# one-hot-encode-technique-2

April 2, 2024

Categorical Variables and One Hot Encoding

[55]:
```
import pandas as pd
df=pd.read_csv('/content/homeprices (2).csv')
df
```

[55]:
```
                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

[56]:
```
dummies=pd.get_dummies(df.town)
dummies
```

[56]:
```
    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
```

cocatenate dummies df with the original df

[57]:

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

[57]:

```
                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
11       robinsville  3100  620000                0            1             0
12       robinsville  3600  695000                0            1             0
```

dropping one of the dummy variable columns so as not to create multicollinearity

[58]:

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

[58]:

```
    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
7   3300  650000                0            0
8   3600  710000                0            0
9   2600  575000                0            1
10  2900  600000                0            1
11  3100  620000                0            1
12  3600  695000                0            1
```

create a linear regression model

[59]:

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

find x in the model and remove the price column since its a dependent variable

```
[60]: x = final.drop('price', axis='columns')
      x
```

```
[60]:     area  monroe township  robinsville
      0   2600                1            0
      1   3000                1            0
      2   3200                1            0
      3   3600                1            0
      4   4000                1            0
      5   2600                0            0
      6   2800                0            0
      7   3300                0            0
      8   3600                0            0
      9   2600                0            1
      10  2900                0            1
      11  3100                0            1
      12  3600                0            1
```

```
[61]: y = final.price
      y
```

```
[61]: 0     550000
      1     565000
      2     610000
      3     680000
      4     725000
      5     585000
      6     615000
      7     650000
      8     710000
      9     575000
      10    600000
      11    620000
      12    695000
      Name: price, dtype: int64
```

Training the machine learning model

```
[62]: model.fit(x,y)
```

```
[62]: LinearRegression()
```

making predictions

```
[63]: model.predict([[2800,0,1]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

```
[63]: array([590775.63964739])
```

```
[64]: model.predict([[3400,0,0]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

```
[64]: array([681241.66845839])
```

Measuring the accuracy of a model

```
[65]: model.score(x,y)
```

```
[65]: 0.9573929037221872
```

```
[66]: df
```

```
[66]:
                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 sklearn OneHotEncoder

In order to use one hot encording you need to do lebel encording on the town column

```
[67]: from sklearn.preprocessing import LabelEncoder
```

```
[68]: le = LabelEncoder()
```

```
[69]: dfle =df
      dfle.town = le.fit_transform(dfle.town)
      dfle
```

```
[69]:     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
```

```
[70]: x =dfle[['town', 'area']].values
      x
```

```
[70]: 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]])
```

```
[71]: y = dfle.price
      y
```

```
[71]: 0      550000
      1      565000
      2      610000
      3      680000
      4      725000
      5      585000
      6      615000
      7      650000
      8      710000
      9      575000
      10     600000
      11     620000
```

```
12    695000
Name: price, dtype: int64
```

Import one hot encorder

[72]:

```
from sklearn.preprocessing import OneHotEncoder
```

[73]:

```
from sklearn.preprocessing import OneHotEncoder
```

[74]:

```
ohe = OneHotEncoder
```

[93]:

```
ohe = OneHotEncoder(sparse=False)  # Avoid sparse matrix output
x_encoded = ohe.fit_transform(x)
print(x_encoded)
```

```
[[1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will
be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its
default value.
  warnings.warn(
```

[94]:

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

[94]:

```
LinearRegression()
```

[100]:

```
model.predict([[1,2800]])
```

[100]:

```
array([587143.58452138])
```

[109]:

```
model.predict([[1,3400]])
```

[109]:

```
array([662776.40384056])
```