

For this assignment I've chosen standard Titanic dataset again. This dataset contains a flag 'survived' and some meaningful features like 'Sex', 'Age' etc. Firstly, we should import all needed dependencies.

```
In [54]: import pandas as pd
import numpy as np
import itertools

import sklearn as sk
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

import matplotlib.pyplot as plt
%matplotlib inline
```

Loading and data transformation:

```
In [55]: data = pd.read_csv('titanic.csv')
data.drop(data.columns[0],axis=1,inplace=True)
replace_sex={'male':1,'female':0}
data.replace(replace_sex,inplace=True)
data.head(n=5)
```

```
Out[55]:
```

	PassengerId	Survived	Pclass	\
0	1	2	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	2	3	

	Name	Sex	Age	SibSp	Parch	\
0	Braund, Mr. Owen Harris	1	22.0	1	0	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	
2	Heikkinen, Miss. Laina	0	26.0	0	0	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	
4	Allen, Mr. William Henry	1	35.0	0	0	

	Ticket	Fare	Cabin	Embarked
0	A/5 21171	7.2500	NaN	S
1	PC 17599	71.2833	C85	C
2	STON/O2. 3101282	7.9250	NaN	S
3	113803	53.1000	C123	S
4	373450	8.0500	NaN	S

```
In [56]: data.drop(data.columns[0],axis=1,inplace=True)
```

```
In [57]: data.dtypes
```

```
Out[57]: Survived      int64
Pclass      int64
Name        object
Sex         int64
Age         float64
SibSp       int64
Parch       int64
Ticket      object
Fare        float64
Cabin       object
Embarked    object
dtype: object
```

```
In [58]: data.describe()
```

```
Out[58]:
```

	Survived	Pclass	Sex	Age	SibSp	Parch	\
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	
mean	1.616162	2.308642	0.647587	29.758889	0.523008	0.381594	
std	0.486592	0.836071	0.477990	13.002570	1.102743	0.806057	
min	1.000000	1.000000	0.000000	0.420000	0.000000	0.000000	
25%	1.000000	2.000000	0.000000	22.000000	0.000000	0.000000	
50%	2.000000	3.000000	1.000000	30.000000	0.000000	0.000000	
75%	2.000000	3.000000	1.000000	35.000000	1.000000	0.000000	
max	2.000000	3.000000	1.000000	80.000000	8.000000	6.000000	

	Fare
count	891.000000
mean	32.204208
std	49.693429
min	0.000000
25%	7.910400
50%	14.454200
75%	31.000000
max	512.329200

We only need these features:

```
In [59]: meaningful_columns=['Survived','Pclass','Sex','Age','SibSp','Parch','Fare']
data=data[meaningful_columns]
data.head(n=5)
```

```
Out[59]:
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	2	3	1	22.0	1	0	7.2500
1	1	1	0	38.0	1	0	71.2833
2	1	3	0	26.0	0	0	7.9250
3	1	1	0	35.0	1	0	53.1000
4	2	3	1	35.0	0	0	8.0500

Replacing missing values in each feature by its mean.

```
In [60]: data.fillna(np.round(data.mean()),inplace=True)
```

```
Out[60]:
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	2	3	1	22.0	1	0	7.2500
1	1	1	0	38.0	1	0	71.2833
2	1	3	0	26.0	0	0	7.9250
3	1	1	0	35.0	1	0	53.1000
4	2	3	1	35.0	0	0	8.0500
5	2	3	1	30.0	0	0	8.4583
6	2	1	1	54.0	0	0	51.8625
7	2	3	1	2.0	3	1	21.0750
8	1	3	0	27.0	0	2	11.1333
9	1	2	0	14.0	1	0	30.0708
10	1	3	0	4.0	1	1	16.7000
11	1	1	0	58.0	0	0	26.5500
12	2	3	1	20.0	0	0	8.0500
13	2	3	1	39.0	1	5	31.2750
14	2	3	0	14.0	0	0	7.8542
15	1	2	0	55.0	0	0	16.0000
16	2	3	1	2.0	4	1	29.1250
17	1	2	1	30.0	0	0	13.0000
18	2	3	0	31.0	1	0	18.0000
19	1	3	0	30.0	0	0	7.2250
20	2	2	1	35.0	0	0	26.0000
21	1	2	1	34.0	0	0	13.0000
22	1	3	0	15.0	0	0	8.0292
23	1	1	1	28.0	0	0	35.5000
24	2	3	0	8.0	3	1	21.0750
25	1	3	0	38.0	1	5	31.3875
26	2	3	1	30.0	0	0	7.2250
27	2	1	1	19.0	3	2	263.0000
28	1	3	0	30.0	0	0	7.8792
29	2	3	1	30.0	0	0	7.8958
...
861	2	2	1	21.0	1	0	11.5000
862	1	1	0	48.0	0	0	25.9292
863	2	3	0	30.0	8	2	69.5500
864	2	2	1	24.0	0	0	13.0000
865	1	2	0	42.0	0	0	13.0000
866	1	2	0	27.0	1	0	13.8583
867	2	1	1	31.0	0	0	50.4958
868	2	3	1	30.0	0	0	9.5000
869	1	3	1	4.0	1	1	11.1333
870	2	3	1	26.0	0	0	7.8958
871	1	1	0	47.0	1	1	52.5542
872	2	1	1	33.0	0	0	5.0000

873	2	3	1	47.0	0	0	9.0000
874	1	2	0	28.0	1	0	24.0000
875	1	3	0	15.0	0	0	7.2250
876	2	3	1	20.0	0	0	9.8458
877	2	3	1	19.0	0	0	7.8958
878	2	3	1	30.0	0	0	7.8958
879	1	1	0	56.0	0	1	83.1583
880	1	2	0	25.0	0	1	26.0000
881	2	3	1	33.0	0	0	7.8958
882	2	3	0	22.0	0	0	10.5167
883	2	2	1	28.0	0	0	10.5000
884	2	3	1	25.0	0	0	7.0500
885	2	3	0	39.0	0	5	29.1250
886	2	2	1	27.0	0	0	13.0000
887	1	1	0	19.0	0	0	30.0000
888	2	3	0	30.0	1	2	23.4500
889	1	1	1	26.0	0	0	30.0000
890	2	3	1	32.0	0	0	7.7500

[891 rows x 7 columns]

Splitting all dataset into train and test parts as relation 3 to 1.

```
In [61]: X_train, X_test, y_train, y_test = train_test_split(data[meaningful_columns[1:]], data[
```

Basic random forest:

```
In [62]: clf = RandomForestClassifier(random_state=123456,n_jobs=-1)
        clf.fit(X_train, y_train)
```

```
Out[62]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
                                oob_score=False, random_state=123456, verbose=0,
                                warm_start=False)
```

Accuracy testing:

```
In [63]: accuracy_score(clf.predict(X_test),y_test)
```

```
Out[63]: 0.7847533632286996
```

Cross validation:

```
In [64]: cross_val_score(clf,X_train,y_train,cv=5,n_jobs=-1)
```

```
Out[64]: array([ 0.80597015,  0.81343284,  0.75373134,  0.84962406,  0.79699248])
```

Display the relative importance of each attribute:

```
In [65]: model = ExtraTreesClassifier()
         model.fit(X_train,y_train)
         print(model.feature_importances_)

[ 0.09868242  0.3001502   0.2540328   0.03841794  0.04034001  0.26837663]
```

For test different values of some parameters we can use GridSearch. For instance, testing of trees number:

```
In [66]: parameters = {'n_estimators':np.arange(10,100,1)}

         GSclf = GridSearchCV(clf, parameters,n_jobs=-1)
         GSclf.fit(X_train,y_train)

Out[66]: GridSearchCV(cv=None, error_score='raise',
                      estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
                      oob_score=False, random_state=123456, verbose=0,
                      warm_start=False),
                      fit_params=None, iid=True, n_jobs=-1,
                      param_grid={'n_estimators': array([10, 11, ..., 98, 99])},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                      scoring=None, verbose=0)
```

We can see that best forest has 17 trees.

```
In [67]: GSclf.best_params_
```

```
Out[67]: {'n_estimators': 17}
```

```
In [68]: GSclf.cv_results_
```

```
Out[68]: {'mean_fit_time': array([ 0.12308081,  0.12903086,  0.13330849,  0.13385685,  0.1347660
                                0.13430985,  0.13018155,  0.13335721,  0.13624589,  0.15376886,
                                0.14019529,  0.15094407,  0.12964137,  0.13111917,  0.13204455,
                                0.13640475,  0.13629874,  0.13807821,  0.13830463,  0.1404171 ,
                                0.143641  ,  0.14201442,  0.14603957,  0.14578509,  0.15163294,
                                0.15384467,  0.16323225,  0.1501259 ,  0.15610862,  0.20380902,
                                0.18014884,  0.16796494,  0.21180749,  0.1630768 ,  0.16681027,
                                0.18191822,  0.20401541,  0.19496433,  0.22481537,  0.20913943,
                                0.19715023,  0.22381433,  0.25217303,  0.29336079,  0.27918498,
                                0.21033303,  0.33955812,  0.34822925,  0.19463325,  0.19965196,
                                0.37207993,  0.31520931,  0.25400225,  0.29019245,  0.22631605,
```

```

0.21047298, 0.20639523, 0.22231332, 0.41093802, 0.36991239,
0.45646946, 0.31120626, 0.45613567, 0.41494083, 0.32571594,
0.52918339, 0.32938488, 0.56741039, 0.43468142, 0.44496155,
0.43628931, 0.51984469, 0.54152544, 0.52384679, 0.53718781,
0.44362744, 0.45981002, 0.40945927, 0.27184637, 0.32054575,
0.34139212, 0.3669095 , 0.47081184, 0.34339412, 0.33855772,
0.34406018, 0.44329381, 0.52051202, 0.61474069, 0.59658368]),
'mean_score_time': array([ 0.10490274, 0.10501027, 0.1031402 , 0.10389892, 0.10307
0.10338521, 0.10351825, 0.10333784, 0.10522008, 0.10428794,
0.10566807, 0.10522532, 0.1038696 , 0.10372202, 0.10335128,
0.10364453, 0.10357841, 0.10340373, 0.10364874, 0.10347605,
0.10343655, 0.10342288, 0.1037848 , 0.103978 , 0.10338434,
0.10671624, 0.10344529, 0.10376366, 0.12158672, 0.11346563,
0.10376263, 0.10528104, 0.11207938, 0.10381087, 0.10426927,
0.10437862, 0.10423517, 0.12675524, 0.10390218, 0.10607203,
0.10442074, 0.10791047, 0.12608711, 0.114242 , 0.10643888,
0.10590355, 0.12675063, 0.10456681, 0.10442225, 0.10506908,
0.10590585, 0.11524653, 0.12108056, 0.11340888, 0.10423541,
0.10406899, 0.10415864, 0.10640446, 0.1104064 , 0.11190629,
0.11707886, 0.1082379 , 0.13208739, 0.10490362, 0.15293495,
0.1145761 , 0.13325699, 0.20029982, 0.14209445, 0.12808466,
0.12124753, 0.13008626, 0.10657072, 0.1240836 , 0.11140784,
0.1145761 , 0.10657144, 0.10640351, 0.10473593, 0.10523566,
0.10473649, 0.1075716 , 0.10506956, 0.10490322, 0.10573681,
0.10723797, 0.11657739, 0.15343523, 0.13025316, 0.10690045]),
'mean_test_score': array([ 0.8008982 , 0.81137725, 0.80838323, 0.81736527, 0.81886
0.81437126, 0.81736527, 0.82185629, 0.81736527, 0.81586826,
0.81287425, 0.81437126, 0.81137725, 0.81886228, 0.81137725,
0.81287425, 0.80988024, 0.81437126, 0.81137725, 0.81287425,
0.80988024, 0.80838323, 0.80688623, 0.80389222, 0.80538922,
0.80239521, 0.80838323, 0.80988024, 0.80838323, 0.80838323,
0.80688623, 0.80538922, 0.80838323, 0.80838323, 0.80538922,
0.80538922, 0.80838323, 0.80988024, 0.80838323, 0.80988024,
0.80838323, 0.80988024, 0.81137725, 0.81137725, 0.80838323,
0.80688623, 0.80239521, 0.80389222, 0.80688623, 0.7994012 ,
0.80688623, 0.80389222, 0.80538922, 0.80389222, 0.80239521,
0.80389222, 0.80239521, 0.80239521, 0.80389222, 0.8008982 ,
0.80538922, 0.8008982 , 0.80239521, 0.80239521, 0.80239521,
0.80239521, 0.80389222, 0.80538922, 0.80389222, 0.80239521,
0.80389222, 0.7994012 , 0.80239521, 0.80239521, 0.80389222,
0.8008982 , 0.80389222, 0.8008982 , 0.79790419, 0.79491018,
0.7994012 , 0.7994012 , 0.7994012 , 0.80239521, 0.80239521,
0.80239521, 0.79790419, 0.8008982 , 0.80239521, 0.80239521]),
'mean_train_score': array([ 0.9730605 , 0.97381462, 0.97829893, 0.97381462, 0.9760
0.97680079, 0.97680416, 0.9797937 , 0.98054108, 0.98054108,
0.98203585, 0.98278323, 0.98353062, 0.98278323, 0.98428137,
0.98428137, 0.98428137, 0.9827866 , 0.98353398, 0.98428137,
0.98428137, 0.98428137, 0.98428137, 0.98428137, 0.98428137,

```

```

0.98502875, 0.98577613, 0.98502875, 0.98428137, 0.98353062,
0.98502875, 0.98502875, 0.98502875, 0.98428137, 0.98428137,
0.98502875, 0.98502875, 0.98502875, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613,
0.98577613, 0.98577613, 0.98577613, 0.98577613, 0.98577613]),
'param_n_estimators': masked_array(data = [10 11 12 13 14 15 16 17 18 19 20 21 22 23 2
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99],
mask = [False False False False False False False False False False False False
False False False False False False False False False False False False False
False False False False False False False False False False False False False
False False False False False False False False False False False False False
False False False False False False False False False False False False False
False False False False False False],
fill_value = ?),
'params': [{'n_estimators': 10},
{'n_estimators': 11},
{'n_estimators': 12},
{'n_estimators': 13},
{'n_estimators': 14},
{'n_estimators': 15},
{'n_estimators': 16},
{'n_estimators': 17},
{'n_estimators': 18},
{'n_estimators': 19},
{'n_estimators': 20},
{'n_estimators': 21},
{'n_estimators': 22},
{'n_estimators': 23},
{'n_estimators': 24},
{'n_estimators': 25},
{'n_estimators': 26},
{'n_estimators': 27},
{'n_estimators': 28},
{'n_estimators': 29},
{'n_estimators': 30},
{'n_estimators': 31}],

```

```
{'n_estimators': 32},  
{'n_estimators': 33},  
{'n_estimators': 34},  
{'n_estimators': 35},  
{'n_estimators': 36},  
{'n_estimators': 37},  
{'n_estimators': 38},  
{'n_estimators': 39},  
{'n_estimators': 40},  
{'n_estimators': 41},  
{'n_estimators': 42},  
{'n_estimators': 43},  
{'n_estimators': 44},  
{'n_estimators': 45},  
{'n_estimators': 46},  
{'n_estimators': 47},  
{'n_estimators': 48},  
{'n_estimators': 49},  
{'n_estimators': 50},  
{'n_estimators': 51},  
{'n_estimators': 52},  
{'n_estimators': 53},  
{'n_estimators': 54},  
{'n_estimators': 55},  
{'n_estimators': 56},  
{'n_estimators': 57},  
{'n_estimators': 58},  
{'n_estimators': 59},  
{'n_estimators': 60},  
{'n_estimators': 61},  
{'n_estimators': 62},  
{'n_estimators': 63},  
{'n_estimators': 64},  
{'n_estimators': 65},  
{'n_estimators': 66},  
{'n_estimators': 67},  
{'n_estimators': 68},  
{'n_estimators': 69},  
{'n_estimators': 70},  
{'n_estimators': 71},  
{'n_estimators': 72},  
{'n_estimators': 73},  
{'n_estimators': 74},  
{'n_estimators': 75},  
{'n_estimators': 76},  
{'n_estimators': 77},  
{'n_estimators': 78},  
{'n_estimators': 79},
```

```

{'n_estimators': 80},
{'n_estimators': 81},
{'n_estimators': 82},
{'n_estimators': 83},
{'n_estimators': 84},
{'n_estimators': 85},
{'n_estimators': 86},
{'n_estimators': 87},
{'n_estimators': 88},
{'n_estimators': 89},
{'n_estimators': 90},
{'n_estimators': 91},
{'n_estimators': 92},
{'n_estimators': 93},
{'n_estimators': 94},
{'n_estimators': 95},
{'n_estimators': 96},
{'n_estimators': 97},
{'n_estimators': 98},
{'n_estimators': 99}],
'rank_test_score': array([77, 14, 26, 4, 2, 8, 4, 1, 4, 7, 11, 8, 14, 2, 14,
      8, 14, 11, 20, 26, 37, 49, 42, 60, 26, 20, 26, 26, 37, 42, 26, 26,
      42, 42, 26, 20, 26, 20, 26, 20, 14, 14, 26, 37, 60, 49, 37, 83, 37,
      49, 42, 49, 60, 49, 60, 60, 49, 77, 42, 77, 60, 60, 60, 60, 49, 42,
      49, 60, 49, 83, 60, 60, 49, 77, 49, 77, 88, 90, 83, 83, 83, 60, 60,
      60, 88, 77, 60, 60]),
'split0_test_score': array([ 0.8125      ,  0.83482143,  0.83035714,  0.83928571,  0.848
      0.84375      ,  0.84821429,  0.84821429,  0.84821429,  0.83928571,
      0.82589286,  0.82589286,  0.82142857,  0.83035714,  0.81696429,
      0.82142857,  0.82142857,  0.83482143,  0.82589286,  0.83482143,
      0.83035714,  0.83035714,  0.82589286,  0.83035714,  0.82589286,
      0.82142857,  0.82142857,  0.82142857,  0.82589286,  0.83035714,
      0.82142857,  0.82589286,  0.83035714,  0.83482143,  0.82142857,
      0.83482143,  0.82589286,  0.83035714,  0.82142857,  0.83035714,
      0.82589286,  0.82589286,  0.82589286,  0.83035714,  0.82589286,
      0.83482143,  0.82142857,  0.82589286,  0.81696429,  0.8125      ,
      0.81696429,  0.81696429,  0.81696429,  0.81696429,  0.81696429,
      0.81696429,  0.81696429,  0.81696429,  0.81696429,  0.8125      ,
      0.81696429,  0.8125      ,  0.8125      ,  0.81696429,  0.8125      ,
      0.8125      ,  0.8125      ,  0.8125      ,  0.8125      ,  0.8125      ,
      0.8125      ,  0.8125      ,  0.8125      ,  0.81696429,  0.81696429,
      0.81696429,  0.8125      ,  0.8125      ,  0.8125      ,  0.8125      ,
      0.8125      ,  0.8125      ,  0.81696429,  0.81696429,  0.81696429,
      0.8125      ,  0.8125      ,  0.81696429,  0.81696429,  0.81696429]),
'split0_train_score': array([ 0.97747748,  0.98198198,  0.98198198,  0.98198198,  0.97
      0.97972973,  0.98198198,  0.98198198,  0.98198198,  0.98198198,
      0.98198198,  0.98198198,  0.98198198,  0.98198198,  0.98423423,
      0.98423423,  0.98423423,  0.98423423,  0.98423423,

```

[illegible]

```

0.99327354, 0.99327354, 0.99327354, 0.99327354, 0.99327354,
0.99327354, 0.99327354, 0.99327354, 0.99327354, 0.99327354]),
'split2_test_score': array([ 0.83333333, 0.82882883, 0.83333333, 0.83333333, 0.828
0.82432432, 0.81981982, 0.82882883, 0.82432432, 0.83333333,
0.82882883, 0.82882883, 0.82432432, 0.83783784, 0.82432432,
0.81981982, 0.81981982, 0.81981982, 0.81981982, 0.81981982,
0.81531532, 0.81531532, 0.81531532, 0.81081081, 0.81531532,
0.81081081, 0.81981982, 0.81981982, 0.81081081, 0.81081081,
0.81081081, 0.7972973 , 0.80630631, 0.80630631, 0.81081081,
0.80630631, 0.81081081, 0.81531532, 0.81081081, 0.81081081,
0.81081081, 0.81081081, 0.81531532, 0.81081081, 0.81531532,
0.80630631, 0.80630631, 0.80630631, 0.81531532, 0.80630631,
0.81531532, 0.80630631, 0.81081081, 0.80630631, 0.8018018 ,
0.80630631, 0.8018018 , 0.8018018 , 0.80630631, 0.8018018 ,
0.81081081, 0.80630631, 0.80630631, 0.80630631, 0.81081081,
0.81081081, 0.81531532, 0.81531532, 0.81531532, 0.81531532,
0.81531532, 0.8018018 , 0.81081081, 0.81081081, 0.81531532,
0.81081081, 0.81981982, 0.81081081, 0.8018018 , 0.7972973 ,
0.8018018 , 0.80630631, 0.8018018 , 0.80630631, 0.81081081,
0.81081081, 0.80630631, 0.80630631, 0.81081081, 0.81081081]),
'split2_train_score': array([ 0.96860987, 0.96412556, 0.97309417, 0.97085202, 0.97
0.97309417, 0.97309417, 0.97757848, 0.97757848, 0.97757848,
0.97757848, 0.97982063, 0.97982063, 0.97757848, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97757848, 0.97757848,
0.97757848, 0.97757848, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063,
0.97982063, 0.97982063, 0.97982063, 0.97982063, 0.97982063]),
'std_fit_time': array([ 0.0031899 , 0.00143328, 0.00335883, 0.00281048, 0.00450705
0.00562753, 0.00621771, 0.00318506, 0.00060216, 0.00838451,
0.00992437, 0.01759739, 0.00111607, 0.00061524, 0.0007997 ,
0.00291533, 0.00251316, 0.0007932 , 0.00159033, 0.00201541,
0.00449069, 0.00064715, 0.00348832, 0.00159714, 0.00066098,
0.00802656, 0.0196615 , 0.00078377, 0.00717901, 0.03253608,
0.0339617 , 0.00608945, 0.00618461, 0.00563357, 0.00314339,
0.00269857, 0.02543743, 0.0146271 , 0.02616575, 0.04862298,
0.011428 , 0.02023864, 0.04479119, 0.01302904, 0.01365251,
0.00663208, 0.0874557 , 0.06325988, 0.01182278, 0.01134337,

```

```

0.02879717, 0.00991411, 0.01412865, 0.0793998 , 0.00881411,
0.02085583, 0.01510697, 0.00581033, 0.03480131, 0.0448151 ,
0.05752053, 0.08430024, 0.07203134, 0.09882249, 0.07126995,
0.08604252, 0.0515682 , 0.03428328, 0.04092037, 0.02146334,
0.0461248 , 0.09239292, 0.06663801, 0.02249907, 0.01800315,
0.03572891, 0.03013082, 0.05423436, 0.03493327, 0.02157538,
0.05684171, 0.08090003, 0.0667237 , 0.04517683, 0.0475567 ,
0.00873584, 0.0714337 , 0.02999837, 0.03618633, 0.02988068]),
'std_score_time': array([ 8.50850607e-04, 1.34869915e-06, 2.32468942e-04,
4.05786102e-04, 1.41506331e-05, 2.32100456e-04,
1.28707615e-04, 3.51386892e-05, 1.06595616e-03,
1.97449964e-04, 7.60887810e-04, 4.18285721e-04,
3.10172054e-04, 2.03925283e-04, 9.82157870e-05,
1.25575820e-04, 1.94908837e-04, 1.45310772e-04,
3.00399925e-04, 4.96248709e-04, 4.84002491e-04,
2.11427951e-04, 3.88016306e-04, 7.87974458e-04,
5.90728732e-04, 1.31900499e-03, 4.43950631e-04,
2.19092391e-04, 1.78655522e-02, 6.20421092e-03,
7.09268332e-04, 1.62169835e-03, 1.13319521e-02,
2.48907271e-04, 1.29392688e-03, 1.12504252e-03,
9.43415137e-04, 2.99751867e-02, 4.71145933e-04,
1.08431556e-03, 6.31640560e-04, 1.70296448e-03,
2.66256097e-02, 6.51484376e-03, 8.26883136e-04,
1.54679372e-03, 2.96006602e-02, 1.41765140e-03,
4.57930052e-04, 2.12290866e-03, 6.26147724e-04,
1.08247260e-02, 4.24609833e-03, 5.04272154e-03,
2.36360809e-04, 1.41478542e-03, 9.54573704e-04,
3.32729015e-03, 5.56120675e-03, 2.25094559e-03,
9.28072793e-03, 3.40131316e-03, 1.73945632e-02,
8.51536609e-04, 5.74927284e-02, 7.15794129e-03,
2.53356764e-02, 4.14202812e-02, 3.99824450e-02,
1.29893008e-02, 2.07920067e-02, 1.84692927e-02,
1.08140635e-03, 1.04231357e-02, 5.57624289e-03,
1.27433302e-02, 1.08140635e-03, 1.92990092e-03,
6.23989183e-04, 9.42684637e-04, 1.31267355e-03,
2.16181940e-03, 1.12391596e-07, 2.35517575e-04,
1.02743132e-03, 1.65112891e-03, 8.04696484e-03,
1.85650581e-02, 1.08570773e-02, 2.25301461e-03]),
'std_test_score': array([ 0.03228482, 0.02910493, 0.0332677 , 0.02698023, 0.029027
0.02904154, 0.02637996, 0.02497206, 0.02858813, 0.02909407,
0.02055892, 0.0184411 , 0.01633261, 0.02178598, 0.01345207,
0.01100948, 0.01524772, 0.01939584, 0.01647758, 0.02141947,
0.01941007, 0.02143329, 0.01995089, 0.02502905, 0.02202716,
0.01996374, 0.01736783, 0.01524772, 0.0154585 , 0.01910446,
0.01381967, 0.01467872, 0.01724957, 0.02090283, 0.01584828,
0.02454057, 0.0154585 , 0.01941007, 0.01182384, 0.01719975,
0.0154585 , 0.01353895, 0.01380549, 0.0153521 , 0.01788532,
0.02269541, 0.01743976, 0.01911997, 0.01313849, 0.01442016,

```

```

0.01313849, 0.01183872, 0.01232418, 0.01183872, 0.01172305,
0.01183872, 0.01172305, 0.01172305, 0.01183872, 0.00991221,
0.01232418, 0.01233686, 0.01026935, 0.01383387, 0.01314885,
0.01314885, 0.01423344, 0.01211974, 0.01423344, 0.01634925,
0.01423344, 0.01185362, 0.01314885, 0.01650137, 0.01737797,
0.01860126, 0.01762053, 0.01526818, 0.01384808, 0.01550437,
0.01185362, 0.01442016, 0.01548907, 0.01383387, 0.01650137,
0.01314885, 0.01651328, 0.01586194, 0.01650137, 0.01650137]],
'std_train_score': array([ 0.00362027, 0.00736884, 0.00378462, 0.00584729, 0.00275
0.00276421, 0.00377396, 0.00179782, 0.00209514, 0.00209514,
0.00366162, 0.0028039 , 0.00382167, 0.00461171, 0.00366157,
0.00366157, 0.00366157, 0.00209746, 0.00279035, 0.00366157,
0.00366157, 0.00366157, 0.00366157, 0.00366157, 0.00366157,
0.00461113, 0.0055993 , 0.00461113, 0.00366157, 0.00382167,
0.00461113, 0.00461113, 0.00461113, 0.00549223, 0.00549223,
0.00643207, 0.00643207, 0.00643207, 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ,
0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 , 0.0055993 ])}

```

Comparison of train and test score for different number of trees in the forest. The train accuracy tend to 1.000 while test accuracy slightly decreases.

```

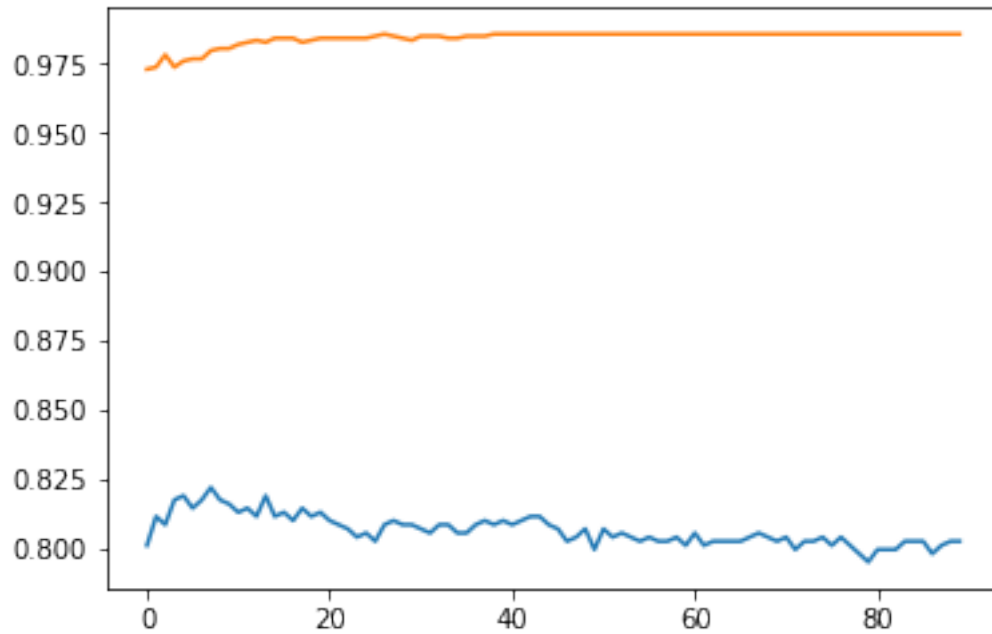
In [69]: plt.plot(GScf.cv_results_['mean_test_score'])
plt.plot(GScf.cv_results_['mean_train_score'])

```

```

Out[69]: [<matplotlib.lines.Line2D at 0x200d9714b00>]

```



As we can see, the model suggests that Sex, Age and Fare are the most important features.

```
In [70]: list(zip(meaningful_columns[1:],clf.feature_importances_))
```

```
Out[70]: [('Pclass', 0.085935985318704333),  
          ('Sex', 0.30223224952496092),  
          ('Age', 0.27601798303347069),  
          ('SibSp', 0.048533457918938885),  
          ('Parch', 0.032491203395172565),  
          ('Fare', 0.25478912080875254)]
```

```
In [71]: accuracy_score(GScf.best_estimator_.predict(X_test),y_test)
```

```
Out[71]: 0.78923766816143492
```