Titanic Survival Prediction (Classification)

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

```
In [2]: df = pd.read_csv("titanic.csv")
```

```
In [3]: df.head()
```

```
Out[3]:
```

	Passengerid	Age	Fare	Sex	sibsp	zero	zero1	zero2	zero3	zero4	...	zero12	zero13	zero14	Pclass	zero15
0	1	22.0	7.2500	0	1	0	0	0	0	0	...	0	0	0	3	0
1	2	38.0	71.2833	1	1	0	0	0	0	0	...	0	0	0	1	0
2	3	26.0	7.9250	1	0	0	0	0	0	0	...	0	0	0	3	0
3	4	35.0	53.1000	1	1	0	0	0	0	0	...	0	0	0	1	0
4	5	35.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0

5 rows × 25 columns

```
In [4]: df.shape
```

```
Out[4]: (1309, 25)
```

```
In [5]: df = df.dropna()
```

```
In [6]: df = df.drop(['zero','zero1','zero2','zero3','zero4','zero5','zero6','zero7','zero8','zero9','zero10','zero11','zero12','zero13','zero14','zero15','zero16','zero17','zero18','zero19','zero20','zero21','zero22','zero23','zero24','wife'],1)
```

```
In [7]: df.describe()
```

```
Out[7]:
```

	Passengerid	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked	Survived
count	1307.000000	1307.000000	1307.000000	1307.000000	1307.000000	1307.000000	1307.000000	1307.000000	1307.000
mean	655.219816	29.477921	33.259556	0.355011	0.499617	0.385618	2.298392	1.492731	0.292
std	377.822235	12.991702	51.748758	0.478700	1.042272	0.865552	0.837642	0.814628	0.438
min	1.000000	0.170000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000
25%	328.500000	22.000000	7.895800	0.000000	0.000000	0.000000	2.000000	1.000000	0.000
50%	655.000000	28.000000	14.454200	0.000000	0.000000	0.000000	3.000000	2.000000	0.000
75%	982.500000	35.000000	31.275000	1.000000	1.000000	0.000000	3.000000	2.000000	1.000
max	1309.000000	80.000000	512.329200	1.000000	8.000000	9.000000	3.000000	2.000000	1.000

```
In [8]: df.head(11)
```

```
Out[8]:
```

	Passengerid	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked	Survived
0	1	22.0	7.2500	0	1	0	3	2.0	0
1	2	38.0	71.2833	1	1	0	1	0.0	1
2	3	26.0	7.9250	1	0	0	3	2.0	1
3	4	35.0	53.1000	1	1	0	1	2.0	1
4	5	35.0	8.0500	0	0	0	3	2.0	0
5	6	28.0	8.4583	0	0	0	3	1.0	0
6	7	54.0	51.8625	0	0	0	1	2.0	0
7	8	2.0	21.0750	0	3	1	3	2.0	0
8	9	27.0	11.1333	1	0	2	3	2.0	1
9	10	14.0	30.0708	1	1	0	2	0.0	1
10	11	4.0	16.7000	1	1	1	3	2.0	1

Embarked : 0 = Cherbourg, 1 = Queenstown, 2 = Southampton

Pclass : 1 = 1st, 2 = 2nd, 3 = 3rd

Survived : 0 = No, 1 = Yes

sibsp : no. of siblings / spouses aboard the Titanic

Parch : no. of parents / children aboard the Titanic

```
In [9]: from sklearn.preprocessing import LabelEncoder
```

```
In [10]: k = df.iloc[:,1:]
         k
```

```
Out[10]:
```

	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked
0	22.0	7.2500	0	1	0	3	2.0
1	38.0	71.2833	1	1	0	1	0.0
2	26.0	7.9250	1	0	0	3	2.0
3	35.0	53.1000	1	1	0	1	2.0
4	35.0	8.0500	0	0	0	3	2.0
5	28.0	8.4583	0	0	0	3	1.0
6	54.0	51.8625	0	0	0	1	2.0
7	2.0	21.0750	0	3	1	3	2.0
8	27.0	11.1333	1	0	2	3	2.0
9	14.0	30.0708	1	1	0	2	0.0
10	4.0	16.7000	1	1	1	3	2.0
11	58.0	26.5500	1	0	0	1	2.0
12	20.0	8.0500	0	0	0	3	2.0
13	39.0	31.2750	0	1	5	3	2.0
14	14.0	7.8542	1	0	0	3	2.0
15	55.0	16.0000	1	0	0	2	2.0
16	2.0	29.1250	0	4	1	3	1.0
17	20.0	13.0000	0	0	0	2	2.0
18	31.0	18.0000	1	1	0	3	2.0
19	26.0	7.2250	1	0	0	3	0.0
20	35.0	26.0000	0	0	0	2	2.0
21	34.0	13.0000	0	0	0	2	2.0
22	15.0	8.0292	1	0	0	3	1.0
23	28.0	35.5000	0	0	0	1	2.0
24	8.0	21.0750	1	3	1	3	2.0
25	38.0	31.3875	1	1	5	3	2.0
26	26.0	7.2250	0	0	0	3	0.0
27	19.0	263.0000	0	3	2	1	2.0
28	26.0	7.8792	1	0	0	3	1.0
29	26.0	7.8958	0	0	0	3	2.0
...	...	...	...	...	...	...	...
1279	21.0	7.7500	0	0	0	3	1.0
1280	6.0	21.0750	0	3	1	3	2.0
1281	23.0	93.5000	0	1	0	1	2.0
1282	51.0	39.4000	1	0	1	1	2.0
1283	13.0	20.2500	0	0	2	3	2.0
1284	47.0	10.5000	0	1	2	2	2.0
1285	29.0	22.0250	0	0	0	3	2.0
1286	18.0	60.0000	1	1	0	1	2.0
1287	24.0	7.2500	0	0	0	3	2.0
1288	48.0	79.2000	1	1	1	1	0.0
1289	22.0	7.7750	0	0	0	3	2.0
1290	31.0	7.7333	0	0	0	3	1.0
1291	30.0	164.8667	1	0	0	1	0.0
1292	38.0	21.0000	0	1	0	2	2.0
1293	22.0	59.4000	1	0	0	1	0.0
1294	17.0	47.1000	0	0	0	1	2.0
1295	43.0	27.7208	0	1	0	1	0.0
1296	20.0	13.8625	0	0	0	2	0.0
1297	23.0	10.5000	0	1	0	2	2.0
1298	50.0	211.5000	1	1	0	1	0.0
1299	26.0	7.7208	1	0	0	3	1.0
1300	3.0	13.7750	1	1	1	3	2.0
1301	26.0	7.7500	1	0	0	3	1.0
1302	37.0	90.0000	1	1	1	1	1.0
1303	28.0	7.7750	1	0	0	3	2.0
1304	28.0	8.0500	0	0	0	3	2.0
1305	39.0	108.9000	1	0	0	1	0.0
1306	38.5	7.2500	0	0	0	3	2.0
1307	26.0	8.0500	0	0	0	3	2.0
1308	26.0	22.3583	0	1	1	3	0.0

1307 rows × 7 columns

```
In [11]: y = df.iloc[:,-1].values
```

```
In [12]: #l = k.apply(LabelEncoder().fit_transform)
         #l
```

```
Out[12]:
```

	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked
0	22.0	7.2500	0	1	0	3	2.0
1	38.0	71.2833	1	1	0	1	0.0
2	26.0	7.9250	1	0	0	3	2.0
3	35.0	53.1000	1	1	0	1	2.0
4	35.0	8.0500	0	0	0	3	2.0
5	28.0	8.4583	0	0	0	3	1.0
6	54.0	51.8625	0	0	0	1	2.0
7	2.0	21.0750	0	3	1	3	2.0
8	27.0	11.1333	1	0	2	3	2.0
9	14.0	30.0708	1	1	0	2	0.0
10	4.0	16.7000	1	1	1	3	2.0
11	58.0	26.5500	1	0	0	1	2.0
12	20.0	8.0500	0	0	0	3	2.0
13	39.0	31.2750	0	1	5	3	2.0
14	14.0	7.8542	1	0	0	3	2.0
15	55.0	16.0000	1	0	0	2	2.0
16	2.0	29.1250	0	4	1	3	1.0
17	20.0	13.0000	0	0	0	2	2.0
18	31.0	18.0000	1	1	0	3	2.0
19	26.0	7.2250	1	0	0	3	0.0
20	35.0	26.0000	0	0	0	2	2.0
21	34.0	13.0000	0	0	0	2	2.0
22	15.0	8.0292	1	0	0	3	1.0
23	28.0	35.5000	0	0	0	1	2.0
24	8.0	21.0750	1	3	1	3	2.0
25	38.0	31.3875	1	1	5	3	2.0
26	26.0	7.2250	0	0	0	3	0.0
27	19.0	263.0000	0	3	2	1	2.0
28	26.0	7.8792	1	0	0	3	1.0
29	26.0	7.8958	0	0	0	3	2.0
...	...	...	...	...	...	...	...
1279	21.0	7.7500	0	0	0	3	1.0
1280	6.0	21.0750	0	3	1	3	2.0
1281	23.0	93.5000	0	1	0	1	2.0
1282	51.0	39.4000	1	0	1	1	2.0
1283	13.0	20.2500	0	0	2	3	2.0
1284	47.0	10.5000	0	1	2	2	2.0
1285	29.0	22.0250	0	0	0	3	2.0
1286	18.0	60.0000	1	1	0	1	2.0
1287	24.0	7.2500	0	0	0	3	2.0
1288	48.0	79.2000	1	1	1	1	0.0
1289	22.0	7.7750	0	0	0	3	2.0
1290	31.0	7.7333	0	0	0	3	1.0
1291	30.0	164.8667	1	0	0	1	0.0
1292	38.0	21.0000	0	1	0	2	2.0
1293	22.0	59.4000	1	0	0	1	0.0
1294	17.0	47.1000	0	0	0	1	2.0
1295	43.0	27.7208	0	1	0	1	0.0
1296	20.0	13.8625	0	0	0	2	0.0
1297	23.0	10.5000	0	1	0	2	2.0
1298	50.0	211.5000	1	1	0	1	0.0
1299	26.0	7.7208	1	0	0	3	1.0
1300	3.0	13.7750	1	1	1	3	2.0
1301	26.0	7.7500	1	0	0	3	1.0
1302	37.0	90.0000	1	1	1	1	1.0
1303	28.0	7.7750	1	0	0	3	2.0
1304	28.0	8.0500	0	0	0	3	2.0
1305	39.0	108.9000	1	0	0	1	0.0
1306	38.5	7.2500	0	0	0	3	2.0
1307	26.0	8.0500	0	0	0	3	2.0
1308	26.0	22.3583	0	1	1	3	0.0

1307 rows × 7 columns

```
In [13]: from sklearn.model_selection import train_test_split
```

```
In [14]: X_train, X_test, y_train = train_test_split(k, test_size=0.2, random_state=0)
```

Class 'Models' to define and save the models used

```
In [15]: class Models:

             def __init__(self,name,accuracy):
                 self.name = name
                 self.accuracy = accuracy

             def output(self):
                 return "Name of model is {self.name} and accuracy is {self.accuracy}"
```

List to save all models

```
In [16]: mod_list = []
```

Logistic Regression

```
In [17]: from sklearn.linear_model import LogisticRegression
```

```
In [18]: clf = LogisticRegression(random_state=0,max_iter=1000, solver='lbfgs',multi_class='ovr')
```

```
In [19]: clf.fit(X_train,y_train)
         #_train
```

```
Out[19]:
```

	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked
497	3.00	18.7942	0	1	1	3	2.0
44	19.00	7.8792	1	0	0	3	1.0
1072	37.00	83.1583	0	1	1	1	0.0
788	25.00	7.2500	0	0	0	3	2.0
1228	16.00	7.6500	1	0	0	3	2.0
478	10.00	24.1500	1	0	2	3	2.0
994	28.00	8.0500	0	0	0	3	2.0
940	27.00	10.5000	0	0	0	2	2.0
276	35.00	31.0500	0	0	0	1	2.0
874	18.00	7.7750	0	0	0	3	2.0
313	28.00	7.8958	0	0	0	3	2.0
648	34.00	13.0000	1	0	0	2	2.0
783	36.00	120.0000	1	1	2	1	2.0
879	62.00	10.5000	0	0	0	2	2.0
798	1.00	20.5750	0	1	2	3	2.0
196	29.00	7.7500	0	0	0	3	1.0
865	36.00	7.4958	1	0	0	3	2.0
1268	50.00	106.6333	1	0	0	1	0.0
71	16.00	46.9000	1	5	2	3	2.0
1166	0.83	9.3500	0	0	1	3	2.0
16	4.00	16.7000	1	1	1	3	2.0
900	21.00	24.1500	0	2	0	3	2.0
885	31.00	57.0000	0	1	2	1	2.0
1066	29.00	9.1125	1	0	0	3	2.0
1138	20.00	7.2292	0	0	0	3	0.0
679	36.00	512.3292	0	0	1	1	0.0
96	28.00	13.5000	0	1	0	2	2.0
479	28.00	52.0000	0	0	0	1	2.0
942	2.00	27.9000	1	3	2	3	2.0
536	45.00	26.5500	0	0	0	1	2.0
...	...	...	...	...	...	...	...
685	50.00	10.5000	0	2	0	2	2.0
768	30.00	7.2292	0	0	0	3	0.0
1256	19.00	60.0000	1	0	0	1	2.0
1253	28.00	7.8792	0	0	0	3	2.0
195	28.00	7.8958	1	0	0	3	2.0
892	9.00	15.2458	1	1	1	3	0.0
449	52.00	30.5000	0	0	0	1	2.0
786	28.00	7.7958	0	0	0	3	2.0
979	16.00	8.0500	1	0	0	3	2.0
118	70.50	7.7500	0	0	0	3	1.0
174	29.00	7.7375	0	0	0	3	1.0
73	26.00	14.4542	0	1	0	3	0.0
647	35.00	7.8958	0	0	0	3	0.0
808	39.00	14.5000	0	1	0	3	2.0
851	74.00	7.7750	0	0	0	3	2.0
179	16.00	7.6542	0	1	1	3	2.0
88	23.00	263.0000	1	3	2	1	2.0
1160	29.00	7.8292	0	0	0	3	1.0
796	49.00	13.9000	1	0	0	2	2.0
315	26.00	7.8542	1	0	0	3	2.0
901	26.00	7.8958	0	0	0	3	2.0
1096	38.00	25.7417	0	0	0	3	0.0
668	24.00	27.7208	1	2	1	2	2.0
378	7.00	29.1250	0	4	1	3	1.0
1019	42.00	26.2500	0	0	0	2	2.0
764	16.00	7.7750	0	0	0	3	2.0
837	28.00	8.0500	0	0	0	3	2.0
1218	46.00	79.2000	0	0	0	1	0.0
560	28.00	7.7500	0	0	0	3	1.0
688	25.00	41.5792	0	1	2	2	0.0

1045 rows × 7 columns

```
In [20]: lg_pred = clf.predict(X_test)
```

```
In [21]: from sklearn.metrics import accuracy_score
```

```
In [22]: acc = accuracy_score(y_test,lg_pred)
```

```
In [23]: print("Accuracy of this model is",acc*100)
         Accuracy of this model is 78.26417925243
```

```
In [24]: from sklearn.metrics import confusion_matrix
```

```
In [25]: confusion_matrix(y_test,lg_pred)
```

```
Out[25]: array([[175,  20],
                [ 37,  51]])
```

```
In [26]: mod_list.append(Models('Logistic Regression',acc*100))
```

SVC (Simple Vector Classification)

```
In [27]: from sklearn.svm import SVC
```

```
In [28]: clf = SVC(gamma='auto', kernel = 'rbf')
```

```
In [29]: clf.fit(X_train,y_train)
```

```
Out[29]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)
```

```
In [30]: svc_pred = clf.predict(X_test)
```

```
In [31]: acc = accuracy_score(y_test,svc_pred)
```

```
In [32]: print("Accuracy is",acc*100)
         Accuracy is 72.52390849694344
```

```
In [33]: confusion_matrix(y_test,svc_pred)
```

```
Out[33]: array([[195,   0],
                [ 74,  14]])
```

```
In [34]: mod_list.append(Models('SVC',acc*100))
```

Naive-Bayes Classification

```
In [35]: from sklearn.naive_bayes import GaussianNB
```

```
In [36]: clf = GaussianNB()
```

```
In [37]: clf.fit(X_train,y_train)
```

```
Out[37]: GaussianNB(priors=None, var_smoothing=1e-09)
```

```
In [38]: nb_pred = clf.predict(X_test)
```

```
In [39]: acc = accuracy_score(y_test,nb_pred)
```

```
In [40]: print("Accuracy is",acc*100)
         Accuracy is 75.95419847328245
```

```
In [41]: confusion_matrix(y_test,nb_pred)
```

```
Out[41]: array([[167,  28],
                [ 35,  53]])
```

```
In [42]: mod_list.append(Models('Naive-Bayes',acc*100))
```

Analysing performance of all models used

```
In [43]: for model in mod_list:
             print(model.output())
         Name of model is Logistic Regression and accuracy is 78.26417925243
         Name of model is SVC and accuracy is 72.52390849694344
         Name of model is Naive-Bayes and accuracy is 75.95419847328245
```