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

from csc665 import features, metrics

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

```
/Users/kalininalex/miniconda3/envs/py35_intel/lib/python3.5/site-pa
ckages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning:
numpy.core.umath_tests is an internal NumPy module and should not b
e imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:
```
csv_df = pd.read_csv("../Melbourne_housing_FULL.csv")
```

In [3]:
```
X, y = features.preprocess_ver_1(csv_df)
```

In [4]:
```
RANDOM_STATE = 10
X_train, X_test, y_train, y_test = train_test_split(X, y, test_si
```

In [5]:
```
X_train.shape, X_test.shape
```

Out[5]:
```
((7109, 20), (1778, 20))
```

In [6]:
```
rf = RandomForestRegressor(
    n_estimators=100,
    random_state=RANDOM_STATE,
    n_jobs=-1)
```

In [7]:
```
rf.fit(X_train, y_train)
```

Out[7]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=No
ne,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_split=1e-07, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=100, n_jobs=-1, oob_score=False, random_sta
te=10,
           verbose=0, warm_start=False)
```

In [8]:
```
y_train_pred = rf.predict(X_train)
```

In [9]:
```
metrics.r2_score(y_train_pred, y_train)
```

Out[9]:
```
0.9742581071101623
```

In [10]:
```
1 metrics.r2_score(rf.predict(X_test), y_test)
```

Out[10]:
```
0.8285220493562198
```

In [11]:
```
1 from sklearn.tree import DecisionTreeRegressor
```

In [12]:
```
1 dt = DecisionTreeRegressor()
```

In [13]:
```
1 dt.fit(X_train, y_train)
```

Out[13]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features
=None,
           max_leaf_nodes=None, min_impurity_split=1e-07,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, presort=False, random_stat
e=None,
           splitter='best')
```

In [14]:
```
1 metrics.r2_score(dt.predict(X_train), y_train)
```

Out[14]:
```
1.0
```

In [15]:
```
1 metrics.r2_score(dt.predict(X_test), y_test)
```

Out[15]:
```
0.6664240158986743
```

In [16]:
```
1 N = X_train.shape[0]
2 N
```

Out[16]:
```
7109
```

In [17]:
```
1 X_train.shape
```

Out[17]:
```
(7109, 20)
```

In [18]:
```
1 indices = np.random.randint(0, 7109, 700)
```

In [19]:
```
1 X_train.iloc[indices].shape
```

Out[19]:
```
(700, 20)
```

In [20]:
```
np.random.seed(RANDOM_STATE)
n = 200
trees = []
n_samples = X_train.shape[0]

for i in range(n):
    tree = DecisionTreeRegressor()
    indices = np.random.randint(0, n_samples, int(n_samples * 0.1
    _ = tree.fit(X_train.iloc[indices, :], y_train[indices])
    trees.append(tree)
```

Test Score

In [21]:
```
y_test.shape
```

Out[21]:
```
(1778,)
```

In [22]:
```
y_pred_list = []
for t in trees:
    y_pred_list.append(t.predict(X_test))
y_pred_list = np.array(y_pred_list)
```

In [23]:
```
y_pred_list.shape
```

Out[23]:
```
(200, 1778)
```

In [24]:
```
y_pred_sampled = y_pred_list.mean(axis=0)
```

In [25]:
```
metrics.r2_score(y_pred_sampled, y_test)
```

Out[25]:
```
0.8028782437450729
```

In [ ]:
```

```