**Otra alternativa**

**Using sklearn OneHotEncoder**

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

Monroe township = 1

West Windsor = 2

Robbinszvilee =3

**from** sklearn.preprocessing **import** LabelEncoder

le **=** LabelEncoder()

dfle **=** df

dfle**.**town **=** le**.**fit\_transform(dfle**.**town)

dfle

Out[25]:

|  | **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

Out[28]:

array([550000, 565000, 610000, 680000, 725000,

585000, 615000, 650000, 710000,

575000, 600000, 620000, 695000])

**Otra forma**

y **=** dfle**.**price**.**values

**X =** dfle**.drop(‘**price’, inplace=True, axis=1)

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

Out[30]:

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]])

Son 3 grupos

(1,0,0, y)

(0,0,1, y)

(0,1,0, y)

Para simplificar eliminamos primera fila

(0, 0, y)

(0, 1, y)

(1,0, y)

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)

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

Out[34]:

array([681241.6684584])

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

Out[35]:

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())**