**La data tiene campo town : monrow township / west windsor / robinsville**

1. **Convertimos campo town en 3 campos con valores 1, 0**
   1. **Eliminamos campo town**
2. **Eliminamos west Windsor para hacer el modelo menos complejo colineal**
3. **Creamos los dataset**
   1. **X sin Price**
   2. **Y con Price**
4. **Damos fit model(X, y)**
5. **Predecimos los valores de y con model.predic(X) y comparamos**
6. **Verificamos accuracy model.score(X,y)**

Predecir

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

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

**área Price monroe west Windsor Robbinsville**

**2600 550 000 0 0 1 Robbinsville (0,0,1)**

**3400 650 000 0 1 0 west Windsor (0,1,0)**

**3460 800 000 1 0 0 monroe (1,0,0)**

**Y se puede simplificar a**

**área Price west Windsor Robbinsville**

**2600 550 000 0 1 Robbinsville (0,1)**

**3400 650 000 1 0 west Windsor (1,0)**

**3460 800 000 0 0 monroe (0,0)**

**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 |

**Creamos 3 columnas basadas en Town y luego borramos Town**

**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')

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 |
| **5** | 2600 | 585000 | 0 | 0 | 1 |
| **6** | 2800 | 615000 | 0 | 0 | 1 |
| **7** | 3300 | 650000 | 0 | 0 | 1 |
| **8** | 3600 | 710000 | 0 | 0 | 1 |
| **9** | 2600 | 575000 | 0 | 1 | 0 |
| **10** | 2900 | 600000 | 0 | 1 | 0 |
| **11** | 3100 | 620000 | 0 | 1 | 0 |
| **12** | 3600 | 695000 | 0 | 1 | 0 |

### Dummy Variable Trap

When you can derive one variable from other variables, they are known to be multi-colinear.

Values from California ar like the ones from Georgia. 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,**

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

final

|  |  |  |  |
| --- | --- | --- | --- |
|  | **area** | **price** | **Monroe** | **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 |

In [14]:

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

X

| **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 |

y **=** final**.**price

**from** sklearn.linear\_model **import** LinearRegression

model **=** LinearRegression()

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

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

Out[20]:

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])

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

Out[21]:

0.9573929037221873

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

Out[22]:

array([681241.66845839])

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

Out[23]:

array([590775.63964739])