```
In [14]: import numpy as np
         import pandas as pd
```

```
In [15]: csv_df = pd.read_csv("Melbourne_housing_FULL.csv")
```

```
In [16]: from sklearn.ensemble import RandomForestRegressor
```

```
In [17]: rf = RandomForestRegressor()
```

```
In [28]: csv_df.head()
```

Out[28]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017

5 rows × 21 columns

```
In [19]: feat_df = csv_df.drop('Price', axis=1)
```

```
In [20]: csv_df.shape

Out[20]: (34857, 21)
```

```
In [21]: feat_df.shape

Out[21]: (34857, 20)
```

```
In [24]: y = csv_df['Price'].values
```

```
In [25]: y

Out[25]: array([      nan, 1480000., 1035000., ...,  705000., 1140000., 10200
         00.])
```

In [26]:

```
y.shape
```

Out[26]:

```
(34857,)
```

In [34]:

```
rows_labeled_na = csv_df.isnull().any(axis=1)
```

In [37]:

```
rows_with_na = csv_df[rows_labeled_na]
```

In [38]:

```
rows_with_data = csv_df[~rows_labeled_na]
```

In [40]:

```
csv_df.shape, rows_with_na.shape, rows_with_data.shape
```

Out[40]:

```
((34857, 21), (25970, 21), (8887, 21))
```

In [43]:

```
feat_df = rows_with_data.drop('Price', axis=1)
```

In [44]:

```
feat_df.shape
```

Out[44]:

```
(8887, 20)
```

In [45]:

```
y = rows_with_data['Price'].values
y.shape
```

Out[45]:

```
(8887,)
```

In [51]:

```
suburbs = {}
for s in feat_df['Suburb'].values:
    if s not in suburbs:
        suburbs[s] = len(suburbs)
```

In [53]:

```
len(suburbs)
```

Out[53]:

```
315
```

In [60]:

```
feat_df['Suburb'] = feat_df['Suburb'].replace(suburbs)
```

In [61]:

```python
feat_df.head()
```

Out[61]:

	Suburb	Address	Rooms	Type	Method	SellerG	Date	Distance
2	0	25 Bloomburg St	2	h	S	Biggin	4/02/2016	2.5
4	0	5 Charles St	3	h	SP	Biggin	4/03/2017	2.5
6	0	55a Park St	4	h	VB	Nelson	4/06/2016	2.5
11	0	124 Yarra St	3	h	S	Nelson	7/05/2016	2.5
14	0	98 Charles St	2	h	S	Nelson	8/10/2016	2.5

In [72]:

```python
feat_df['Type'] = feat_df['Type'].astype('category').cat.codes
```

In [74]:

```python
feat_df['Address'] = feat_df['Address'].astype('category').cat.cod
feat_df['Method'] = feat_df['Method'].astype('category').cat.codes
feat_df['SellerG'] = feat_df['SellerG'].astype('category').cat.cod
feat_df['CouncilArea'] = feat_df['CouncilArea'].astype('category')
feat_df['Regionname'] = feat_df['Regionname'].astype('category').c
```

In [85]:

```python
feat_df.head()
```

Out[85]:

	Suburb	Address	Rooms	Type	Method	SellerG	Date
2	0	3922	2	0	1	22	1459555200000000000
4	0	6458	3	0	3	22	1491177600000000000
6	0	6960	4	0	4	147	1459900800000000000
11	0	1374	3	0	1	147	1467676800000000000
14	0	8740	2	0	1	147	1470787200000000000

In [80]:

```python
feat_df['Date'] = pd.to_datetime(feat_df['Date'],
    infer_datetime_format=True)
```

In [84]:

```python
feat_df['Date'] = feat_df['Date'].astype(np.int64)
```

In [86]:

```
feat_df.head()
```

Out[86]:

	Suburb	Address	Rooms	Type	Method	SellerG	Date	D
2	0	3922	2	0	1	22	1459555200000000000	
4	0	6458	3	0	3	22	1491177600000000000	
6	0	6960	4	0	4	147	1459900800000000000	
11	0	1374	3	0	1	147	1467676800000000000	
14	0	8740	2	0	1	147	1470787200000000000	

In [88]:

```
rf = RandomForestRegressor()
rf.fit(feat_df, y)
```

```
/Users/kalininalex/miniconda3/envs/py36/lib/python3.6/site-package
s/sklearn/ensemble/forest.py:248: FutureWarning: The default value
of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[88]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=No
ne,
           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=No
ne,
           oob_score=False, random_state=None, verbose=0, warm_star
t=False)
```

In [92]:

```
from sklearn.model_selection import train_test_split
```

In [93]:

```
? train_test_split
```

In [106]:

```
X_train, X_test, y_train, y_test = train_test_split(feat_df, y, ra
```

In [100]:

```
feat_df.shape, X_train.shape, X_test.shape, y_train.shape, y_test.
```

Out[100]:

```
((8887, 20), (6665, 20), (2222, 20), (6665,), (2222,))
```

In [124]:
```
rf = RandomForestRegressor(n_estimators=100 , random_state=17)
%time rf.fit(X_train, y_train)
```

```
CPU times: user 5.64 s, sys: 139 ms, total: 5.77 s
Wall time: 6.39 s
```

Out[124]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=No
ne,
           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=N
one,
           oob_score=False, random_state=17, verbose=0, warm_start=
False)
```

In [125]:
```
rf.score(X_train, y_train)
```

Out[125]:
```
0.9748081366886574
```

In [126]:
```
rf.score(X_test, y_test)
```

Out[126]:
```
0.8306171578843503
```

In [115]:
```
rf.predict(X_test)
```

Out[115]:
```
array([1178050., 1010750.,  865900., ..., 1120000.,  620400., 23226
00.])
```

In [ ]:
```
rf = RandomForestRegressor(random_state=17)
rf.fit(X_train, y_train)
```

In [129]:
```
rf.estimators_[0].predict(X_test)
```

Out[129]:
```
array([1350000.,  786000.,  890000., ...,  854000.,  623500., 27000
00.])
```

In [130]:
```
rf.estimators_[1].predict(X_test)
```

Out[130]:
```
array([ 910000., 1220000.,  795000., ..., 1120000.,  560000., 18500
00.])
```

In [131]:
```
rf.estimators_[2].predict(X_test)
```

Out[131]:
```
array([1250000., 1140000.,  875000., ..., 1115000.,  680500., 22000
00.])
```

In [132]:
```
rf.predict(X_test)
```

Out[132]:
```
array([1317370.  , 1030150.  ,  861895.  , ..., 1075386.28,  61103
8.64,
       2238520.  ])
```

```
In [134]: y_hat = rf.predict(X_test)
```

```
In [135]: y_hat
```

```
Out[135]: array([1317370.  , 1030150.  ,  861895.  , ..., 1075386.28,  61103
          8.64,
                 2238520.  ])
```

```
In [136]: y_test
```

```
Out[136]: array([ 981000.,  875000.,  700000., ...,  932000.,  572000., 22000
          00.])
```

```
In [138]: y_hat.shape, y_test.shape
```

```
Out[138]: ((2222,), (2222,))
```

```
In [150]: mse = ((y_hat - y_test) ** 2).mean()
          mse
```

```
Out[150]: 67775704512.086395
```

```
In [142]: rmse = np.sqrt(mse)
```

```
In [143]: mse, rmse
```

```
Out[143]: (67775704512.086395, 260337.6740160486)
```

```
In [151]: v = ((y_test - y_test.mean()) ** 2).mean()
          v
```

```
Out[151]: 400133234662.6414
```

```
In [153]: score = 1 - mse / v
          score
```

```
Out[153]: 0.8306171578843503
```