data['Fare']>31.0),'Fare']=3
training_data['Fare']=training_data['Fare'].astype(int)
test_data.loc[test_data['Fare']<=7.91,'Fare']=0
test_data.loc[(test_data['Fare']>7.91) & (test_data['Fare']<=14.454),'Fare']=1
test_data.loc[(test_data['Fare']>14.454) & (test_data['Fare']<=31),'Fare']=2
test_data.loc[(test_data['Fare']>31.0),'Fare']=3
test_data['Fare']=test_data['Fare'].astype(int)
```

In [24]:
```
training_data=training_data.drop(['FareBand'],axis=1)
```

In [25]:
```
training_data['Title']=training_data['Title'].astype(int)
test_data['Title']=test_data['Title'].astype(int)
```

In [26]:
```
training_data['Age'].fillna(training_data['Age'].dropna().median(),inplace=True)
```

In [27]:
```
training_data['Age']=round(training_data['Age'])
```

In [28]:
```
test_data['Age'].fillna(test_data['Age'].dropna().median(),inplace=True)
```

In [29]:
```
training_data.loc[ training_data['Age'] <= 16, 'Age'] = 0
training_data.loc[(training_data['Age'] > 16) & (training_data['Age'] <= 32), 'Age'] = 1
training_data.loc[(training_data['Age'] > 32) & (training_data['Age'] <= 48), 'Age'] = 2
training_data.loc[(training_data['Age'] > 48) & (training_data['Age'] <= 64), 'Age'] = 3
training_data.loc[ training_data['Age'] > 64, 'Age'] = 4

test_data.loc[ test_data['Age'] <= 16, 'Age'] = 0
test_data.loc[(test_data['Age'] > 16) & (test_data['Age'] <= 32), 'Age'] = 1
test_data.loc[(test_data['Age'] > 32) & (test_data['Age'] <= 48), 'Age'] = 2
test_data.loc[(test_data['Age'] > 48) & (test_data['Age'] <= 64), 'Age'] = 3
test_data.loc[ test_data['Age'] > 64, 'Age'] = 4
```

In [30]:
```
training_data=training_data.drop(['PassengerId'],axis=1)
```

In [31]:
```
X_train=training_data.drop(['Survived'],axis=1)
Y_train=training_data['Survived']
X_test=test_data.drop("PassengerId", axis=1).copy()
```

In [32]:
```
# Logistic Regression
logreg=LogisticRegression()
logreg.fit(X_train,Y_train)
Y_pred=logreg.predict(X_test)
acc_log=round(logreg.score(X_train,Y_train)*100,2)
acc_log
```

Out[32]: 78.900000000000006

In [33]:
```
# Supported vector machines
svc= SVC()
svc.fit(X_train,Y_train)
Y_pred=svc.predict(X_test)
acc_svc=round(svc.score(X_train,Y_train)*100,2)
acc_svc
```

Out[33]: 83.280000000000001

In [34]:
```
# K-nearest neighbor
Knn=KNeighborsClassifier(n_neighbors=3)
Knn.fit(X_train,Y_train)
Y_pred=Knn.predict(X_test)
acc_knn=round(Knn.score(X_train,Y_train)*100,2)
acc_knn
```

Out[34]: 83.840000000000003

In [35]:
```
# Gaussian Naive Bayes
gaussian= GaussianNB()
gaussian.fit(X_train,Y_train)
Y_pred=gaussian.predict(X_test)
acc_gaus=round(gaussian.score(X_train,Y_train)*100,2)
acc_gaus
```

Out[35]: 78.790000000000006

In [36]:
```
# Perceptron
perceptron=Perceptron()
perceptron.fit(X_train,Y_train)
Y_pred=perceptron.predict(X_test)
acc_percep=round(perceptron.score(X_train,Y_train)*100,2)
acc_percep
```

```
C:\Miniconda2\envs\py3k\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:128: FutureWarning: max_iter and tol parameters have be
en added in <class 'sklearn.linear_model.perceptron.Perceptron'> in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If
tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

Out[36]: 80.359999999999999

In [37]:
```
#Liner Support  Vector Machines
linear_SVC=LinearSVC()
linear_SVC.fit(X_train,Y_train)
Y_pred=linear_SVC.predict(X_test)
acc_linear_svc=round(linear_SVC.score(X_train,Y_train)*100,2)
acc_linear_svc
```

Out[37]: 80.019999999999996

In [38]:
```
#Stocastic Gradient Classifier
SGD= SGDClassifier()
SGD.fit(X_train,Y_train)
Y_pred=SGD.predict(X_test)
acc_SGD=round(SGD.score(X_train,Y_train)*100,2)
acc_SGD
```

```
C:\Miniconda2\envs\py3k\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:128: FutureWarning: max_iter and tol parameters have be
en added in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If both are left unset, they default to max_iter=5 and
tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

Out[38]: 79.459999999999994

In [39]:
```
#Decision Tree Classifier
decision_tree=DecisionTreeClassifier()
decision_tree.fit(X_train,Y_train)
Y_pred=decision_tree.predict(X_test)
acc_decision_tree=round(decision_tree.score(X_train,Y_train)*100,2)
acc_decision_tree
```

Out[39]: 86.870000000000005

In [40]:
```
#Random Forest
random_forest=RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train,Y_train)
Y_pred=random_forest.predict(X_test)
acc_random_forest=round(random_forest.score(X_train,Y_train)*100,2)
acc_random_forest
```

Out[40]: 86.870000000000005

In [41]:
```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent', 'Linear SVC',
              'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaus, acc_percep,
              acc_SGD, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

Out[41]:

	Model	Score
3	Random Forest	86.87
8	Decision Tree	86.87
1	KNN	83.84
0	Support Vector Machines	83.28
5	Perceptron	80.36
7	Linear SVC	80.02
6	Stochastic Gradient Decent	79.46
2	Logistic Regression	78.90
4	Naive Bayes	78.79

In [42]:
```
submission = pd.DataFrame({
        "PassengerId": test_data["PassengerId"],
        "Survived": Y_pred
    })
path = ("submissions.csv")

submission.to_csv(path, index=False)
```