In [4]:

```
import sklearn
print (sklearn.__version__)
```

```
0.21.3
```

In [6]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.externals import joblib
```

```
C:\Users\Rodrigo\Anaconda3\lib\site-packages\sklearn\externals\joblib\__in
it__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in
0.21 and will be removed in 0.23. Please import this functionality directl
y from joblib, which can be installed with: pip install joblib. If this wa
rning is raised when loading pickled models, you may need to re-serialize
those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

In [21]:

```
dataset_url = 'file:///C:/Users/Rodrigo/Downloads/wineData.data'
data = pd.read_csv(dataset_url)
```

In [22]:

```
print (data.head())
```

```
   Alcohol  Malic acid   Ash  Alcalinity of ash  Magnesium  Total phenols
\
1    14.23        1.71  2.43               15.6        127           2.80
1    13.20        1.78  2.14               11.2        100           2.65
1    13.16        2.36  2.67               18.6        101           2.80
1    14.37        1.95  2.50               16.8        113           3.85
1    13.24        2.59  2.87               21.0        118           2.80

   Flavanoids  Nonflavanoid phenols  Proanthocyanins  Color intensity   Hu
e  \
1        3.06                  0.28             2.29             5.64  1.0
4
1        2.76                  0.26             1.28             4.38  1.0
5
1        3.24                  0.30             2.81             5.68  1.0
3
1        3.49                  0.24             2.18             7.80  0.8
6
1        2.69                  0.39             1.82             4.32  1.0
4

   OD280/OD315 of diluted wines  Proline
1                          3.92        1065
1                          3.40        1050
1                          3.17        1185
1                          3.45        1480
1                          2.93         735
```

In [17]:

```
print (data.shape)
```

```
(177, 14)
```

In [23]:

```
print (data.describe())
```

```
          Alcohol  Malic acid         Ash  Alcalinity of ash   Magnesium
\
count  178.000000  178.000000  178.000000         178.000000  178.000000
mean    13.000618    2.336348    2.366517          19.494944   99.741573
std      0.811827    1.117146    0.274344           3.339564   14.282484
min     11.030000    0.740000    1.360000          10.600000   70.000000
25%     12.362500    1.602500    2.210000          17.200000   88.000000
50%     13.050000    1.865000    2.360000          19.500000   98.000000
75%     13.677500    3.082500    2.557500          21.500000  107.000000
max     14.830000    5.800000    3.230000          30.000000  162.000000

       Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000
mean        2.295112    2.029270              0.361854         1.590899
std         0.625851    0.998859              0.124453         0.572359
min         0.980000    0.340000              0.130000         0.410000
25%         1.742500    1.205000              0.270000         1.250000
50%         2.355000    2.135000              0.340000         1.555000
75%         2.800000    2.875000              0.437500         1.950000
max         3.880000    5.080000              0.660000         3.580000

       Color intensity         Hue  OD280/OD315 of diluted wines  Proline
count       178.000000  178.000000                    178.000000   178.000
000
mean          5.058090    0.957449                      2.611685   746.893
258
std           2.318286    0.228572                      0.709990   314.907
474
min           1.280000    0.480000                      1.270000   278.000
000
25%           3.220000    0.782500                      1.937500   500.500
000
50%           4.690000    0.965000                      2.780000   673.500
000
75%           6.200000    1.120000                      3.170000   985.000
000
max          13.000000    1.710000                      4.000000  1680.000
000
```

In [42]:

```
y = data.Hue
X = data.drop('Hue', axis = 1)
```

In [44]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                    test_size= 0.2,
                                                    random_state = 123)
```

In [49]:

```
X_trained_scaled = preprocessing.scale(X_train)
print(X_trained_scaled)
```

```
[[ 0.57139079  0.85626511  1.31794859 ... -0.27123614 -0.8839588
  -0.7217344 ]
 [ 2.14699283 -0.55993585 -0.62856777 ...  0.11753244  0.25244388
   0.93748909]
 [ 0.5353083  -0.54142342 -0.38082932 ... -0.55023477  0.15174998
  -0.87343483]
 ...
 [ 0.07826343 -1.14307742 -2.29195447 ...  0.16326992  0.72714374
  -0.77862206]
 [-0.73960634 -0.66175422 -0.20387329 ... -0.45875981  0.23805904
  -1.25268591]
 [-1.72586105 -0.8283661   1.24716618 ... -1.0487733   0.84222249
  -0.21606629]]
```

In [52]:

```
print(X_trained_scaled.mean(axis=0))
```

```
[ 9.38216641e-17  3.31503213e-16 -5.94203872e-16 -2.03280272e-16
 -2.15789827e-16  6.25477760e-17 -1.12585997e-16  2.50191104e-17
  0.00000000e+00  1.53242051e-16 -2.78337603e-16 -1.00076442e-16]
```

In [51]:

```
print(X_trained_scaled.std(axis=0))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [53]:

```
scaler = preprocessing.StandardScaler().fit(X_train)
```

In [54]:

```
X_train_scaled = scaler.transform(X_train)
print (X_train_scaled.mean(axis=0))
```

```
[ 9.38216641e-17  3.31503213e-16 -5.94203872e-16 -2.03280272e-16
 -2.15789827e-16  6.25477760e-17 -1.12585997e-16  2.50191104e-17
  0.00000000e+00  1.53242051e-16 -2.78337603e-16 -1.00076442e-16]
```

In [55]:

```
print(X_train_scaled.std(axis=0))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [59]:

```
X_test_scaled = scaler.transform(X_test)

print(X_test_scaled.mean(axis=0))
```

```
[-0.26351793  0.4183947   0.33092717  0.49384707  0.06325336 -0.26658949
 -0.5608802   0.48946899 -0.27040281  0.26020797 -0.44681937 -0.02301695]
```

In [58]:

```
print(X_test_scaled.std(axis=0))
```

```
[0.82929361 1.08311744 0.77667272 0.92630625 1.10528134 0.88878185
 0.98278206 1.14188373 1.1915591  1.23664934 1.01229222 0.96183193]
```

In [60]:

```
pipeline = make_pipeline(preprocessing.StandardScaler(),
                         RandomForestRegressor(n_estimators = 100))
```

In [61]:

```
print (pipeline.get_params())
```

```
{'memory': None, 'steps': [('standardscaler', StandardScaler(copy=True, wi
th_mean=True, with_std=True)), ('randomforestregressor', RandomForestRegre
ssor(bootstrap=True, criterion='mse', 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=100,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False))], 'verbose': False, 's
tandardscaler': StandardScaler(copy=True, with_mean=True, with_std=True),
'randomforestregressor': RandomForestRegressor(bootstrap=True, criterion
='mse', 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=100,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False), 'standardscaler__cop
y': True, 'standardscaler__with_mean': True, 'standardscaler__with_std': T
rue, 'randomforestregressor__bootstrap': True, 'randomforestregressor__cri
terion': 'mse', 'randomforestregressor__max_depth': None, 'randomforestreg
ressor__max_features': 'auto', 'randomforestregressor__max_leaf_nodes': No
ne, 'randomforestregressor__min_impurity_decrease': 0.0, 'randomforestregr
essor__min_impurity_split': None, 'randomforestregressor__min_samples_lea
f': 1, 'randomforestregressor__min_samples_split': 2, 'randomforestregress
or__min_weight_fraction_leaf': 0.0, 'randomforestregressor__n_estimators':
100, 'randomforestregressor__n_jobs': None, 'randomforestregressor__oob_sc
ore': False, 'randomforestregressor__random_state': None, 'randomforestreg
ressor__verbose': 0, 'randomforestregressor__warm_start': False}
```

In [62]:

```
hyperparameters = {'randomforestregressor__max_features' : ['auto','sqrt','log2'],
                   'randomforestregressor__max_depth': [None, 5,3,1]}
```

In [63]:

```
clf = GridSearchCV(pipeline, hyperparameters, cv = 10)
clf.fit(X_train, y_train)
```

```
C:\Users\Rodrigo\Anaconda3\lib\site-packages\sklearn\model_selection\_sear
ch.py:814: DeprecationWarning: The default of the `iid` parameter will cha
nge from True to False in version 0.22 and will be removed in 0.24. This w
ill change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[63]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
             estimator=Pipeline(memory=None,
                                steps=[('standardscaler',
                                        StandardScaler(copy=True,
                                                       with_mean=True,
                                                       with_std=True)),
                                       ('randomforestregressor',
                                        RandomForestRegressor(bootstrap=Tr
ue,
                                                              criterion='m
se',
                                                              max_depth=No
ne,
                                                              max_features
='auto',
                                                              max_leaf_nod
es=None,
                                                              min_impurity
_decrease=0.0,
                                                              min_impurity
_split=None,
                                                              min_...
                                                              min_weight_f
raction_leaf=0.0,
                                                              n_estimators
=100,
                                                              n_jobs=None,
                                                              oob_score=Fa
lse,
                                                              random_state
=None,
                                                              verbose=0,
                                                              warm_start=F
alse))],
                                verbose=False),
             iid='warn', n_jobs=None,
             param_grid={'randomforestregressor__max_depth': [None, 5, 3,
1],
                         'randomforestregressor__max_features': ['auto',
'sqrt',
                                                                 'log2']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring=None, verbose=0)
```

In [64]:

```
print(clf.best_params_)
```

```
{'randomforestregressor__max_depth': 3, 'randomforestregressor__max_featur
es': 'sqrt'}
```

In [65]:

```
print(clf.refit)
```

```
True
```

In [66]:

```
y_pred = clf.predict(X_test)
```

In [67]:

```
print(r2_score(y_test,y_pred))
```

```
0.5710623506567701
```

In [68]:

```
print(mean_squared_error(y_test,y_pred))
```

```
0.02484776889912628
```

In [69]:

```
joblib.dump(clf,'Rodrigo1randomForest_regressor.pkl')
```

Out[69]:

```
['Rodrigo1randomForest_regressor.pkl']
```

In [70]:

```
clf2 = joblib.load('Rodrigo1randomForest_regressor.pkl')

clf2.predict(X_test)
```

Out[70]:

```
array([0.74290006, 0.89704189, 0.70026183, 0.98213262, 1.05748068,
       0.76374498, 1.00102514, 0.72816523, 0.65690074, 1.01595131,
       0.74086949, 0.85065216, 0.64105684, 1.06085675, 1.05775599,
       0.82969856, 1.05881877, 0.96256935, 1.0618023 , 1.05325841,
       0.86353123, 0.71120722, 0.70664931, 0.80718862, 0.98043698,
       0.73417922, 0.72958309, 1.08920808, 1.07300193, 1.04364018,
       1.0685707 , 1.08071772, 0.8008607 , 1.02435671, 0.66032069,
       1.01952005])
```

In [ ]: