Categorical Variables and One Hot Encoding

```
import pandas as pd
```

```
df = pd.read_csv("homeprices.csv")
df
```

	town	area	price
0	monroe township	2600	550000
1	monroe township	3000	565000
2	monroe township	3200	610000
3	monroe township	3600	680000
4	monroe township	4000	725000
5	west windsor	2600	585000
6	west windsor	2800	615000
7	west windsor	3300	650000
8	west windsor	3600	710000
9	robinsville	2600	575000
10	robinsville	2900	600000
11	robinsville	3100	620000
12	robinsville	3600	695000

Using pandas to create dummy variables

```
dummies = pd.get_dummies(df.town)
dummies
```

	monroe township	robinsville	west windsor
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	0	0	1
6	0	0	1
7	0	0	1
8	0	0	1
9	0	1	0
10	0	1	0
11	0	1	0
12	0	1	0

```
merged = pd.concat([df,dummies],axis='columns')
merged
```

	town	area	price	monroe township	robinsville	west windsor
0	monroe township	2600	550000	1	0	0
1	monroe township	3000	565000	1	0	0
2	monroe township	3200	610000	1	0	0
3	monroe township	3600	680000	1	0	0
4	monroe township	4000	725000	1	0	0
5	west windsor	2600	585000	0	0	1
6	west windsor	2800	615000	0	0	1
7	west windsor	3300	650000	0	0	1
8	west windsor	3600	710000	0	0	1
9	robinsville	2600	575000	0	1	0
10	robinsville	2900	600000	0	1	0

```
final = merged.drop(['town'], axis='columns')
final
```

	area	price	monroe township	robinsville	west windsor
0	2600	550000	1	0	0
1	3000	565000	1	0	0
2	3200	610000	1	0	0
3	3600	680000	1	0	0
4	4000	725000	1	0	0

Dummy Variable Trap

When you can derive one variable from other variables, they are known to be multi-colinear. Here if you know values of california and georgia then you can easily infer value of new jersey state, i.e. california=0 and georgia=0. There for these state variables are called to be multi-colinear. In this situation linear regression won't work as expected. Hence you need to drop one column.

NOTE: sklearn library takes care of dummy variable trap hence even if you don't drop one of the state columns it is going to work, however we should make a habit of taking care of dummy variable trap ourselves just in case library that you are using is not handling this for you

```
final = final.drop(['west windsor'], axis='columns')
final
```

	area	price	monroe township	robinsville
0	2600	550000	1	0
1	3000	565000	1	0
2	3200	610000	1	0
3	3600	680000	1	0
4	4000	725000	1	0
5	2600	585000	0	0
6	2800	615000	0	0

```
X = final.drop('price', axis='columns')
X
```

	area	monroe township	robinsville
0	2600	1	0

```
y = final.price
```

2 3200 1 0

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

4	4000	1	0

```
model.fit(X,y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

7 3300 0 0

```
model.predict(X) # 2600 sqr ft home in new jersey
```

```
array([539709.7398409 , 590468.71640508, 615848.20468716, 666607.18125134,
       717366.15781551, 579723.71533005, 605103.20361213, 668551.92431735,
       706621.15674048, 565396.15136531, 603465.38378844, 628844.87207052,
       692293.59277574])
```

11 3100 0 1

```
model.score(X,y)
```

```
0.9573929037221873
```

```
model.predict([[3400,0,0]]) # 3400 sqr ft home in west windsor
```

```
array([681241.66845839])
```

```
model.predict([[2800,0,1]]) # 2800 sqr ft home in robbinsville
```

```
array([590775.63964739])
```

Using sklearn OneHotEncoder

First step is to use label encoder to convert town names into numbers

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
dfle = df
dfle.town = le.fit_transform(dfle.town)
dfle
```

	town	area	price
0	0	2600	550000
1	0	3000	565000
2	0	3200	610000
3	0	3600	680000
4	0	4000	725000
5	2	2600	585000
6	2	2800	615000
7	2	3300	650000
8	2	3600	710000
9	1	2600	575000
10	1	2900	600000
11	1	3100	620000
12	1	3600	695000

```
X = dfle[['town','area']].values
```

```
X
```

```
array([[   0, 2600],
       [   0, 3000],
       [   0, 3200],
       [   0, 3600],
       [   0, 4000],
       [   2, 2600],
       [   2, 2800],
       [   2, 3300],
       [   2, 3600],
       [   1, 2600],
       [   1, 2900],
       [   1, 3100],
       [   1, 3600]])
```

```
y = dfle.price.values
y
```

```
array([550000, 565000, 610000, 680000, 725000, 585000, 615000, 650000,
       710000, 575000, 600000, 620000, 695000])
```

Now use one hot encoder to create dummy variables for each of the town

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([('town', OneHotEncoder(), [0])], remainder = 'passthrough')
```

```
X = ct.fit_transform(X)
X
```

```
array([[1.0e+00, 0.0e+00, 0.0e+00, 2.6e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.0e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.2e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.6e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 4.0e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 2.6e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 2.8e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 3.3e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 3.6e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 2.6e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 2.9e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 3.1e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 3.6e+03]])
```

```
X = X[:,1:]
```

```
X
```

```
     array([[0.0e+00, 0.0e+00, 2.6e+03],
            [0.0e+00, 0.0e+00, 3.0e+03],
            [0.0e+00, 0.0e+00, 3.2e+03],
            [0.0e+00, 0.0e+00, 3.6e+03],
            [0.0e+00, 0.0e+00, 4.0e+03],
            [0.0e+00, 1.0e+00, 2.6e+03],
            [0.0e+00, 1.0e+00, 2.8e+03],
            [0.0e+00, 1.0e+00, 3.3e+03],
            [0.0e+00, 1.0e+00, 3.6e+03],
            [1.0e+00, 0.0e+00, 2.6e+03],
            [1.0e+00, 0.0e+00, 2.9e+03],
            [1.0e+00, 0.0e+00, 3.1e+03],
            [1.0e+00, 0.0e+00, 3.6e+03]])
```

```
model.fit(X,y)
```

```
     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
model.predict([[0,1,3400]]) # 3400 sqr ft home in west windsor
```

```
     array([681241.6684584])
```

```
model.predict([[1,0,2800]]) # 2800 sqr ft home in robbinsville
```

```
    array([590775.63964739])
```

Exercise

At the same level as this notebook on github, there is an Exercise folder that contains carprices.csv. This file has car sell prices for 3 different models. First plot data points on a scatter plot chart to see if linear regression model can be applied. If yes, then build a model that can answer following questions,

1) Predict price of a mercedez benz that is 4 yr old with mileage 45000

2) Predict price of a BMW X5 that is 7 yr old with mileage 86000

3) Tell me the score (accuracy) of your model. (Hint: use LinearRegression().score())