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

```
In [1]: import pandas as pd

        import numpy as np
        import graphviz
        import itertools
        import pydotplus
        import pydot

        import sklearn as sk
        from sklearn.tree import export_graphviz
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import confusion_matrix

        from IPython.display import Image
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Loading and data transformation:

```
In [2]: 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[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	2	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0				
4	5	2	3	Allen, Mr. William Henry	1	35.0	0	0				

```
3            113803  53.1000  C123        S
4            373450   8.0500   NaN        S
```

```
In [3]: data.dtypes
```

```
Out[3]: PassengerId      int64
        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 [4]: data.describe()
```

```
Out[4]:        PassengerId    Survived      Pclass         Sex         Age  \
        count   891.000000  891.000000  891.000000  891.000000  891.000000
        mean    446.000000    1.616162    2.308642    0.647587   29.758889
        std     257.353842    0.486592    0.836071    0.477990   13.002570
        min       1.000000    1.000000    1.000000    0.000000    0.420000
        25%     223.500000    1.000000    2.000000    0.000000   22.000000
        50%     446.000000    2.000000    3.000000    1.000000   30.000000
        75%     668.500000    2.000000    3.000000    1.000000   35.000000
        max     891.000000    2.000000    3.000000    1.000000   80.000000

                    SibSp       Parch        Fare
        count  891.000000  891.000000  891.000000
        mean     0.523008    0.381594   32.204208
        std      1.102743    0.806057   49.693429
        min      0.000000    0.000000    0.000000
        25%      0.000000    0.000000    7.910400
        50%      0.000000    0.000000   14.454200
        75%      1.000000    0.000000   31.000000
        max      8.000000    6.000000  512.329200
```

We only need these features:

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

```
Out[5]:    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 [6]: data.fillna(np.round(data.mean()),inplace=True)
```

Out[6]:

	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 [7]: X_train, X_test, y_train, y_test = train_test_split(data[meaningful_columns[1:]], data[m
```

Tree building:

```
In [8]: clf = DecisionTreeClassifier(max_leaf_nodes = 10)
        clf.fit(X_train, y_train)
```

```
Out[8]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                    max_features=None, max_leaf_nodes=10,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                    splitter='best')
```

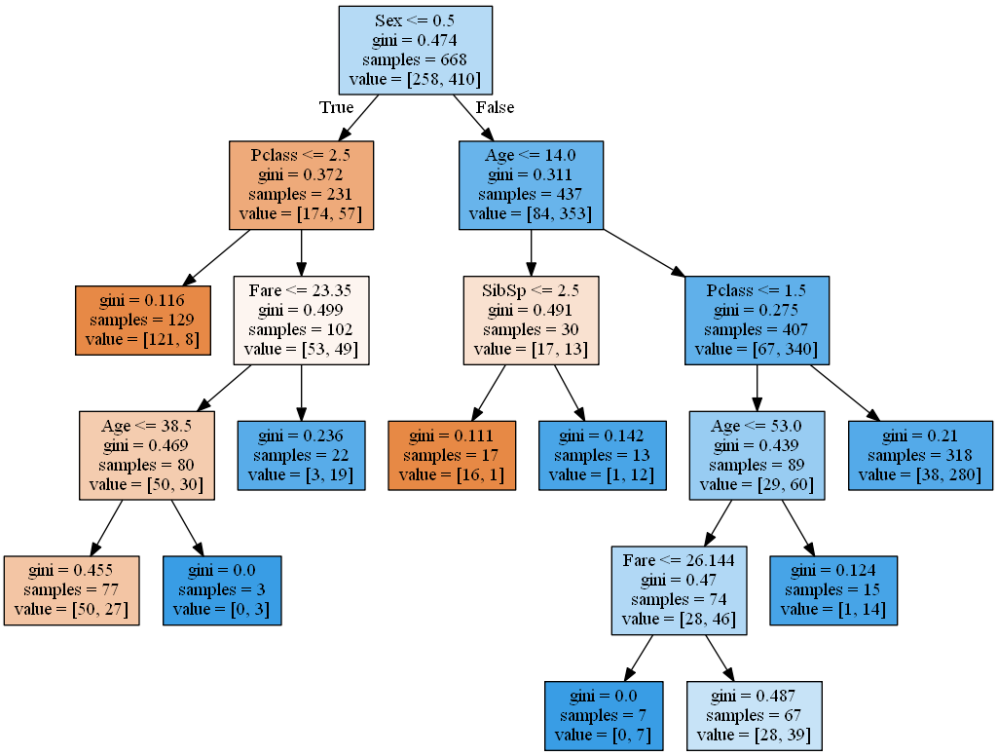
Test for our model:

```
In [9]: accuracy_score(clf.predict(X_test),y_test)
```

```
Out[9]: 0.82511210762331844
```

Plotting our tree:

```
In [10]: export_graphviz(clf, feature_names=data[meaningful_columns[1:]].columns, out_file='smal
```



According to tree above the most important predictor is sex of passenger. For female the second predictor is class and for male it is age.

Confusion matrix:

```
In [11]: cnf_matrix = confusion_matrix(y_test,clf.predict(X_test))
         print(cnf_matrix)

[[ 62  22]
 [ 17 122]]
```

We can see that total number of uncorrected classification is 23+19 = 42, while 65+116 = 181 observations are right.