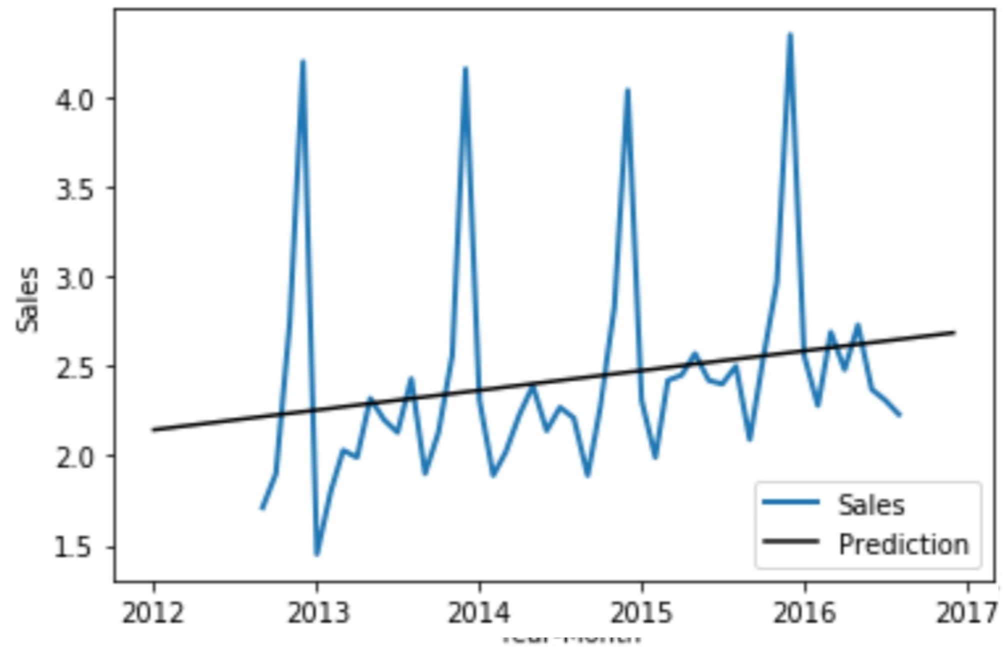
LINEAR REGRESSION WITH CATEGORICAL VARIABLES

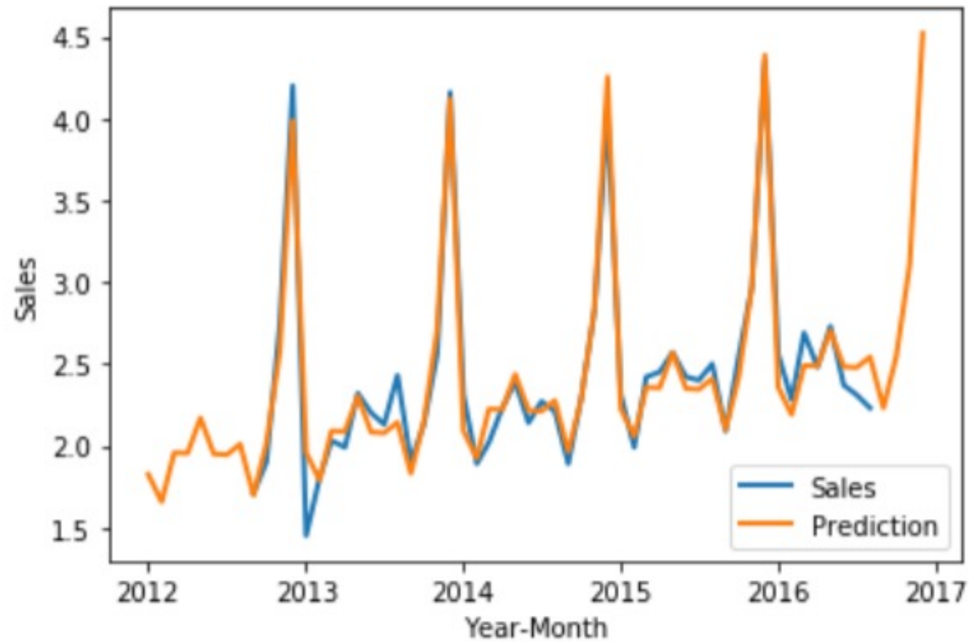
OVERVIEW

- This is linear regression



OVERVIEW

- Is this linear regression?



- Negative Adj R-square
- one-hot encoding with sklearn

REGRESSION ASSUMPTIONS

- $Y_1, Y_2, \ldots, Y_n$ are random vars.
- independent (*independence)*
- normal (*normality)*
- with same variance (*constant variance)*
- Model

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

REGRESSION ASSUMPTIONS

- Y_1, Y_2, ...,Y_n are random vars.
- independent (*independence)*
- normal (*normality)*
- with same variance (*constant variance)*
- Model

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

random variables

REGRESSION ASSUMPTIONS

- $Y_1, Y_2, \ldots, Y_n$ are random vars.
- independent (*independence)*
- normal (*normality)*
- with same variance (*constant variance)*
- Model

$$Y = \underbrace{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}_{\textit{not random}} + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

not random

REGRESSION ASSUMPTIONS

- $Y_1, Y_2, \ldots, Y_n$ are random vars.
- independent (*independence*)
- normal (*normality*)
- with same variance (*constant variance*)

Regression Model

$$\begin{cases} Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon & \epsilon \sim N(0, \sigma^2) \\ E[Y] = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \\ \hat{Y} = b_0 + b_1 X_1 + \cdots + b_p X_p \end{cases}$$

OVERVIEW

- Regression models review
- Regression with a Categorical predictor

  (with 2 or 3 categories)
- Interaction between predictors
- Examples
  - Encoding methods
  - Forecasting with categorical variables
  - Regression with many categorical variables

EXAMPLES

Regression with a Categorical Variable

with 2 categories

EXAMPLES

Introductory Example 1

REGRESSION WITH A CATEGORICAL VARIABLE

	X1	X2	Y
0	S	-0.10	19.19
1	S	2.53	22.74
2	S	4.86	23.91
3	M	0.26	7.07
4	M	2.55	7.93
5	M	4.87	8.93
6	L	0.08	20.63
7	L	2.62	23.46
8	L	5.09	25.75

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

How do we incorporate X_1 in the model?

REGRESSION WITH A CATEGORICAL VARIABLE

- X_1 to be replaced by binary variables
- The number *p* of predictors will change

	X1	L	M	S	X2	Y
0	S	0	0	1	-0.10	19.19
1	S	0	0	1	2.53	22.74
2	S	0	0	1	4.86	23.91
3	M	0	1	0	0.26	7.07
4	M	0	1	0	2.55	7.93
5	M	0	1	0	4.87	8.93
6	L	1	0	0	0.08	20.63
7	L	1	0	0	2.62	23.46
8	L	1	0	0	5.09	25.75

REGRESSION WITH A CATEGORICAL VARIABLE

- Binary variable S is not needed
- When M = 0, L = 0, X_1 must be S

	X1	L	M	S	X2	Y
0	S	0	0	1	-0.10	19.19
1	S	0	0	1	2.53	22.74
2	S	0	0	1	4.86	23.91
3	M	0	1	0	0.26	7.07
4	M	0	1	0	2.55	7.93
5	M	0	1	0	4.87	8.93
6	L	1	0	0	0.08	20.63
7	L	1	0	0	2.62	23.46
8	L	1	0	0	5.09	25.75

REGRESSION WITH A CATEGORICAL VARIABLE

- Binary variable S is not needed
- When M = 0, L = 0, X_1 must be S
- The number of binary variables is equal to the number of categories minus 1

	X1	L	M	X2	Y
0	S	0	0	-0.10	19.19
1	S	0	0	2.53	22.74
2	S	0	0	4.86	23.91
3	M	0	1	0.26	7.07
4	M	0	1	2.55	7.93
5	M	0	1	4.87	8.93
6	L	1	0	0.08	20.63
7	L	1	0	2.62	23.46
8	L	1	0	5.09	25.75

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \rightarrow Y = \beta_0 + \beta_M M + \beta_L L + \beta_2 X_2 + \epsilon$$

	X1	X2	Y
0	S	-0.10	19.19
1	S	2.53	22.74
2	S	4.86	23.91
3	M	0.26	7.07
4	M	2.55	7.93
5	M	4.87	8.93
6	L	0.08	20.63
7	L	2.62	23.46
8	L	5.09	25.75

p = 2

	L	M	X2	Y
0	0	0	-0.10	19.19
1	0	0	2.53	22.74
2	0	0	4.86	23.91
3	0	1	0.26	7.07
4	0	1	2.55	7.93
5	0	1	4.87	8.93
6	1	0	0.08	20.63
7	1	0	2.62	23.46
8	1	0	5.09	25.75

p = 3

REGRESSION WITH A CATEGORICAL VARIABLE

Transform this model $$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

into

this new model $$Y = \beta_0 + \beta_M M + \beta_L L + \beta_2 X_2 + \epsilon$$

$$M = \begin{cases} 1 & \text{if } X_1 = M \\ 0 & \text{ow} \end{cases}$$

$$L = \begin{cases} 1 & \text{if } X_1 = L \\ 0 & \text{ow} \end{cases}$$

REGRESSION WITH A CATEGORICAL VARIABLE

Transform this model $$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

into

this new model $$Y = \beta_0 + \beta_M M + \beta_L L + \beta_2 X_2 + \epsilon$$

substituting X_1 with binary variables M and L

$$M = \begin{cases} 1 & \text{if } X_1 = M \\ 0 & \text{ow} \end{cases}$$

$$L = \begin{cases} 1 & \text{if } X_1 = L \\ 0 & \text{ow} \end{cases}$$

EXAMPLES

Introductory Example 2

REGRESSION WITH A CATEGORICAL VARIABLE

Predict the Price of a car using MPG.city and Origin

Y	numerical	categorical
Price	**MPG_city**	**Origin**
9.0	31	USA
11.1	23	USA
15.7	22	USA
19.7	17	non-USA
22.7	21	non-USA
9.2	29	USA

REGRESSION WITH A CATEGORICAL VARIABLE

Y	X_2	X_1
Price	**MPG_city**	**Origin**
9.0	31	USA
11.1	23	USA
15.7	22	USA
19.7	17	non-USA
22.7	21	non-USA
9.2	29	USA

Origin with two categories

- USA cars
- non-USA cars

We will find an OLS line
for each category
in just one model

REGRESSION WITH A CATEGORICAL VARIABLE

Notation

Y : Price of the car

X_1 : Origin (USA car, non-USA car)

X_2 : City Mileage (MPG.city)

replace X_1 with a binary variable

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

REGRESSION WITH A CATEGORICAL VARIABLE

Model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

becomes two models

REGRESSION WITH A CATEGORICAL VARIABLE

Model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

becomes two models

$$Y = \beta_0 + \beta_2 X_2 + \epsilon \qquad (x_1 = 0)$$

$$Y = (\beta_0 + \beta_1) + \beta_2 X_2 + \epsilon \qquad (x_1 = 1)$$

REGRESSION WITH A CATEGORICAL VARIABLE

Model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

resulting in two OLS lines

- For US cars $\hat{Y} = b_0 + b_2 X_2 \qquad (x_1 = 0)$
- For non-US cars $\hat{Y} = (b_0 + b_1) + b_2 X_2 \qquad (x_1 = 1)$

REGRESSION WITH A CATEGORICAL VARIABLE

Model $$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

resulting in two OLS lines

$$\hat{Y} = b_0 + b_2 X_2 \qquad (x_1 = 0)$$

$$\hat{Y} = (b_0 + b_1) + b_2 X_2 \qquad (x_1 = 1)$$

additional intercept ↑ ↑ slope

REGRESSION WITH A CATEGORICAL VARIABLE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import statsmodels.formula.api as smf
```

```
df = pd.read_csv('Cars93.csv')
```

```
# statsmodels.formula.api does not work with dots in Column names
# thus, replace dots from Column names with '_'
```

```
df.columns=df.columns.str.replace('.','_', regex = True)
df.columns
```

```
Index(['Manufacturer', 'Model', 'Type', 'Min_Price', 'Price', 'Max_Price', 'MPG_city',
       'MPG_highway', 'AirBags', 'DriveTrain', 'Cylinders', 'EngineSize', 'Horsepower',
       'RPM', 'Rev_per_mile', 'Man_trans_avail', 'Fuel_tank_capacity', 'Passengers',
       'Length', 'Wheelbase', 'Width', 'Turn_circle', 'Rear_seat_room', 'Luggage_room',
       'Weight', 'Origin', 'Make'],
```

REGRESSION WITH A CATEGORICAL VARIABLE

Fit Model

```
m1 = smf.ols(formula = 'Price~MPG_city + C(Origin)',data = df).fit()
m1.params
```

```
Intercept                  42.555991
C(Origin)[T.non-USA]        5.264041    ← Additional intercept
MPG_city                   -1.144322    ← slope
```

REGRESSION WITH A CATEGORICAL VARIABLE

Fit Model

```
m1 = smf.ols(formula = 'Price~MPG_city + C(Origin)',data = df).fit()
m1.params
```

```
Intercept                    42.555991
C(Origin)[T.non-USA]
MPG_city                     -1.144322
```

← slope

$$\hat{Y} = 42.55 - 1.144\,X_2$$

Model for US cars

REGRESSION WITH A CATEGORICAL VARIABLE

Fit Model

```
m1 = smf.ols(formula = 'Price~MPG_city + C(Origin)',data = df).fit()
m1.params
```

```
Intercept                42.555991
C(Origin)[T.non-USA]      5.264041   ← Additional intercept
MPG_city                 -1.144322   ← slope
```

$$\hat{Y} = 42.55 - 1.144\, X_2$$

Model for US cars

$$\hat{Y} = (42.55 + 5.264) - 1.144\, X_2$$

Model for non-US cars

↑ additional intercept

EXAMPLE 2

How much more expensive are non-US cars?

PIVOT TABLE

How much more expensive are non-US cars?

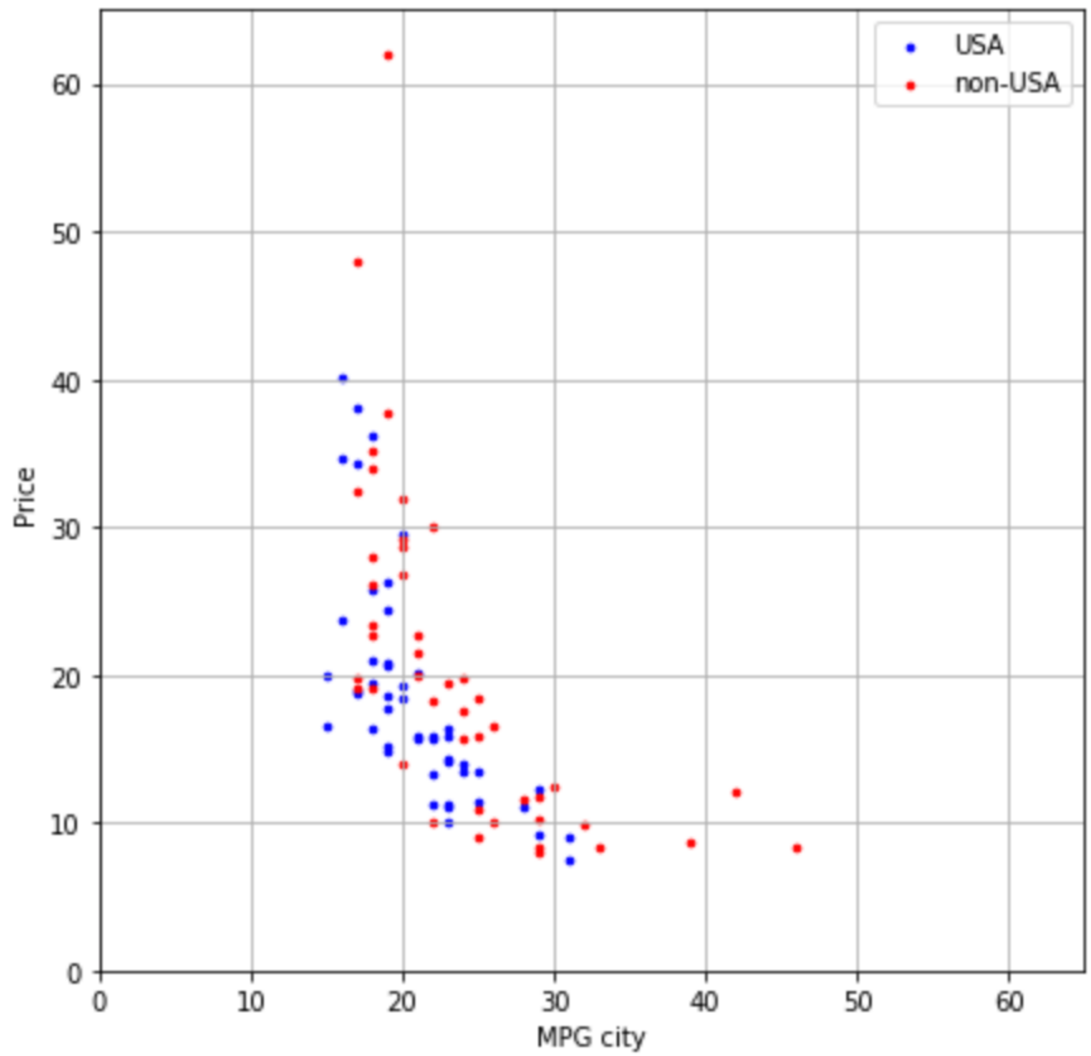
```
df.pivot_table(values = 'Price',index = 'Origin')
```

	Price
Origin	
USA	18.572917
non-USA	20.508889

non-USA cars are on average $1,936 more expensive

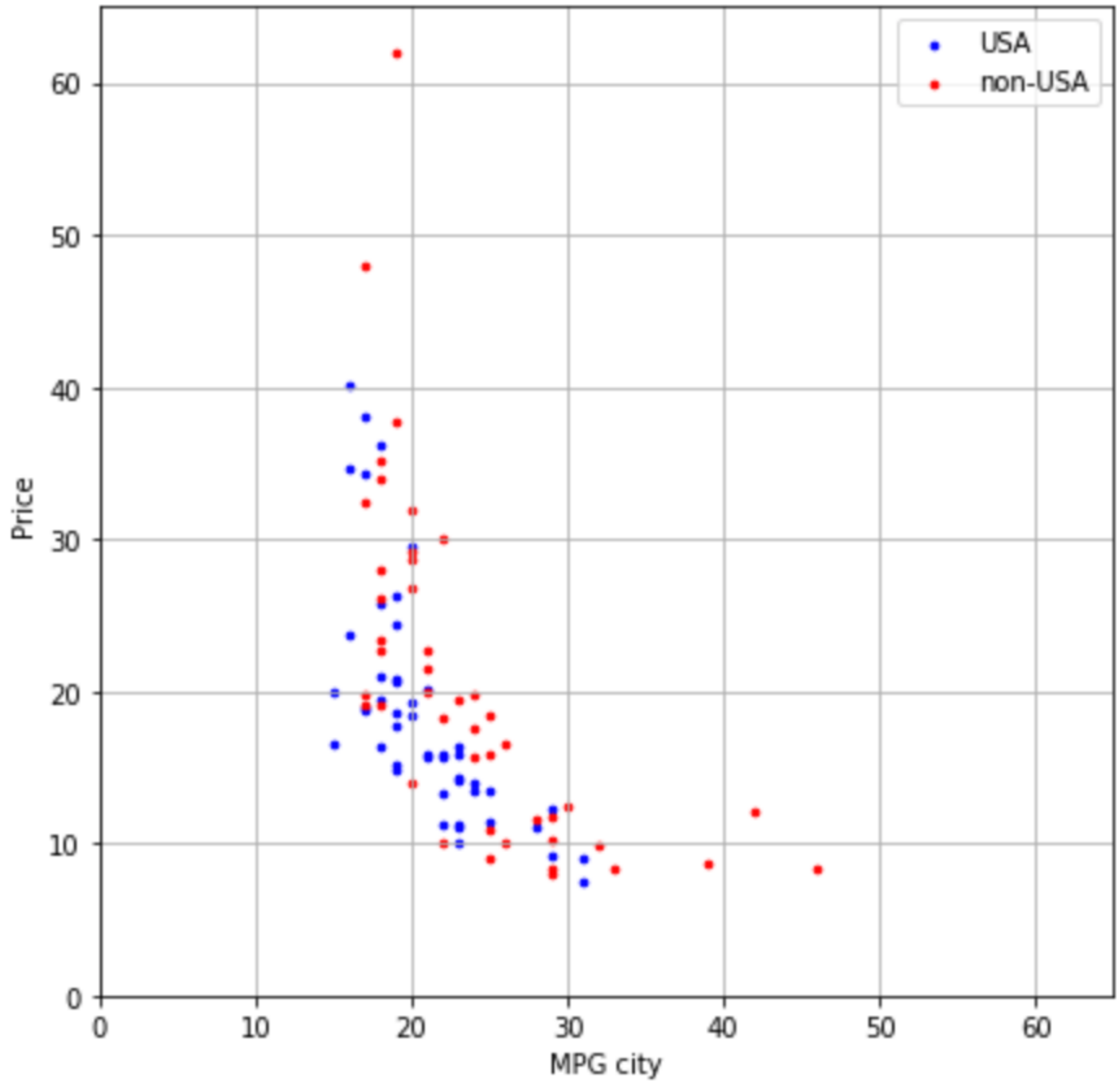
SCATTERPLOT with data points classified by Origin

```
# split DataFrame by categories

df_USA = df3[df3.Origin == 'USA']
df_nonUSA = df3[df3.Origin != 'USA']

# USA cars
plt.scatter(df_USA.MPG_city,df_USA.Price,
            c='b',s=7,label = 'USA')
# non-USA cars
plt.scatter(df_nonUSA.MPG_city,df_nonUSA.Price,
            c='r',s=7,label='non-USA')

plt.axis(xmin=0,xmax=50,ymin=0,ymax=65)
plt.legend()
plt.xlabel('MPG city')
plt.ylabel('Price')
```



SCATTERPLOT with data points classified by Origin

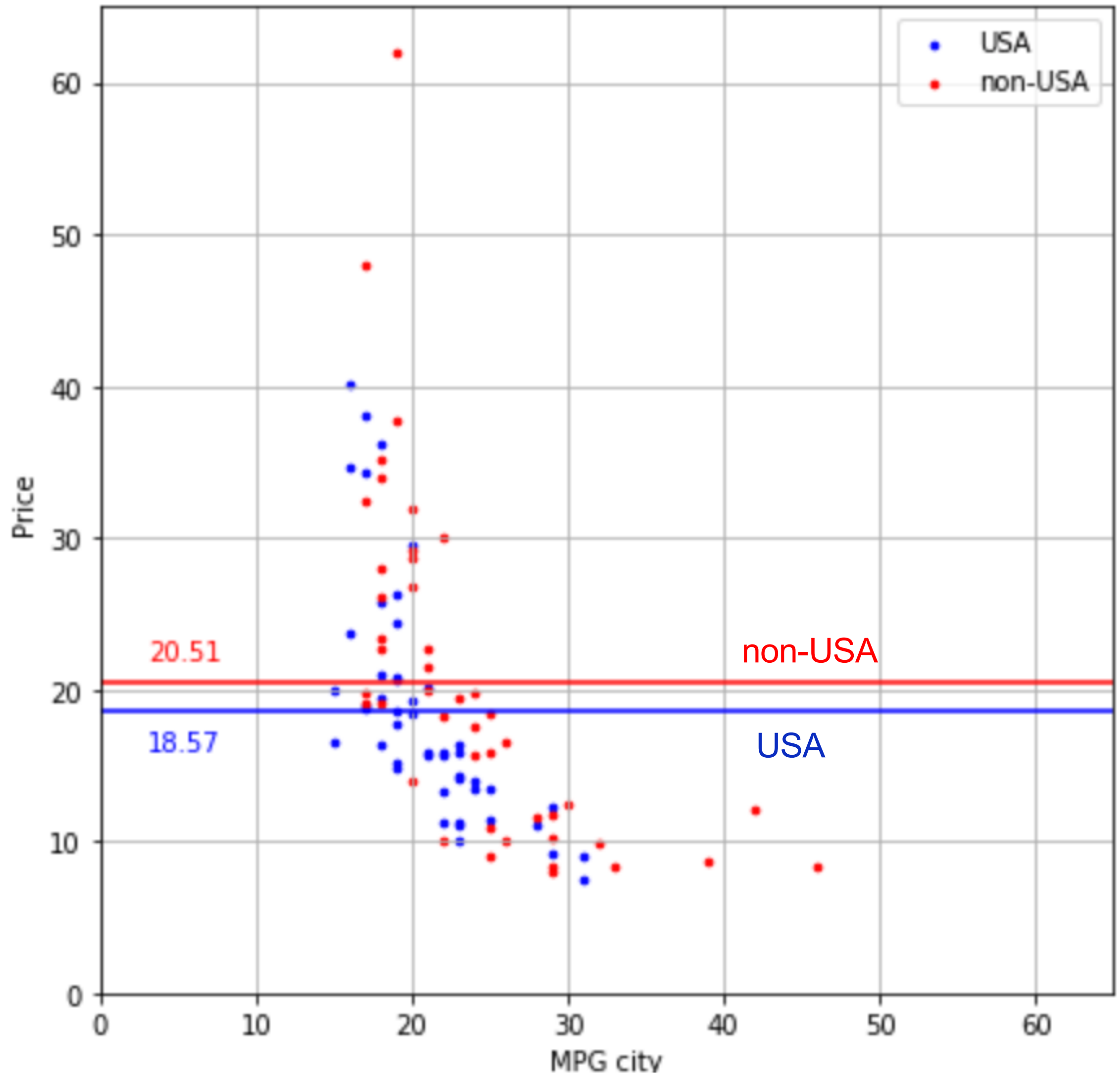
non-USA cars are
on average
more expensive



PIVOT TABLE and SCATTERPLOT

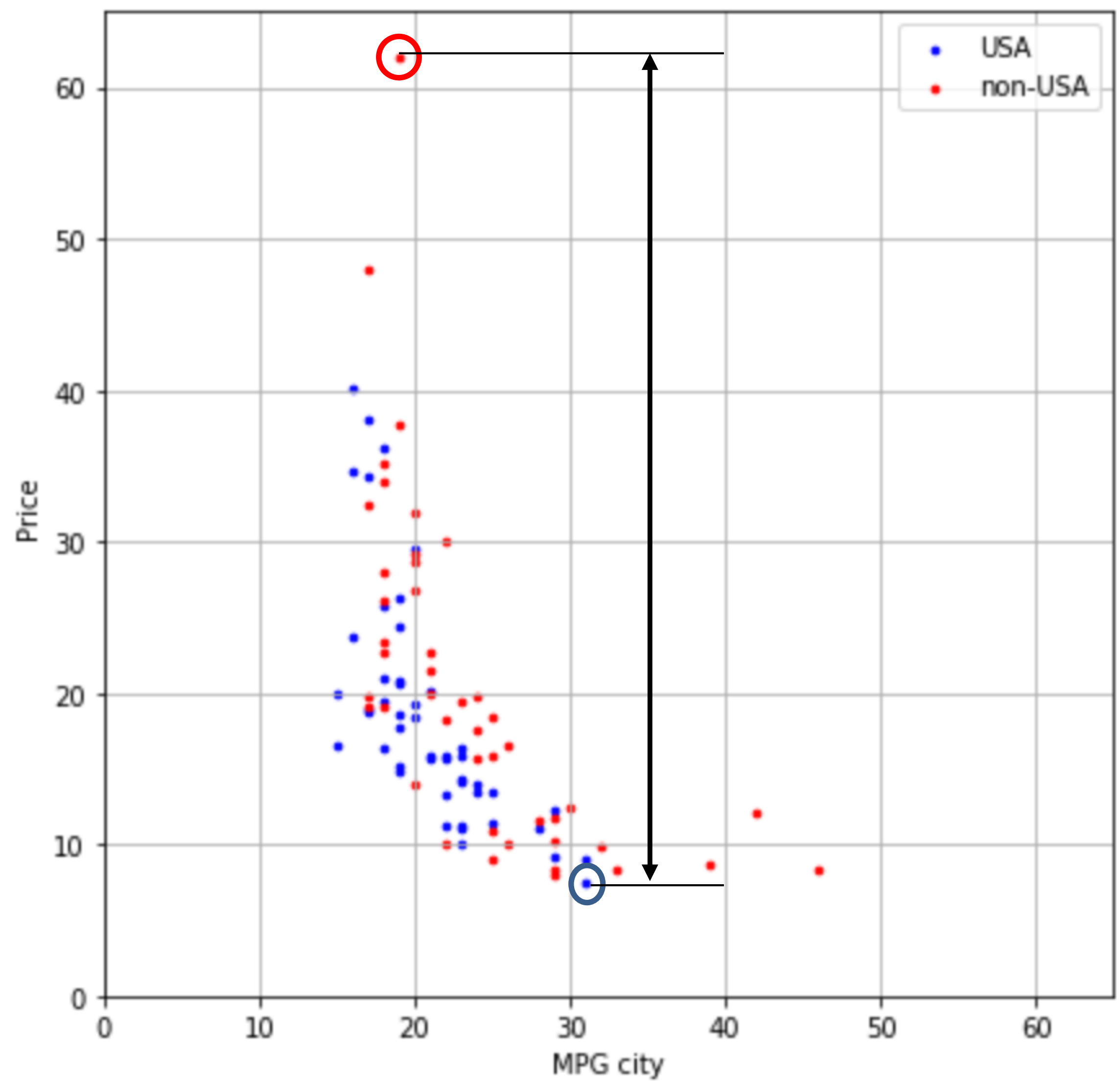
non-USA cars are on average more expensive

Origin	Average Price
USA	18.572917
non-USA	20.508889
difference	1.936



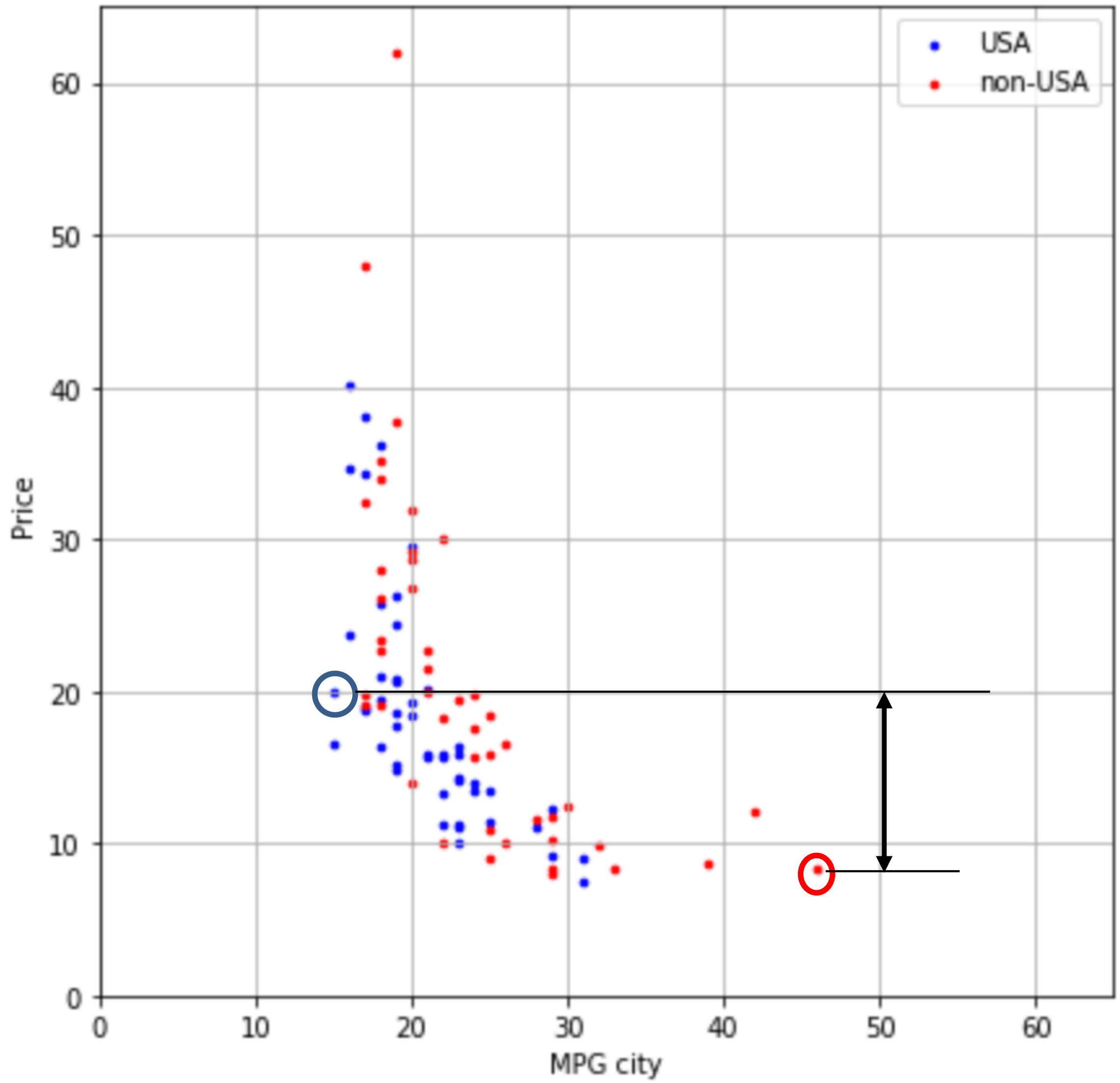
SCATTERPLOT with data points classified by Origin

Origin	Average Price
USA	18.572917
non-USA	20.508889
difference	1.936



SCATTERPLOT with data points classified by Origin

Origin	Average Price
USA	18.572917
non-USA	20.508889
difference	1.936



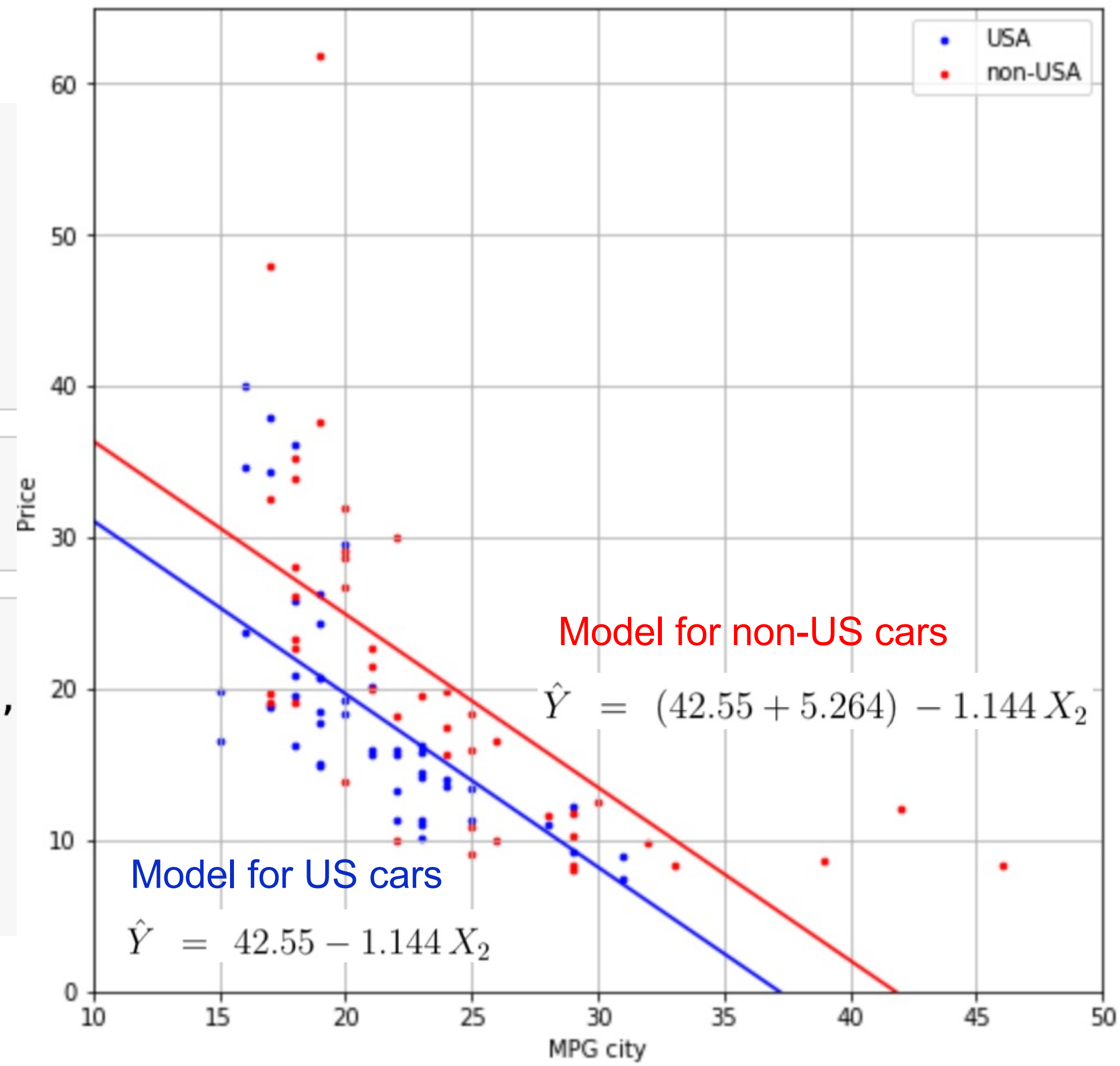
REGRESSION MODEL

.

```
# USA cars fitted line
def f0(x):
    return m1.params[0] + x*m1.params[2]

# non-USA cars fitted line
def f1(x):
    return m1.params[0] + m1.params[1]+\
                          x*m1.params[2]
```

```
x = np.linspace(0,50,100)
y0 = [f0(i) for i in x]
y1 = [f1(i) for i in x]
```

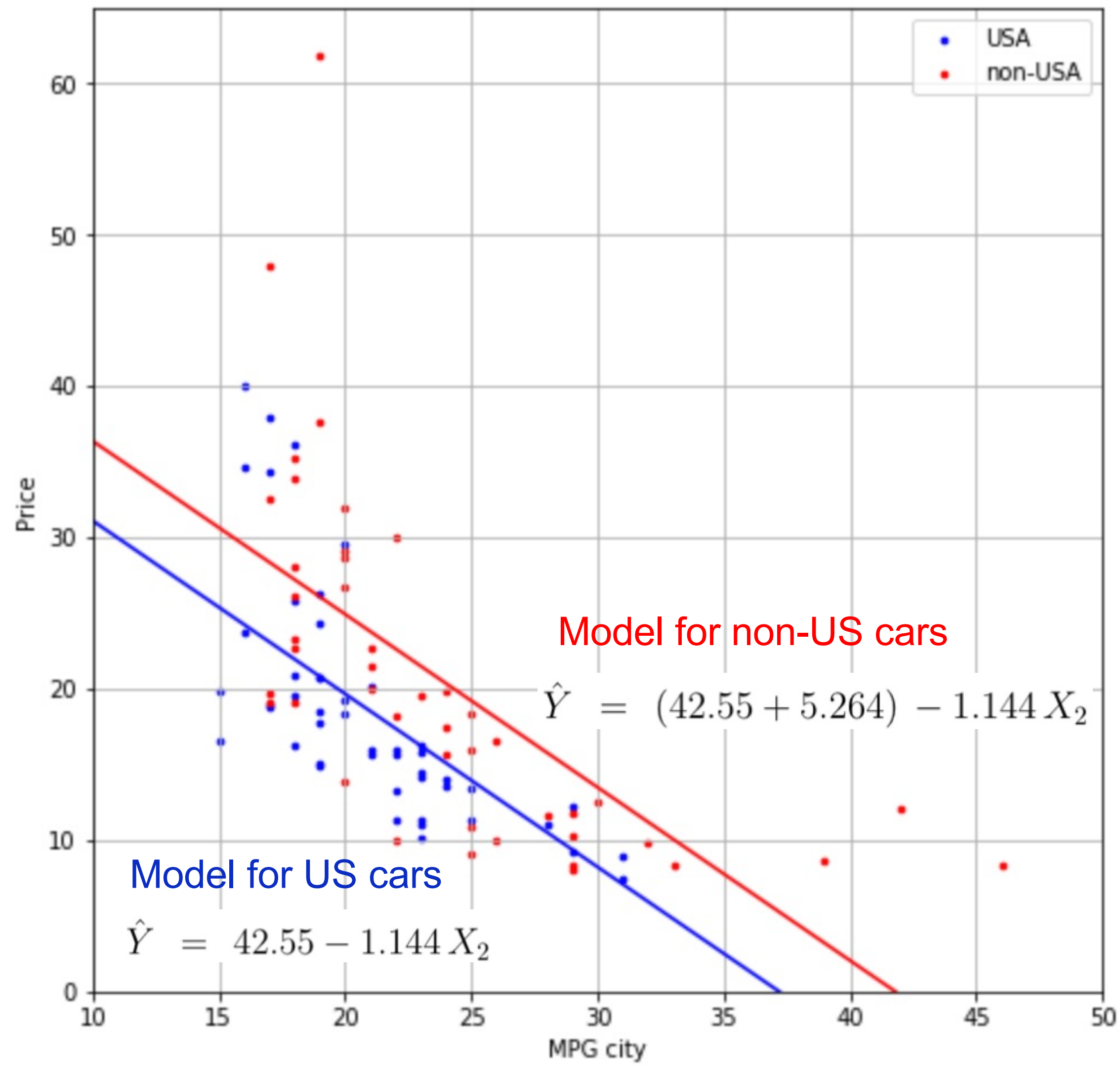
```
plt.scatter(df_USA.MPG_city,df_USA.Price,
            c='b',s=7,label = 'USA')
plt.scatter(df_nonUSA.MPG_city,df_nonUSA.Price,
            c='r',s=7,label='non-USA')

# USA cars line
plt.plot(x,y0,c='b')
# non-USA cars line
plt.plot(x,y1,c='r')
```



REGRESSION MODEL

```
m1.params
```

```
Intercept                42.555991
Origin[T.non-USA]         5.264041
MPG_city                 -1.144322
```

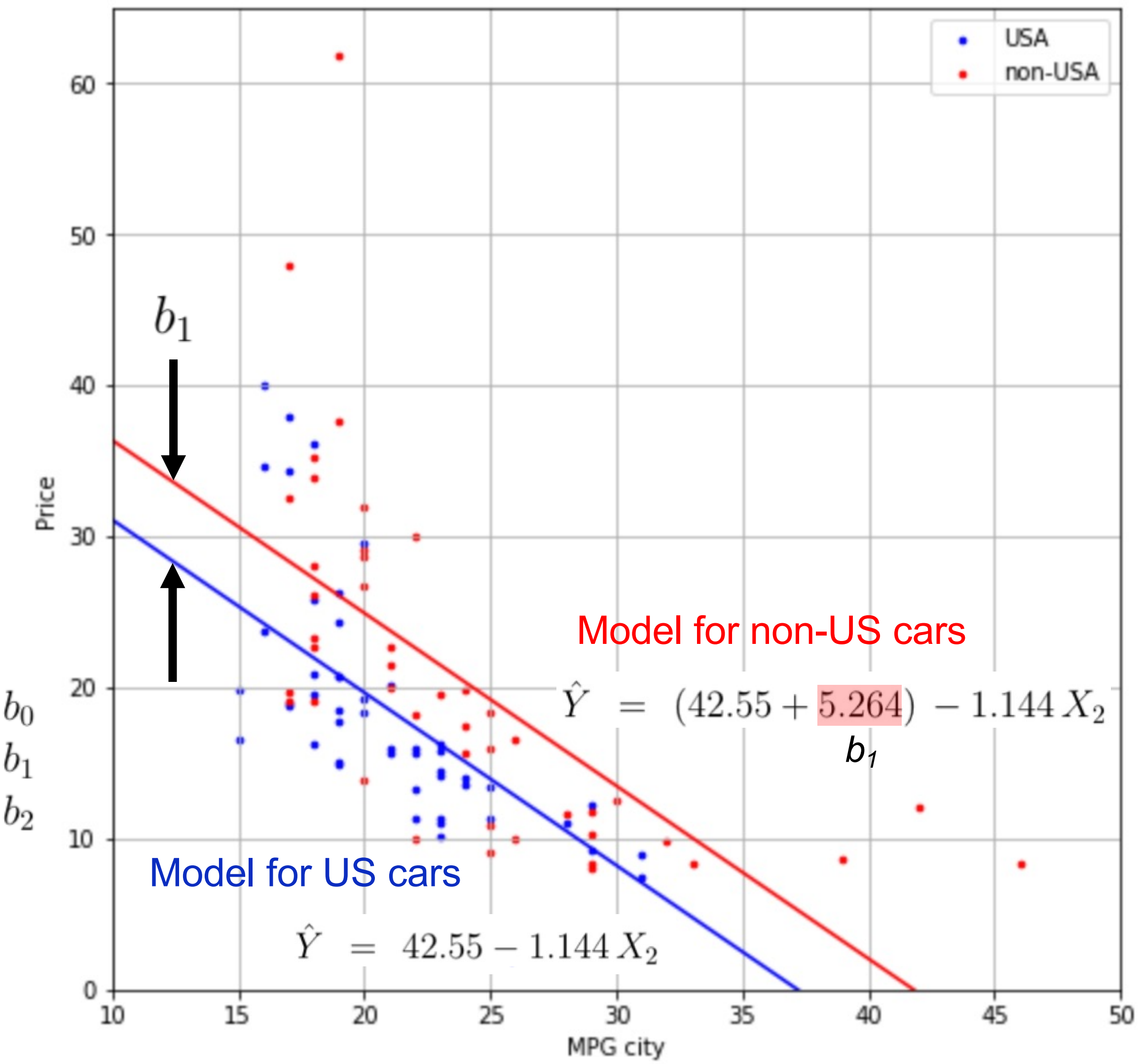


REGRESSION MODEL

b_1 is the difference in the average price between USA and non-USA cars (if X_2 is the same)

```
m1.params
```

Intercept	42.555991	b_0
Origin[T.non-USA]	5.264041	b_1
MPG_city	-1.144322	b_2

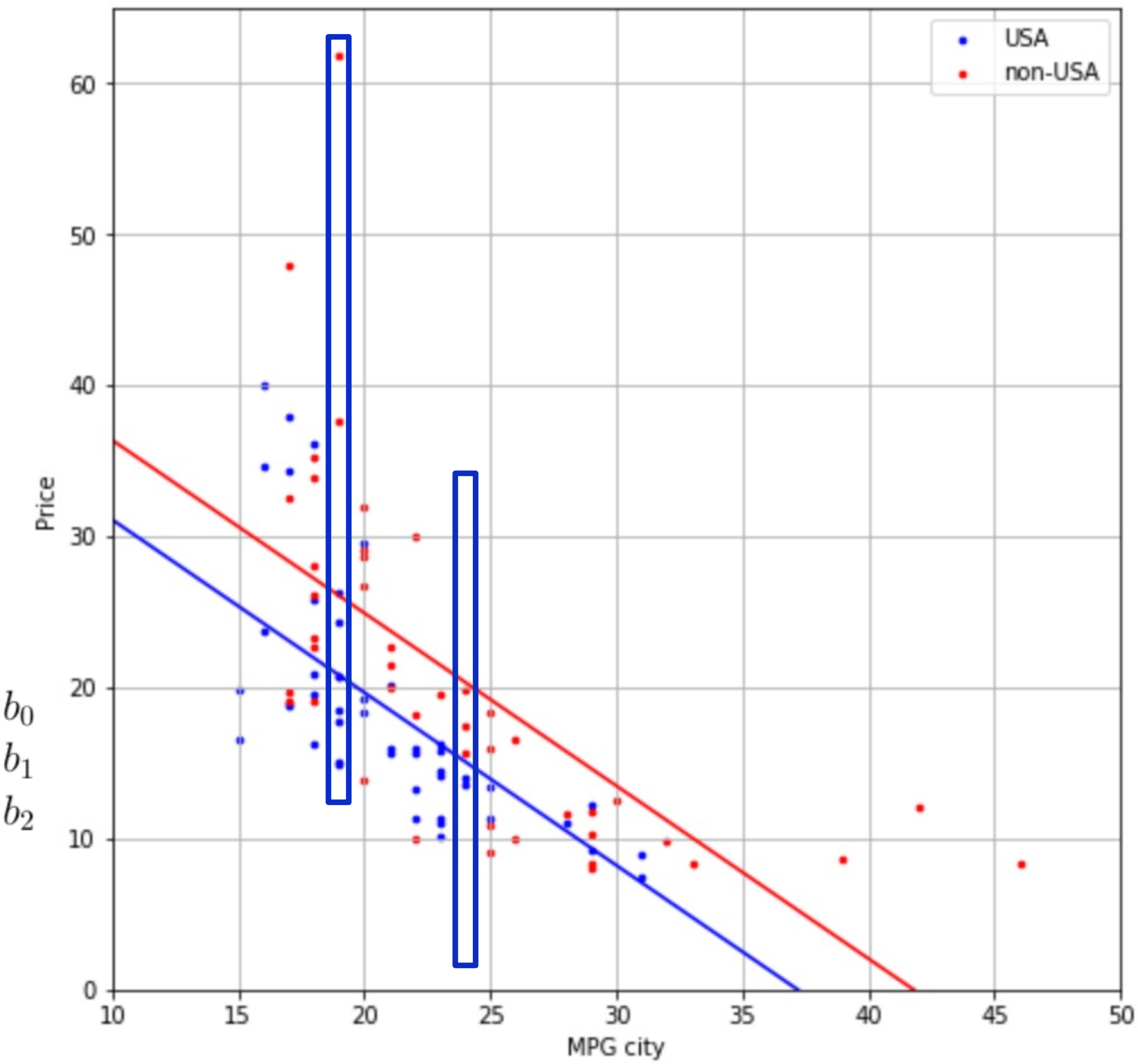


REGRESSION WITH A CATEGORICAL VARIABLE

non-USA cars are on average $5,264 more expensive when comparing cars with the same mileage

```
m1.params
```

Intercept	42.555991	b_0
Origin[T.non-USA]	5.264041	b_1
MPG_city	-1.144322	b_2

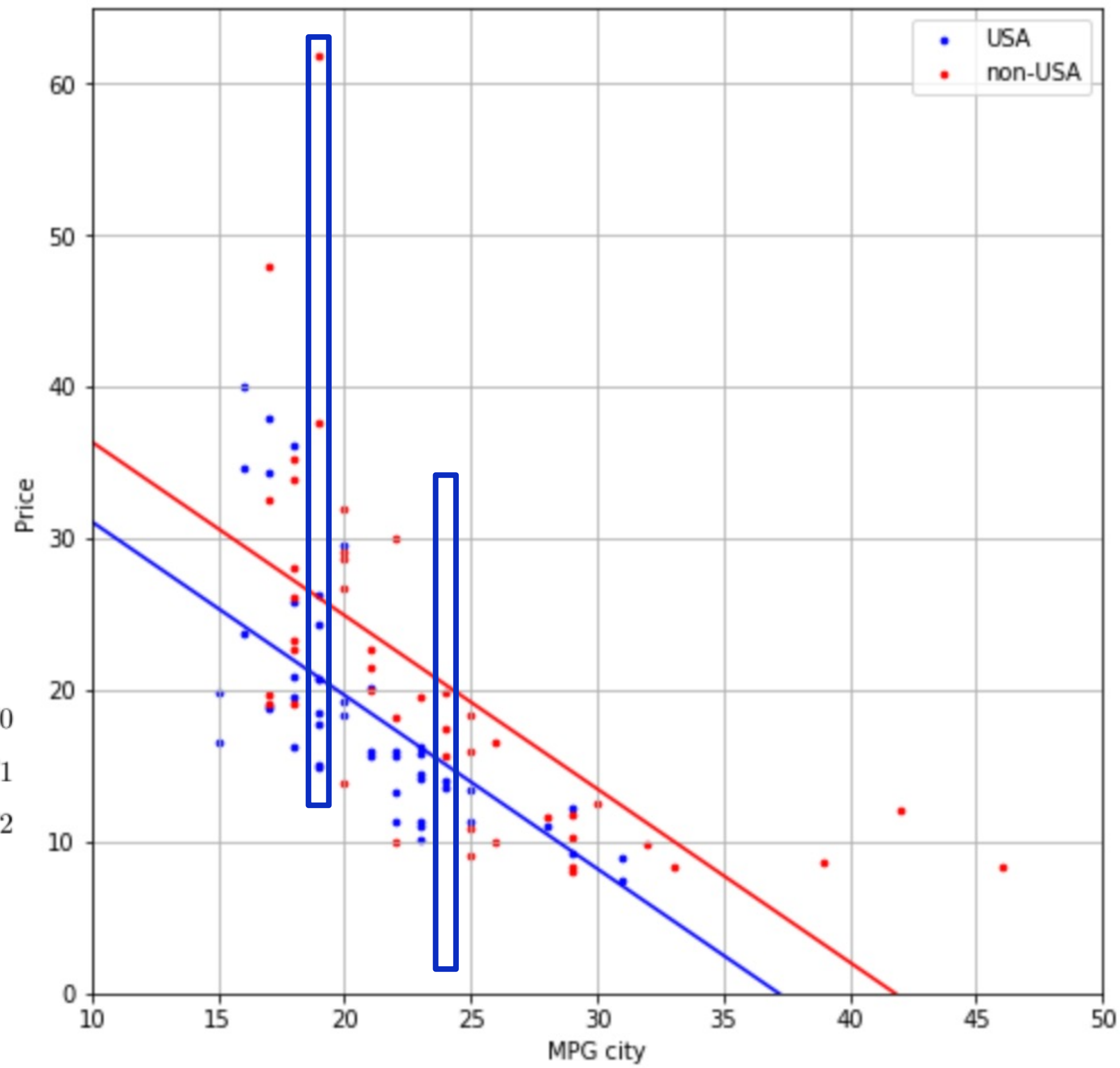


REGRESSION WITH A CATEGORICAL VARIABLE

non-USA cars are on average $5,264 more expensive when the effect of the other variable is removed

```
m1.params
```

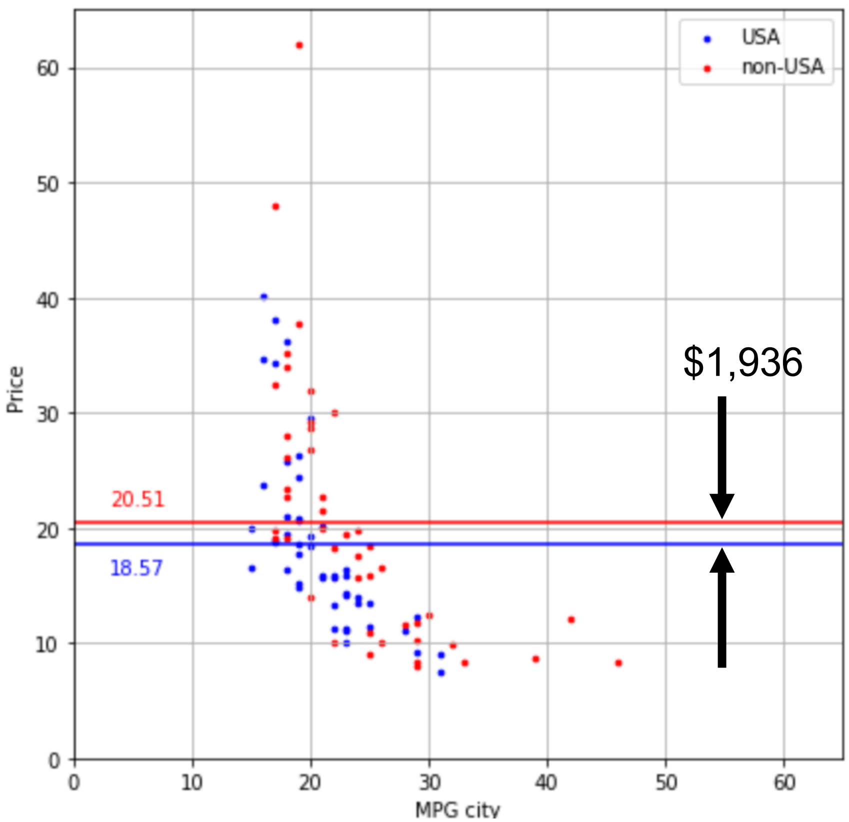
Intercept	42.555991	b_0
Origin[T.non-USA]	5.264041	b_1
MPG_city	-1.144322	b_2



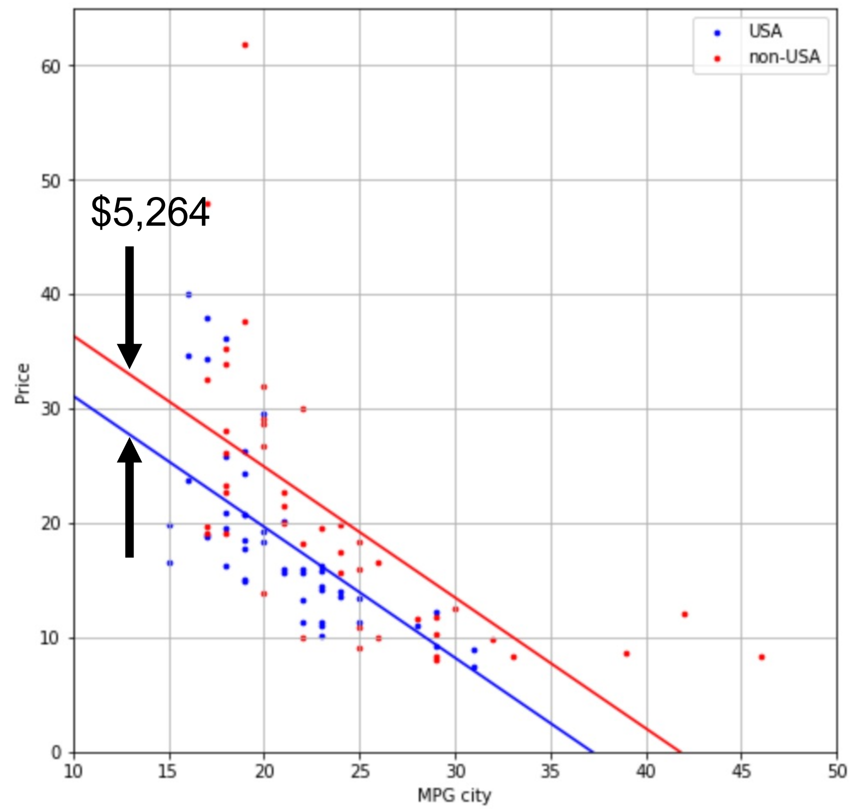
REGRESSION WITH A CATEGORICAL VARIABLE

How much more expensive are non-USA cars?

Pivot Table



Regression model



How much more expensive are non-USA cars?

- How much more expensive are non-USA cars than USA cars, irrespective of the mileage?

non-US cars are on average $1,936 more expensive

How much more expensive are non-USA cars?

- How much more expensive are non-USA cars than USA cars, irrespective of the mileage?

  non-US cars are on average $1,936 more expensive

- How much more expensive are non-USA cars than USA cars, having the same mileage?

  Comparing cars with the same mileage, non-US cars are on average $5,264 more expensive

REGRESSION WITH A CATEGORICAL VARIABLE

Comparing cars with the same mileage X_2, non-US cars are on average $5,264 more expensive

$$\hat{Y} = 42.55 - 1.144\, X_2$$

Model for US cars

$$\hat{Y} = (42.55 + 5.264) - 1.144\, X_2$$

Model for non-US cars

Price difference if both models use the same X_2

EXAMPLES

Regression with a Categorical Variable

with 3 categories

REGRESSION WITH A CATEGORICAL VARIABLE

Predict the Price of a car using MPG.city and AirBags

Y	numerical	categorical
Price	**MPG_city**	**AirBags**
14.9	19	No
20.7	19	Driver only
10.3	29	No
19.3	20	Driver & Passenger
26.3	19	Driver only
15.1	19	Driver & Passenger

REGRESSION WITH A CATEGORICAL VARIABLE

Price	MPG_city	AirBags
14.9	19	No
20.7	19	Driver only
10.3	29	No
19.3	20	Driver & Passenger
26.3	19	Driver only
15.1	19	Driver & Passenger

AirBags categories

- No airbags
- Driver only
- Driver and Passenger

REGRESSION WITH A CATEGORICAL VARIABLE

Price	MPG_city	AirBags
14.9	19	No
20.7	19	Driver only
10.3	29	No
19.3	20	Driver & Passenger
26.3	19	Driver only
15.1	19	Driver & Passenger

AirBags populations

- No airbags
- Driver only
- Driver and Passenger

Find an OLS line for each population

REGRESSION WITH A CATEGORICAL VARIABLE

Price	MPG_city	AirBags
14.9	19	No
20.7	19	Driver only
10.3	29	No
19.3	20	Driver & Passenger
26.3	19	Driver only
15.1	19	Driver & Passenger

How do we incorporate

AirBags into the model?

REGRESSION WITH A CATEGORICAL VARIABLE

	X_2	X_1
Price	MPG_city	AirBags
14.9	19	No
20.7	19	Driver only
10.3	29	No
19.3	20	Driver & Passenger
26.3	19	Driver only
15.1	19	Driver & Passenger

Use binary variables

- No airbags
- Driver only X_{11}
- Driver and Passenger X_{12}

REGRESSION WITH A CATEGORICAL VARIABLE (3 CATEGORIES)

Y: Price of the car

X_2 : MPG.city

X_1 : AirBags

Transform this model (where X_1 has 3 categories and X_2 is numerical)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

into

this new model

$$Y = \beta_0 + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_2 X_2 + \epsilon$$

REGRESSION WITH A CATEGORICAL VARIABLE (3 CATEGORIES)

Y: Price of the car

X_2 : MPG.city

X_1 : AirBags

Transform this model

3 categories → $\beta_1 X_1$; numerical → $\beta_2 X_2$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

into

this new model

$$Y = \beta_0 + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_2 X_2 + \epsilon$$

REGRESSION WITH A CATEGORICAL VARIABLE

Transform this model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$

into this new model

$$Y = \beta_0 + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_2 X_2 + \epsilon$$

$$x_{11} = \begin{cases} 1 & \text{if car has driver only airbag} \\ 0 & \text{ow} \end{cases}$$

$$x_{12} = \begin{cases} 1 & \text{if car has driver and passenger airbags} \\ 0 & \text{ow} \end{cases}$$

REGRESSION WITH A CATEGORICAL VARIABLE

New Model $Y = \beta_0 + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_2 X_2 + \epsilon$

$$x_{11} = \begin{cases} 1 & \text{if car has driver only airbag} \\ 0 & \text{ow} \end{cases}$$

$$x_{12} = \begin{cases} 1 & \text{if car has driver and passenger airbags} \\ 0 & \text{ow} \end{cases}$$

Becomes three models

$$Y = \beta_0 + \beta_2 X_2 + \epsilon \qquad (x_{11} = x_{12} = 0)$$

← base model

is defined when all binary variables are set equal to 0

$$Y = (\beta_0 + \beta_{11}) + \beta_2 X_2 + \epsilon \qquad (x_{11} = 1, x_{12} = 0)$$

$$Y = (\beta_0 + \beta_{12}) + \beta_2 X_2 + \epsilon \qquad (x_{11} = 0, x_{12} = 1)$$

REGRESSION WITH A CATEGORICAL VARIABLE

Model

$$Y = \beta_0 + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_2 X_2 + \epsilon$$

$$x_{11} = \begin{cases} 1 & \text{if car has driver only airbag} \\ 0 & \text{ow} \end{cases}$$

$$x_{12} = \begin{cases} 1 & \text{if car has driver and passenger airbags} \\ 0 & \text{ow} \end{cases}$$

Three OLS lines

Model for cars with

- No AirBags $\hat{Y} = b_0 + b_2 X_2$ $(x_{11} = x_{12} = 0)$

base category is "no airbag"

- Driver Only airbags $\hat{Y} = (b_0 + b_{11}) + b_2 X_2$ $(x_{11} = 1, x_{12} = 0)$
- Driver & Passenger $\hat{Y} = (b_0 + b_{12}) + b_2 X_2$ $(x_{11} = 0, x_{12} = 1)$

additional intercepts

REGRESSION WITH A CATEGORICAL VARIABLE

Price	MPG_city	AirBags
14.9	19	No
20.7	19	Driver only
10.3	29	No
19.3	20	Driver & Passenger
26.3	19	Driver only
15.1	19	Driver & Passenger

category *labels*

- No
- Driver only
- Driver & Passenger

REGRESSION WITH A CATEGORICAL VARIABLE

select “car with No airbag” as base category

```
m2 = smf.ols(formula = 'Price~MPG_city + C(AirBags,Treatment(reference ="No"))',
             data = df).fit()
```

REGRESSION WITH A CATEGORICAL VARIABLE – PARAMETERS

```
m2 = smf.ols(formula = 'Price~MPG_city + C(AirBags,Treatment(reference ="No"))',
             data = df).fit()
m2.params
```

```
Intercept                                                32.536873
C(AirBags, Treatment(reference="No"))[T.Driver & Passenger]   11.219026   ← Add. intercept
C(AirBags, Treatment(reference="No"))[T.Driver only]     5.698106   ← Add. intercept
MPG_city                                                 -0.786564  ← slope
```

REGRESSION WITH A CATEGORICAL VARIABLE – 3 Models

```
m2 = smf.ols(formula = 'Price~MPG_city + C(AirBags,Treatment(reference ="No"))',
             data = df).fit()
m2.params
```

```
Intercept                                                   32.536873
C(AirBags, Treatment(reference="No"))[T.Driver & Passenger]  11.219026
C(AirBags, Treatment(reference="No"))[T.Driver only]          5.698106
MPG_city                                                     -0.786564
```

base model

$$\hat{Y} = 32.53 - 0.786\, X_2$$

No Airbag

REGRESSION WITH A CATEGORICAL VARIABLE

```
m2 = smf.ols(formula = 'Price~MPG_city + C(AirBags,Treatment(reference ="No"))',
             data = df).fit()
m2.params
```

```
Intercept                                                       32.536873
C(AirBags, Treatment(reference="No"))[T.Driver & Passenger]    11.219026
C(AirBags, Treatment(reference="No"))[T.Driver only]            5.698106
MPG_city                                                        -0.786564
```

$$\hat{Y} = 32.53 - 0.786\, X_2$$

No Airbag

$$\hat{Y} = (32.53 + 5.69) - 0.786\, X_2$$

Driver only

REGRESSION WITH A CATEGORICAL VARIABLE

```
m2 = smf.ols(formula = 'Price~MPG_city + C(AirBags,Treatment(reference ="No"))',
             data = df).fit()
m2.params
```

```
Intercept                                                    32.536873
C(AirBags, Treatment(reference="No"))[T.Driver & Passenger]  11.219026
C(AirBags, Treatment(reference="No"))[T.Driver only]          5.698106
MPG_city                                                     -0.786564
```

$$\hat{Y} = 32.53 - 0.786\,X_2$$ No Airbag

$$\hat{Y} = (32.53 + 5.69) - 0.786\,X_2$$ Driver only

$$\hat{Y} = (32.53 + 11.21) - 0.786\,X_2$$ Driver & Passenger

EXAMPLE 2

How much more expensive are

cars with airbags?

REGRESSION WITH A CATEGORICAL VARIABLE – PIVOT TABLE

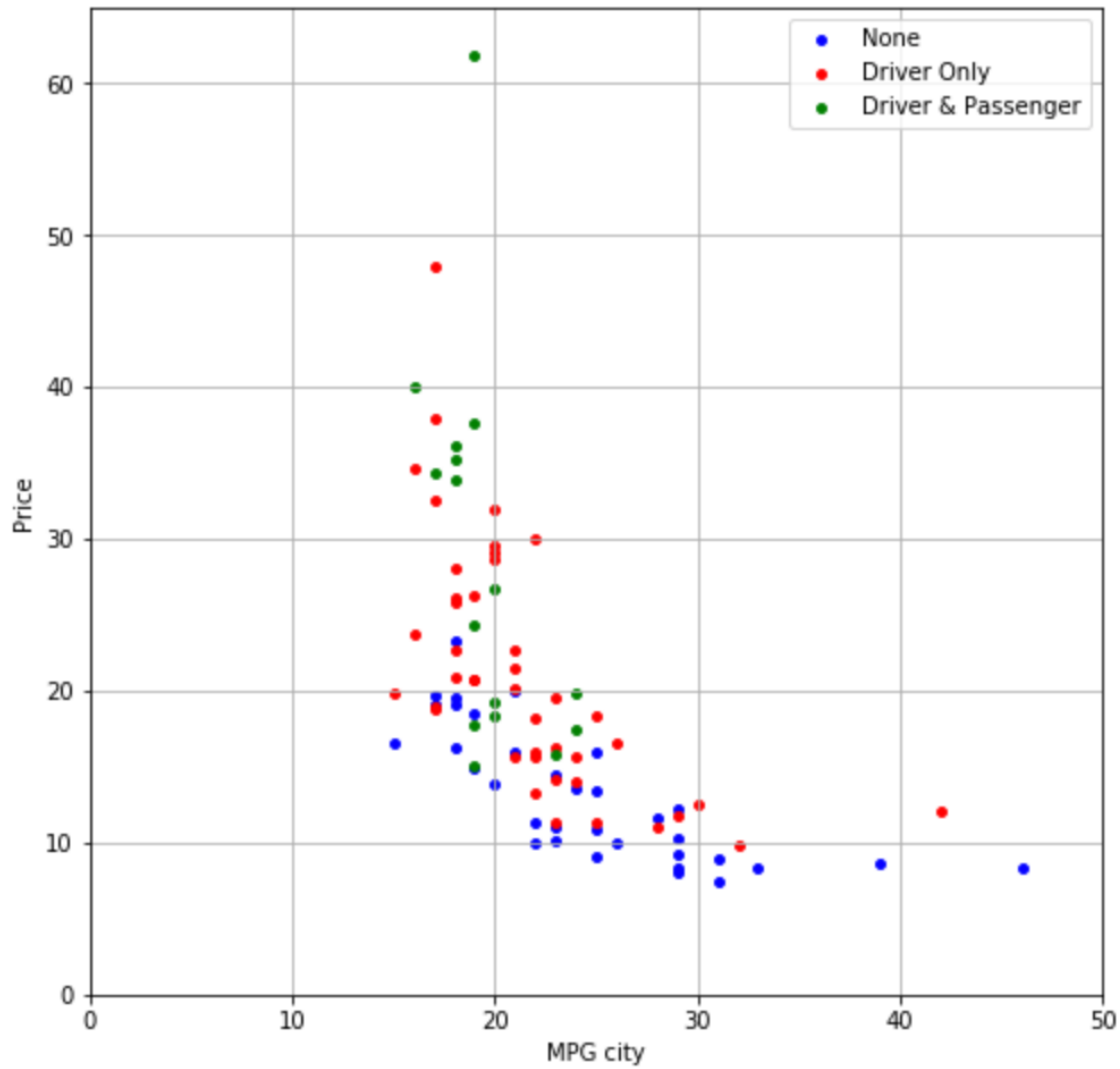
cars with airbags
are on average
more expensive

```
df.pivot_table(values = 'Price',index = 'AirBags')
```

AirBags	Price
Driver & Passenger	28.368750
Driver only	21.223256
None	13.173529

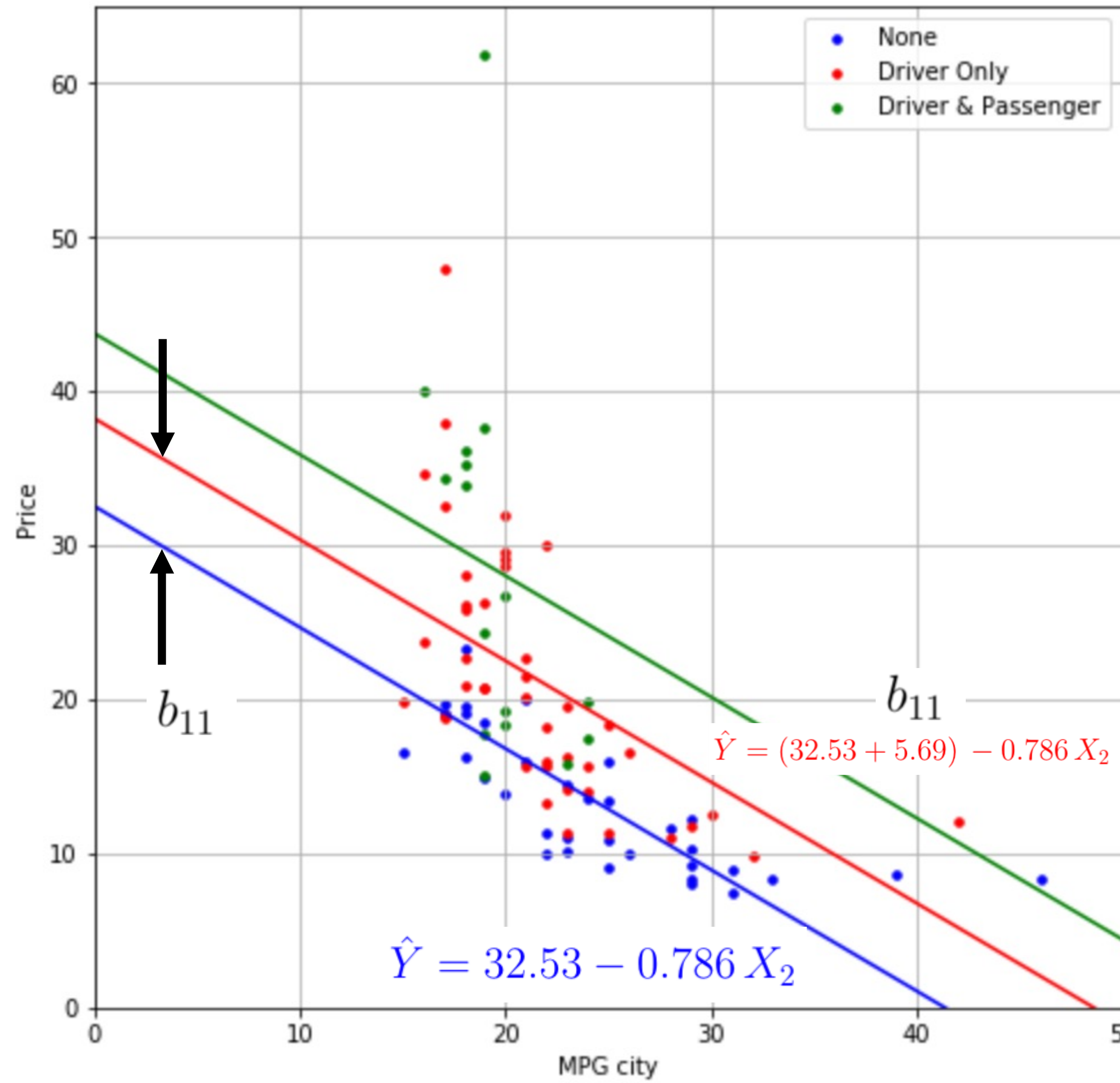
REGRESSION WITH A CATEGORICAL VARIABLE

cars with airbags are on average more expensive



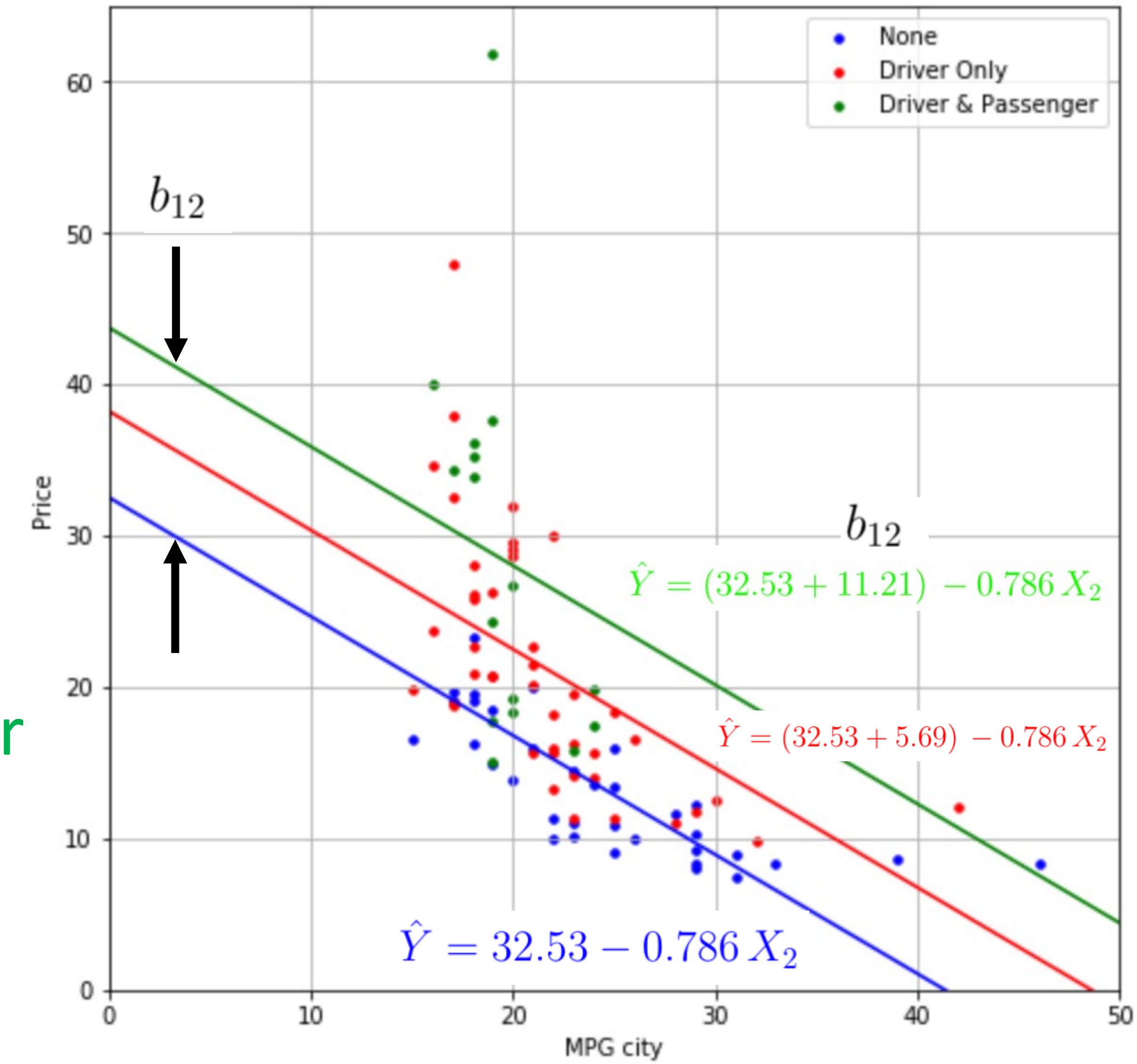
REGRESSION WITH A CATEGORICAL VARIABLE

b_{11} is the difference in the average price between cars with No Airbags (base) and cars with Driver Only airbag



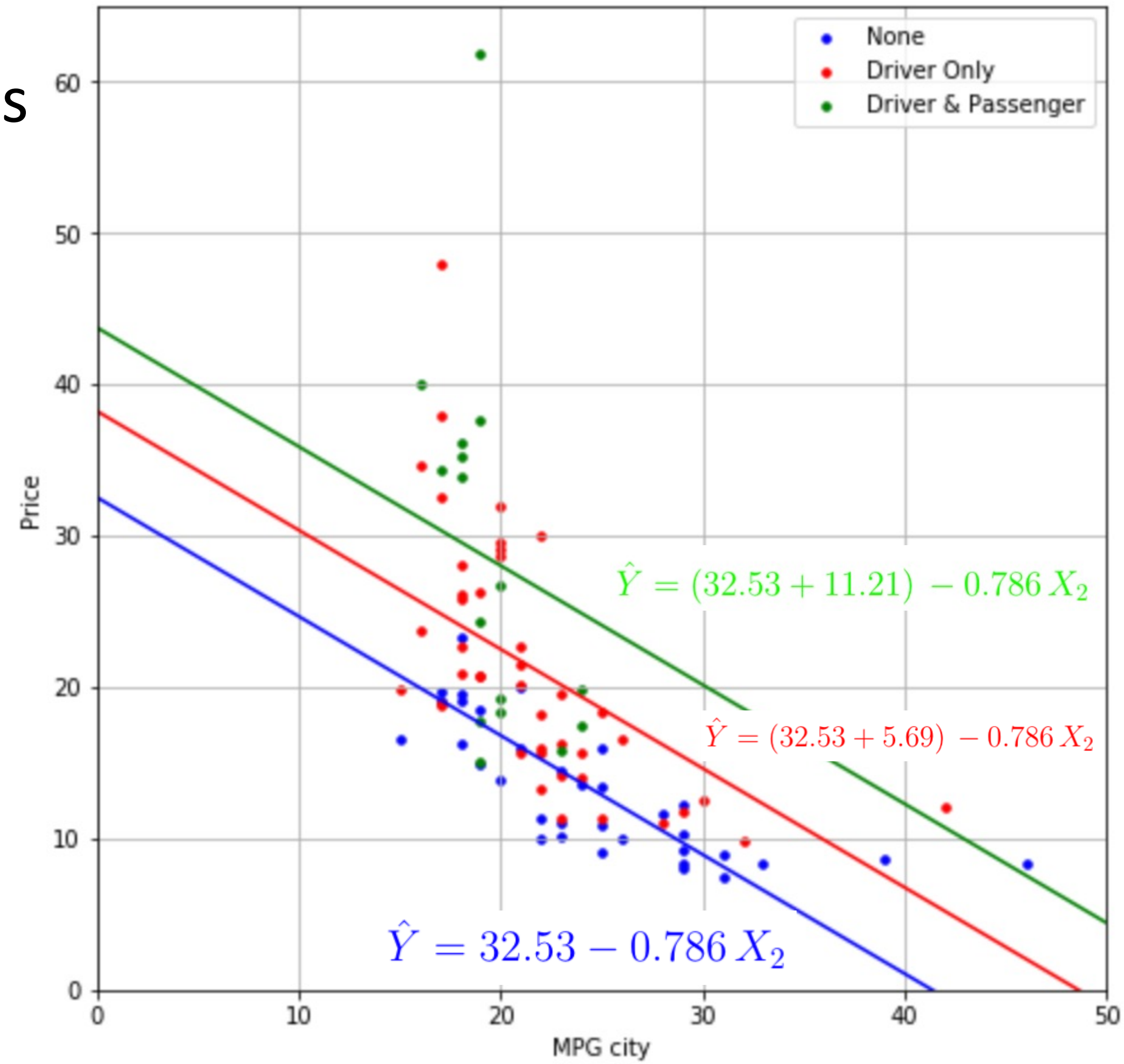
REGRESSION WITH A CATEGORICAL VARIABLE

b_{12} is the difference in the average price between cars with No Airbags (base) and cars with Driver and Passenger airbags



REGRESSION WITH A CATEGORICAL VARIABLE

How much expensive is a car with Driver and Passenger airbags than a car with No Airbags (base)?



REGRESSION WITH A CATEGORICAL VARIABLE

How much expensive is a car with

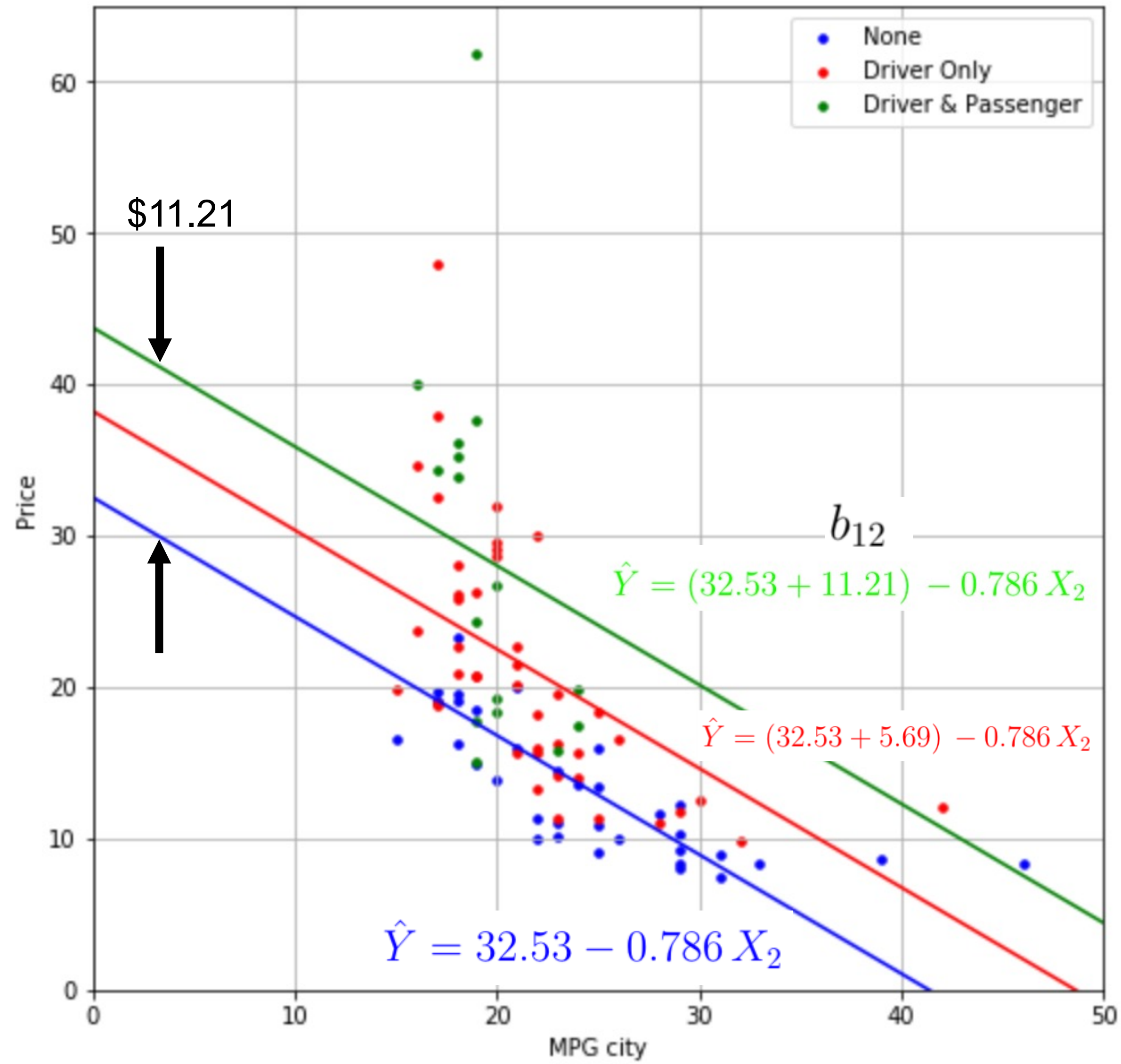
Driver and Passenger airbags

than a car with

No Airbags (base)?

On average, it is $11,210 dollars more expensive,

if the cars have same Mileage



REGRESSION WITH A CATEGORICAL VARIABLE

How much expensive is a car with

Driver and Passenger

airbags

than a car with

No Airbags (base)?

On average, it is $15,190 dollars more expensive, irrespective of the cars Mileage

	Price
AirBags	
Driver & Passenger	28.368750
Driver only	21.223256
None	13.173529
	15.195221

Regression with **interaction** between predictors

- What is a predictor's effect on Y?
- What is interaction?

OVERVIEW

- The effect of predictor X_1 on Y is the average amount Y changes when X_1 increases by one unit
- The effect of predictor X_1 on Y is estimated by the regression coefficient of X_1 in the regression model

OVERVIEW

- The effect of predictor X_1 on Y is the average amount Y changes when X_1 increases by one unit
- The effect of predictor X_1 on Y is estimated by the regression coefficient of X_1 in the regression model

- Interaction occurs when the effect of a predictor X_2 on Y depends on the value or category of another predictor X_1
- The interaction of X_1 and X_2 is estimated by the regression coefficient of the term X_1X_2 in the model

EFFECTS OF X ON Y – NO INTERACTION

- The effect of one predictor on the response Y is given by the slope

OLS Regression Results

Dep. Variable:	MPG.city	**R-squared:**	0.732
Model:	OLS	**Adj. R-squared:**	0.713
Method:	Least Squares	**F-statistic:**	38.61
Date:	Fri, 18 Sep 2020	**Prob (F-statistic):**	2.79e-22
Time:	18:57:08	**Log-Likelihood:**	-228.40
No. Observations:	92	**AIC:**	470.8
Df Residuals:	85	**BIC:**	488.4
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	36.9200	7.294	5.062	0.000	22.417	51.423
x1	0.1015	0.570	0.178	0.859	-1.031	1.234
x2	0.8743	1.076	0.813	0.419	-1.264	3.013
x3	-0.0303	0.023	-1.344	0.183	-0.075	0.015
x4	0.0016	0.001	1.418	0.160	-0.001	0.004
x5	-0.2385	0.540	-0.441	0.660	-1.313	0.836
x6	-0.0066	0.002	-4.006	0.000	-0.010	-0.003

slopes

EFFECT OF X_1 ON Y – NO INTERACTION

- The effect of X_1 on the response Y is given by the slope of X_1
- If X_1 increases by one unit then Y increases by 0.1015, on average
- all other variables held constant

OLS Regression Results

Dep. Variable:	MPG.city	**R-squared:**	0.732
Model:	OLS	**Adj. R-squared:**	0.713
Method:	Least Squares	**F-statistic:**	38.61
Date:	Fri, 18 Sep 2020	**Prob (F-statistic):**	2.79e-22
Time:	18:57:08	**Log-Likelihood:**	-228.40
No. Observations:	92	**AIC:**	470.8
Df Residuals:	85	**BIC:**	488.4
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	36.9200	7.294	5.062	0.000	22.417	51.423
x1	0.1015	0.570	0.178	0.859	-1.031	1.234
x2	0.8743	1.076	0.813	0.419	-1.264	3.013
x3	-0.0303	0.023	-1.344	0.183	-0.075	0.015
x4	0.0016	0.001	1.418	0.160	-0.001	0.004
x5	-0.2385	0.540	-0.441	0.660	-1.313	0.836
x6	-0.0066	0.002	-4.006	0.000	-0.010	-0.003

effect of X_1 on Y

EFFECT OF X_5 ON Y – NO INTERACTION

- The effect of X_5 on the response Y is given by the slope of X_5
- If X_5 increases by one unit then Y decreases by 0.2385, on average
- all other variables held constant

OLS Regression Results

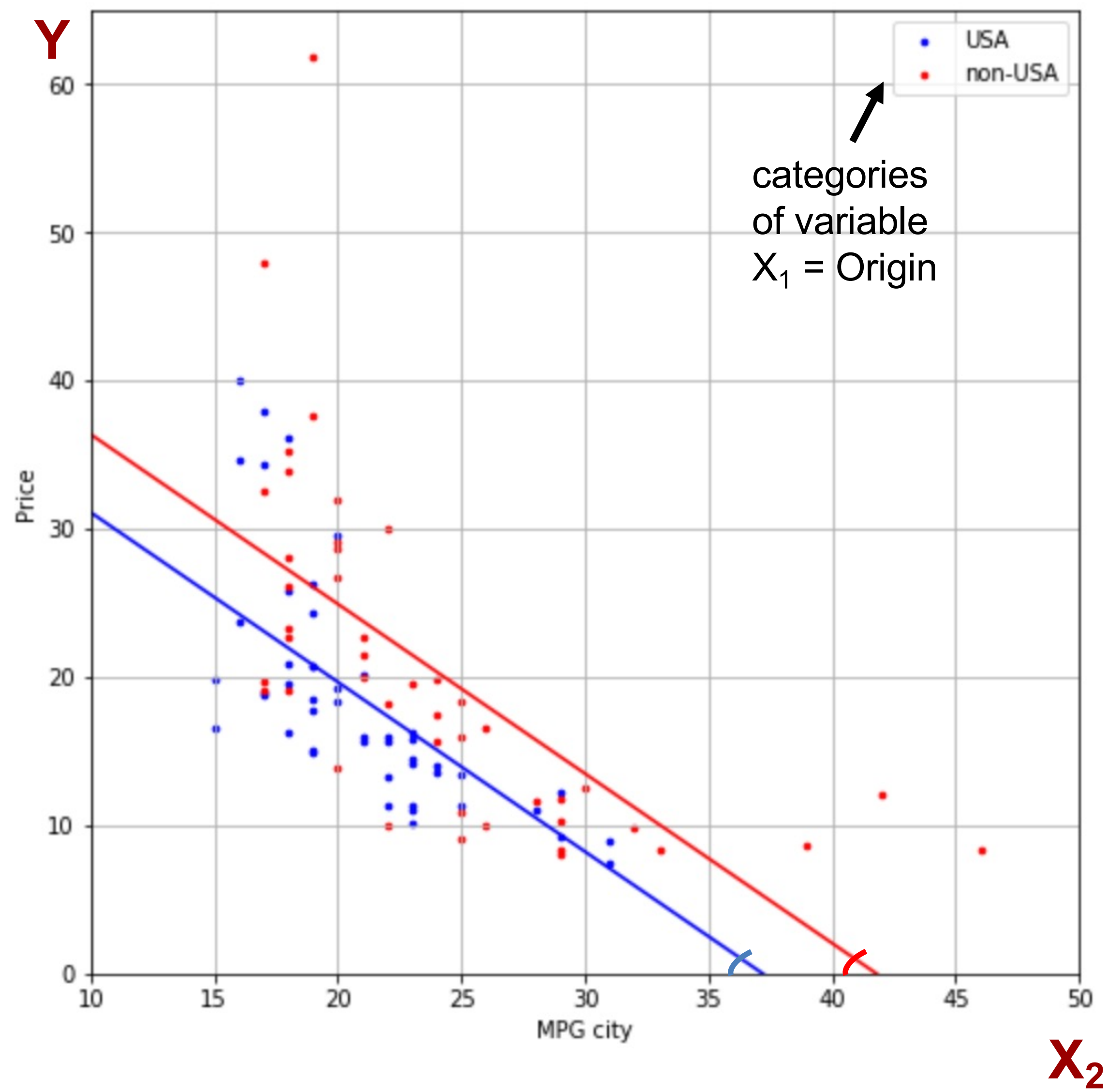
Dep. Variable:	MPG.city	R-squared:	0.732
Model:	OLS	Adj. R-squared:	0.713
Method:	Least Squares	F-statistic:	38.61
Date:	Fri, 18 Sep 2020	Prob (F-statistic):	2.79e-22
Time:	18:57:08	Log-Likelihood:	-228.40
No. Observations:	92	AIC:	470.8
Df Residuals:	85	BIC:	488.4
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	36.9200	7.294	5.062	0.000	22.417	51.423
x1	0.1015	0.570	0.178	0.859	-1.031	1.234
x2	0.8743	1.076	0.813	0.419	-1.264	3.013
x3	-0.0303	0.023	-1.344	0.183	-0.075	0.015
x4	0.0016	0.001	1.418	0.160	-0.001	0.004
x5	-0.2385	0.540	-0.441	0.660	-1.313	0.836
x6	-0.0066	0.002	-4.006	0.000	-0.010	-0.003

effect of X_5 on Y

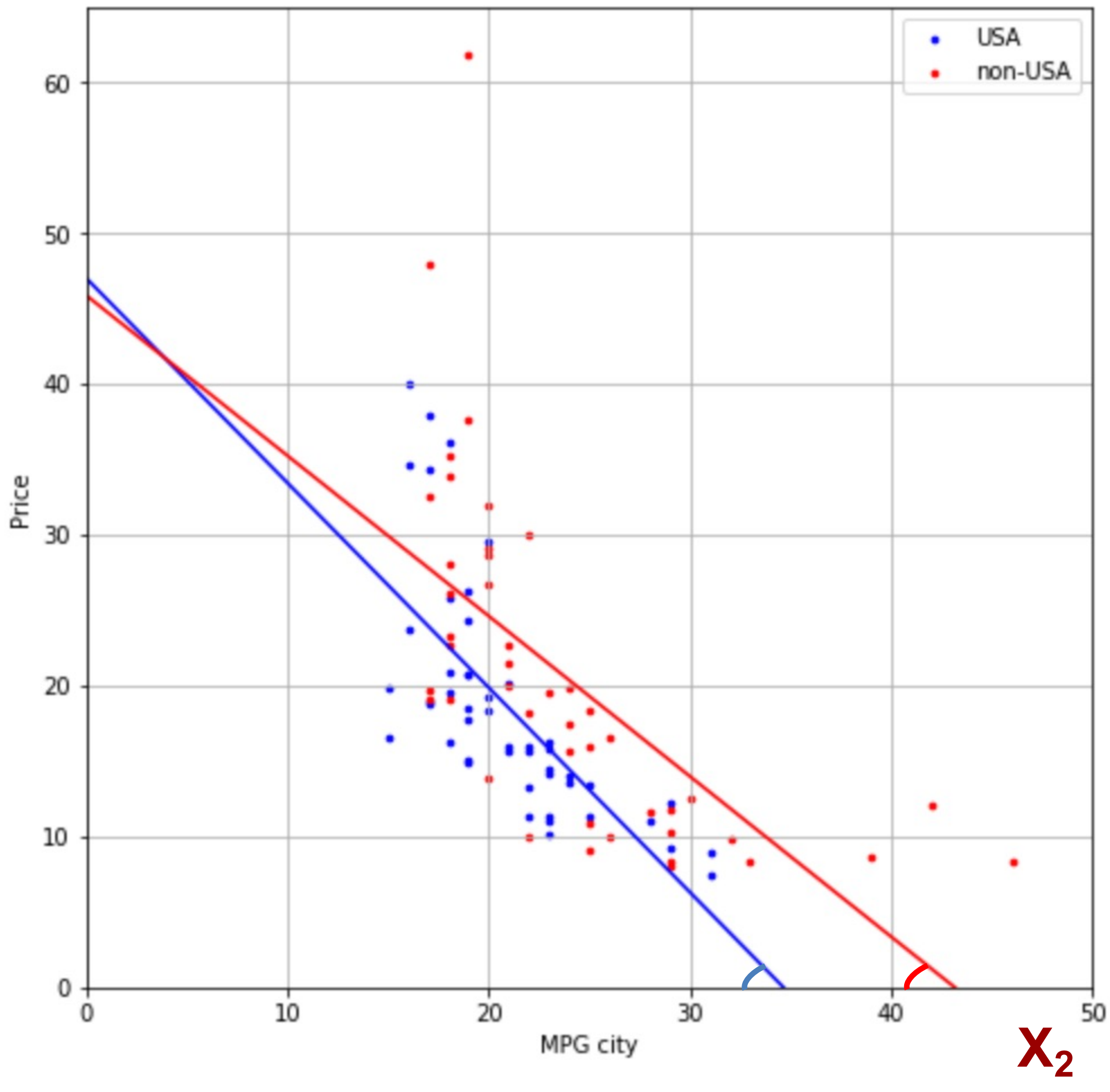
NO INTERACTION MODEL

- Y = Price decreases with X_2 = MPG.city
- The effect of X_2 = MPG.city on Y = Price, is given by the coeff. of X_2 (slope)
- The slope is the same for all categories of X_1 = Origin
- No interaction between X_1 = Origin with X_2 = MPG.city



MODEL WITH INTERACTION

- Price decreases with MPG.city
- Different categories of Origin result in different slopes
- Price decreases faster on non-US cars
- The effect of predictor MPG.city on Price depends on the category of Origin



INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

Model with interaction

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

X_1 = Origin

X_2 = MPG.city

INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

Model with interaction

interaction term

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

X_1 = Origin

X_2 = MPG.city

INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

Model

$$Y = \beta_0 + \beta_1 \underline{X_1} + \beta_2 X_2 + \beta_{12} \underline{X_1} X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

Two OLS lines

$$\hat{Y} = b_0 + b_2 X_2 \qquad (x_1 = 0)$$ base model

INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

Model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \epsilon$$

$$x_1 = \begin{cases} 0 & \text{if USA car} \\ 1 & \text{if non-USA car} \end{cases}$$

Two OLS lines

$$\hat{Y} = b_0 + b_2 X_2 \qquad (x_1 = 0)$$

← model for US cars

$$\hat{Y} = (b_0 + b_1) + (b_2 + b_{12}) X_2 \qquad (x_1 = 1)$$

← model for non-US cars

additional intercept ↑ (b_1) additional slope ↑ (b_{12})

MODEL WITH NO INTERACTION

Fit Model

```
m1 = smf.ols(formula = 'Price~MPG_city + Origin',data = df).fit()
m1.params
```

```
Intercept               42.555991
Origin[T.non-USA]        5.264041
MPG_city                -1.144322
```

b_0

b_1 additional intercept

b_2

Two OLS lines

$$\hat{Y} = 42.55 - 1.144\, X_2$$

← model for US cars

$$\hat{Y} = (42.55 + 5.264) - 1.144\, X_2$$

← model for non-US cars

additional intercept

MODEL WITH INTERACTION

Fit Model

```
m3 = smf.ols(formula = 'Price ~ MPG_city * Origin',
             data = df).fit()
m3.params
```

Intercept	47.011062	b_0	
Origin[T.non-USA]	-1.132917	b_1	additional intercept
MPG_city	-1.356890	b_2	
MPG_city:Origin[T.non-USA]	0.293932	b_{12}	additional slope

Two OLS lines

- For US cars $$\hat{Y} = 47.01 - 1.357\, X_2 \qquad (x_1 = 0)$$
- For non-US cars $$\hat{Y} = (47.01 - 1.1329) + (-1.357 + 0.2939)\, X_2 \qquad (x_1 = 1)$$

MODEL WITH INTERACTION

Fit Model

```
m3 = smf.ols(formula = 'Price ~ MPG_city * Origin',
             data = df).fit()
m3.params
```

```
Intercept                          47.011062
Origin[T.non-USA]                  -1.132917
MPG_city                           -1.356890
MPG_city:Origin[T.non-USA]          0.293932
```

- b_0
- b_1 additional intercept
- b_2
- b_{12} additional slope

Two OLS lines

- For US cars $\hat{Y} = 47.01 - 1.357\, X_2 \qquad (x_1 = 0)$
- For non-US cars $\hat{Y} = (47.01 - 1.1329) + (-1.357 + 0.2939)\, X_2 \qquad (x_1 = 1)$

additional intercept

additional slope

INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

Fit Model

```
m3 = smf.ols(formula = 'Price ~ MPG_city * Origin',
             data = df).fit()
m3.params
```

```
Intercept                      47.011062
Origin[T.non-USA]              -1.132917
MPG_city                       -1.356890
MPG_city:Origin[T.non-USA]      0.293932
```

base model

Two OLS lines

- For US cars $\hat{Y} = 47.01 - 1.357\, X_2 \qquad (x_1 = 0)$
- For non-US cars $\hat{Y} = (47.01 - 1.1329) + (-1.357 + 0.2939)\, X_2 \qquad (x_1 = 1)$

INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

Fit Model

```
m3 = smf.ols(formula = 'Price ~ MPG_city * Origin',
             data = df).fit()
m3.params
```

```
Intercept                      47.011062
Origin[T.non-USA]              -1.132917
MPG_city                       -1.356890
MPG_city:Origin[T.non-USA]      0.293932
```

non-base model

Two OLS lines

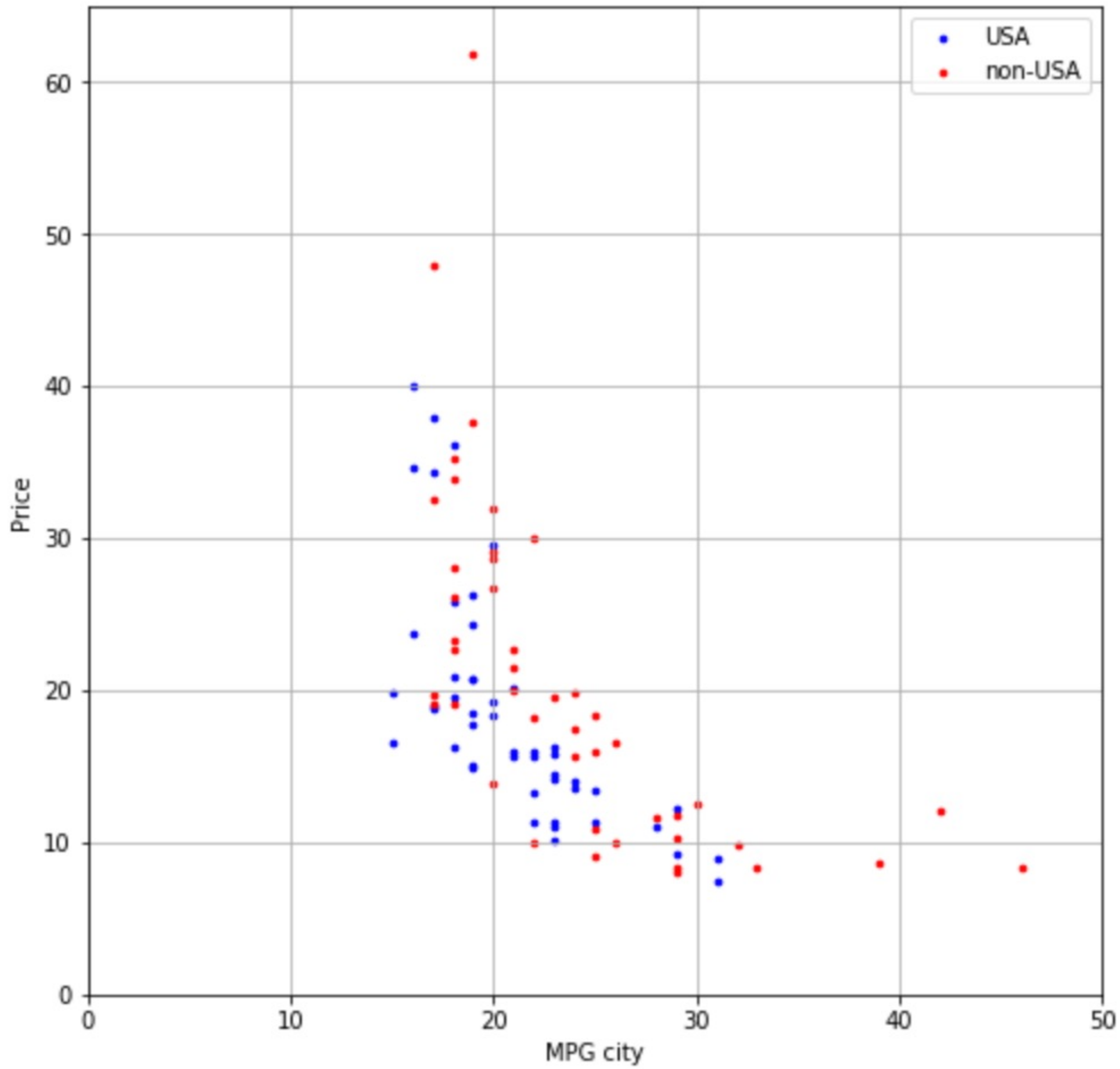
- For US cars $\hat{Y} = 47.01 - 1.357\, X_2 \quad (x_1 = 0)$
- For non-US cars $\hat{Y} = (47.01 - 1.1329) + (-1.357 + 0.2939)\, X_2 \quad (x_1 = 1)$

additional intercept

additional slope

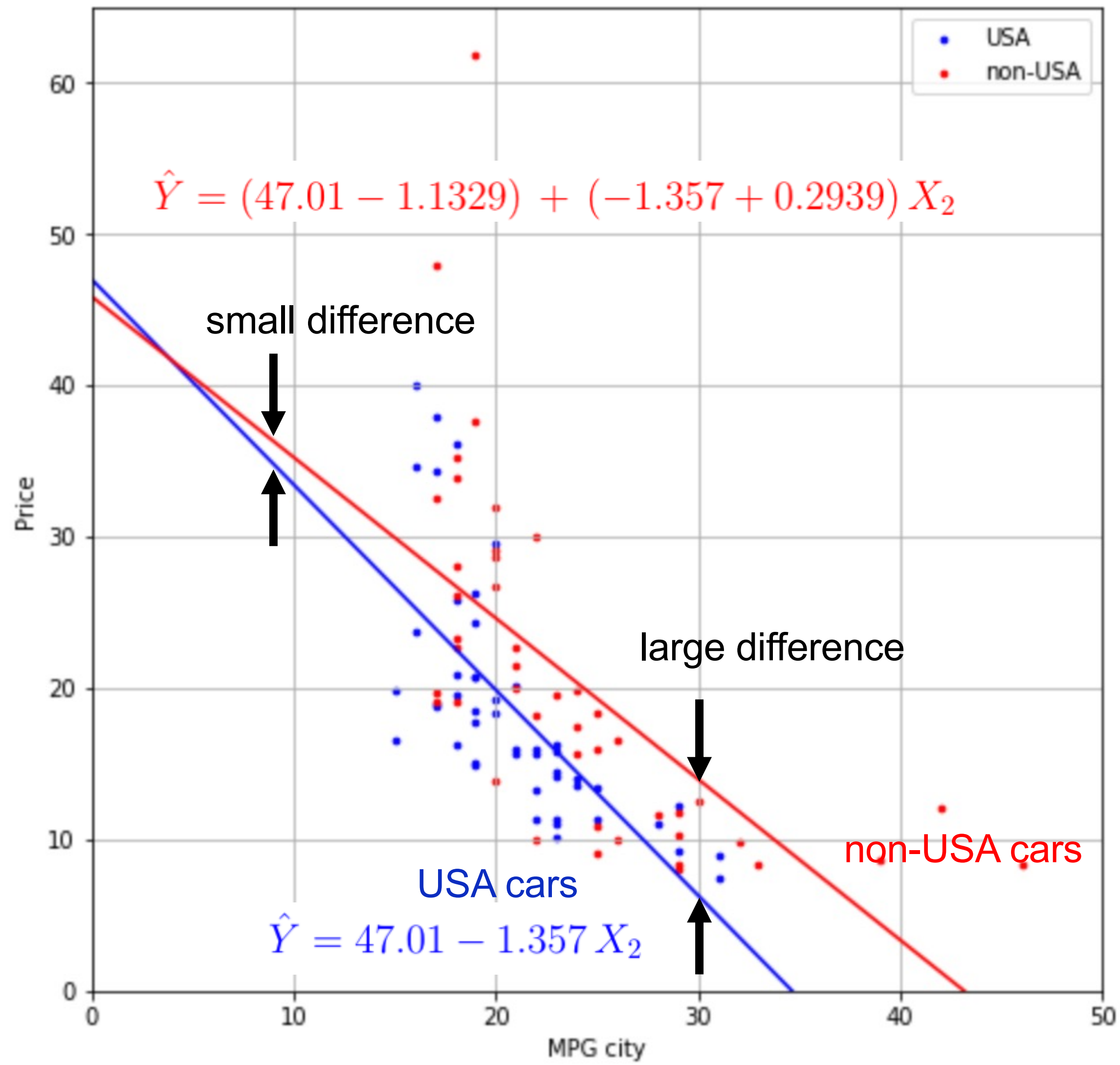
INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

non-USA cars are
on average
more expensive



INTERACTION BETWEEN NUMERIC VARIABLE X_2 AND A CATEGORICAL VARIABLE X_1

The difference in the average price (between USA and non-USA cars) changes as mileage increases



Encoding Methods

Example 1

Categorical Predictors – EXAMPLE

Consider the following dataset

X_1	X_2	Y
S	-0.10	19.19
S	2.53	22.74
S	4.86	23.91
M	0.26	7.07
M	2.55	7.93
M	4.87	8.93
L	0.08	20.63
L	2.62	23.46
L	5.09	25.75

$n = 9$
$p = 2$

```
data0 = pd.read_csv('small.csv')
data0
```

	X1	X2	Y
0	S	-0.10	19.19
1	S	2.53	22.74
2	S	4.86	23.91
3	M	0.26	7.07
4	M	2.55	7.93
5	M	4.87	8.93
6	L	0.08	20.63
7	L	2.62	23.46
8	L	5.09	25.75

Categorical Predictors – EXAMPLE

Consider the following dataset

X_1	X_2	Y
S	-0.10	19.19
S	2.53	22.74
S	4.86	23.91
M	0.26	7.07
M	2.55	7.93
M	4.87	8.93
L	0.08	20.63
L	2.62	23.46
L	5.09	25.75

label encoding →

X_1	X_2	Y
0	-0.10	19.19
0	2.53	22.74
0	4.86	23.91
1	0.26	7.07
1	2.55	7.93
1	4.87	8.93
2	0.08	20.63
2	2.62	23.46
2	5.09	25.75

Categorical Predictors – LABEL ENCODING

```
data1['X1']=data1['X1'].replace(('S','M','L'),
                                (0,1,2))

data1
```

	X1	X2	Y
0	0	-0.10	19.19
1	0	2.53	22.74
2	0	4.86	23.91
3	1	0.26	7.07
4	1	2.55	7.93
5	1	4.87	8.93
6	2	0.08	20.63
7	2	2.62	23.46
8	2	5.09	25.75

```
# Split response, predictors

X, y = data1[['X1','X2']], data1.Y
X
```

	X1	X2
0	0	-0.10
1	0	2.53
2	0	4.86
3	1	0.26
4	1	2.55
5	1	4.87
6	2	0.08
7	2	2.62
8	2	5.09

```
y

0    19.19
1    22.74
2    23.91
3     7.07
4     7.93
5     8.93
6    20.63
7    23.46
8    25.75
```

Get the Regression Model and the results

```
m1 = LinearRegression().fit(X,y)
```

```
m1.intercept_
```

```
15.167783009625168
```

```
m1.coef_
```

```
array([0.60192355, 0.77691744])
```

```
# Least Squares plane
```

```
# Yhat = 15.167 + 0.602 X1 + 0.777 X2
```

Get the Regression Model and the results

```
m1 = LinearRegression().fit(X,y)
```

```
m1.intercept_
```

```
15.167783009625168
```

```
m1.coef_
```

```
array([0.60192355, 0.77691744])
```

```
# Least Squares plane
```

```
# Yhat = 15.167 + 0.602 X1 + 0.777 X2
```

```
# R-squared
```

```
R2 = m1.score(X,y)
R2
```

```
0.052592593041448255
```

very small

```
# number of rows, number of predictors
```

```
n, p = 9, 2
```

```
# adj R-squared
```

```
print (1 - (1-R2)*(n-1)/(n-p-1))
```

```
-0.26320987594473566
```

negative!

Library statsmodels

```
data1
```

	X1	X2	Y
0	0	-0.10	19.19
1	0	2.53	22.74
2	0	4.86	23.91
3	1	0.26	7.07
4	1	2.55	7.93
5	1	4.87	8.93
6	2	0.08	20.63
7	2	2.62	23.46
8	2	5.09	25.75

```
import statsmodels.formula.api as smf
```

```
m11 = smf.ols('Y ~ X1 + X2',data=data1).fit()
m11.summary()
```

OLS Regression Results

Dep. Variable:	Y	**R-squared:**	0.053
Model:	OLS	**Adj. R-squared:**	-0.263
Method:	Least Squares	**F-statistic:**	0.1665

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	15.1678	5.682	2.670	0.037	1.265	29.070
X1	0.6019	3.474	0.173	0.868	-7.899	9.103
X2	0.7769	1.428	0.544	0.606	-2.716	4.270

Categorical Predictors – ONE-HOT ENCODING

Replace X_1 with binary columns

ONE-HOT ENCODING

X_1	X_2	Y
S	-0.10	19.19
S	2.53	22.74
S	4.86	23.91
M	0.26	7.07
M	2.55	7.93
M	4.87	8.93
L	0.08	20.63
L	2.62	23.46
L	5.09	25.75

X_{10}	X_{11}	X_{12}	X_2	Y
1	0	0	-0.10	19.19
1	0	0	2.53	22.74
1	0	0	4.86	23.91
0	1	0	0.26	7.07
0	1	0	2.55	7.93
0	1	0	4.87	8.93
0	0	1	0.08	20.63
0	0	1	2.62	23.46
0	0	1	5.09	25.75

Categorical Predictors – ONE-HOT ENCODING

Replace X_1 with binary columns

X_1	X_2	Y
S	-0.10	19.19
S	2.53	22.74
S	4.86	23.91
M	0.26	7.07
M	2.55	7.93
M	4.87	8.93
L	0.08	20.63
L	2.62	23.46
L	5.09	25.75

$n = 9$
$p = 2$

ONE-HOT ENCODING

X_{11}	X_{12}	X_2	Y
0	0	-0.10	19.19
0	0	2.53	22.74
0	0	4.86	23.91
1	0	0.26	7.07
1	0	2.55	7.93
1	0	4.87	8.93
0	1	0.08	20.63
0	1	2.62	23.46
0	1	5.09	25.75

$n = 9$
$p = 3$

Categorical Predictors – ONE-HOT ENCODING

```
data2 = data0.copy()
y = data2.Y
X = data2.drop(columns='Y',axis=1)
X
```

	X1	X2
0	S	-0.10
1	S	2.53
2	S	4.86
3	M	0.26
4	M	2.55
5	M	4.87
6	L	0.08
7	L	2.62
8	L	5.09

select categorical columns

```
X_binary = pd.get_dummies(X,columns = ['X1'])
X_binary
```

	X2	X1_L	X1_M	X1_S
0	-0.10	0	0	1
1	2.53	0	0	1
2	4.86	0	0	1
3	0.26	0	1	0
4	2.55	0	1	0
5	4.87	0	1	0
6	0.08	1	0	0
7	2.62	1	0	0
8	5.09	1	0	0

Categorical Predictors – ONE-HOT ENCODING

select categorical columns

```
X_binary = pd.get_dummies(X,columns = ['X1'])
X_binary
```

	X2	X1_L	X1_M	X1_S
0	-0.10	0	0	1
1	2.53	0	0	1
2	4.86	0	0	1
3	0.26	0	1	0
4	2.55	0	1	0
5	4.87	0	1	0
6	0.08	1	0	0
7	2.62	1	0	0
8	5.09	1	0	0

drop 1 binary column

```
X_binary.drop(columns = 'X1_S',
              inplace=True)
X_binary
```

	X2	X1_L	X1_M
0	-0.10	0	0
1	2.53	0	0
2	4.86	0	0
3	0.26	0	1
4	2.55	0	1
5	4.87	0	1
6	0.08	1	0
7	2.62	1	0
8	5.09	1	0

Categorical Predictors – ONE-HOT ENCODING

```
# rename columns
X_binary.columns = ['X2', 'L', 'M']
X_binary
```

	X2	L	M
0	-0.10	0	0
1	2.53	0	0
2	4.86	0	0
3	0.26	0	1
4	2.55	0	1
5	4.87	0	1
6	0.08	1	0
7	2.62	1	0
8	5.09	1	0

```
# reorder columns
X_binary = X_binary.reindex(columns = ['M','L','X2'])
X_binary
```

	M	L	X2
0	0	0	-0.10
1	0	0	2.53
2	0	0	4.86
3	1	0	0.26
4	1	0	2.55
5	1	0	4.87

Categorical Predictors – ONE-HOT ENCODING

.

```
m2 = LinearRegression().fit(X_binary,y)
R2 = m2.score(X_binary,y)
R2
```

```
0.9926482907525312
```

```
# Find adj R-squared with sklearn
```

```
n, p = 9, 3
```

```
print (1 - (1-R2)*(n-1)/(n-p-1))
```

```
0.98823726520405
```

Categorical Predictors – ONE-HOT ENCODING

	M	L	X2
0	0	0	-0.10
1	0	0	2.53
2	0	0	4.86
3	1	0	0.26
4	1	0	2.55
5	1	0	4.87
6	0	1	0.08
7	0	1	2.62
8	0	1	5.09

```
m2.intercept_
```

```
19.96499706357271
```

```
m2.coef_
```

```
array([-14.07601525,   1.19741635,   0.81550189])
```

Categorical Predictors – ONE-HOT ENCODING

	M	L	X2
0	0	0	-0.10
1	0	0	2.53
2	0	0	4.86
3	1	0	0.26
4	1	0	2.55
5	1	0	4.87
6	0	1	0.08
7	0	1	2.62
8	0	1	5.09

```
m2.intercept_
```

```
19.96499706357271
```

```
m2.coef_
```

```
array([-14.07601525,   1.19741635,   0.81550189])
```

$$\hat{Y} = \begin{cases} 19.965 & +0.8155\,X_2 \quad \text{when } X_1 = \text{S} \\ (19.965 - 14.076) & +0.8155\,X_2 \quad \text{when } X_1 = \text{M} \\ (19.965 + 1.1974) & +0.8155\,X_2 \quad \text{when } X_1 = \text{L} \end{cases}$$

Categorical Predictors – ONE-HOT ENCODING

	M	L	X2
0	0	0	-0.10
1	0	0	2.53
2	0	0	4.86
3	1	0	0.26
4	1	0	2.55
5	1	0	4.87
6	0	1	0.08
7	0	1	2.62
8	0	1	5.09

```
m2.intercept_
```

```
19.96499706357271
```

```
m2.coef_
```

M L X2

```
array([-14.07601525,   1.19741635,   0.81550189])
```

additional intercepts

slope

$$\hat{Y} = \begin{cases} 19.965 & +0.8155\,X_2 \quad \text{when } X_1 = \text{S} \\ (19.965 - 14.076) & +0.8155\,X_2 \quad \text{when } X_1 = \text{M} \\ (19.965 + 1.1974) & +0.8155\,X_2 \quad \text{when } X_1 = \text{L} \end{cases}$$

Categorical Predictors – ENCODING

Which encoding is best?

Which encoding is best?

	LABEL ENCODING	ONE-HOT ENCODING
R-squared	0.05259	0.9926
Adjusted R-squared:	-0.2632	0.9882

Why are the models different?

Categorical Predictors – EXAMPLE

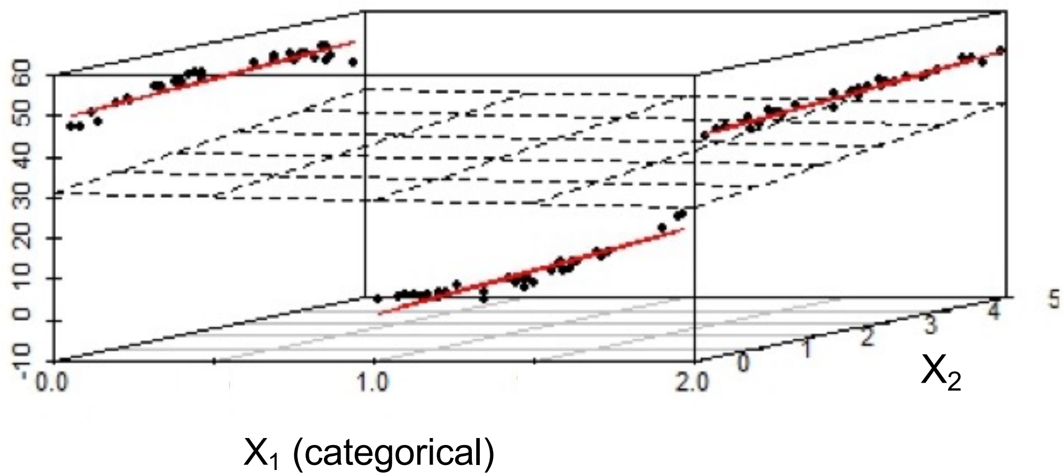
Label encoding equation

$$\hat{Y} = 15.16 + 0.602\,X_1 - 0.77\,X_2$$

One-hot encoding equations

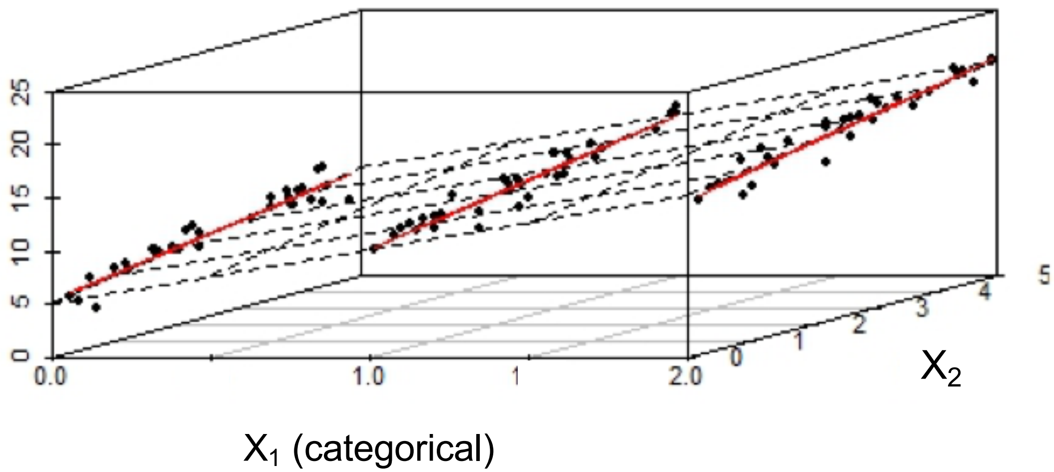
$$\hat{Y} = \begin{cases} 19.965 & +0.8155\,X_2 \quad \text{when } X_1 = \text{S} \\ (19.965 - 14.076) & +0.8155\,X_2 \quad \text{when } X_1 = \text{M} \\ (19.965 + 1.1974) & +0.8155\,X_2 \quad \text{when } X_1 = \text{L} \end{cases}$$

Categorical Predictors – EXAMPLE



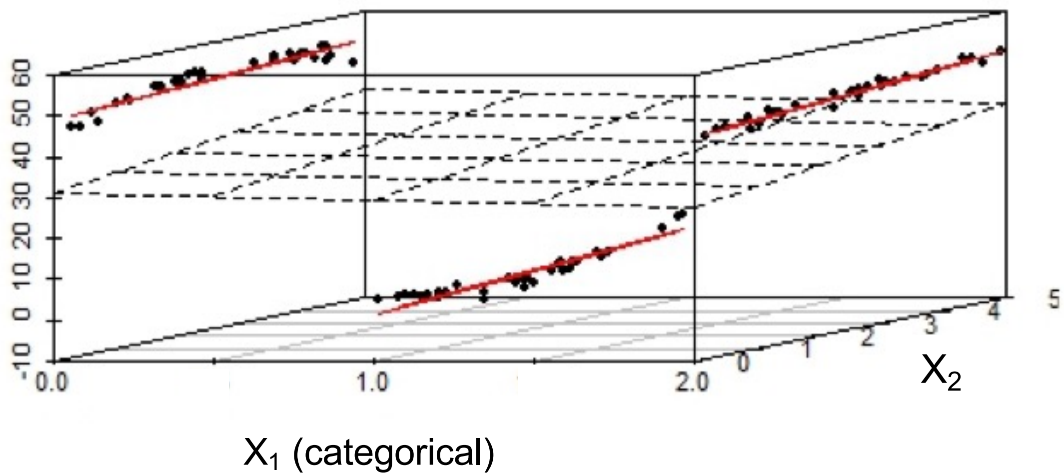
- label encoding results in a regression plane
- one-hot encoding results in three regression lines (one for each category: 0,1,2)

Categorical Predictors – EXAMPLE



- If the observations are close to the plane, then label encoding and one-hot encoding may agree

Categorical Predictors – EXAMPLE

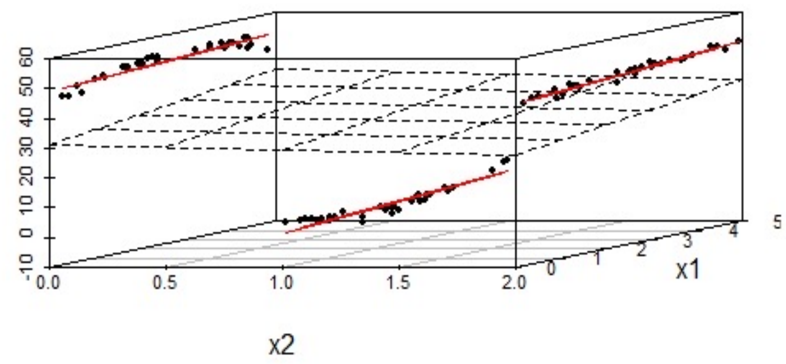


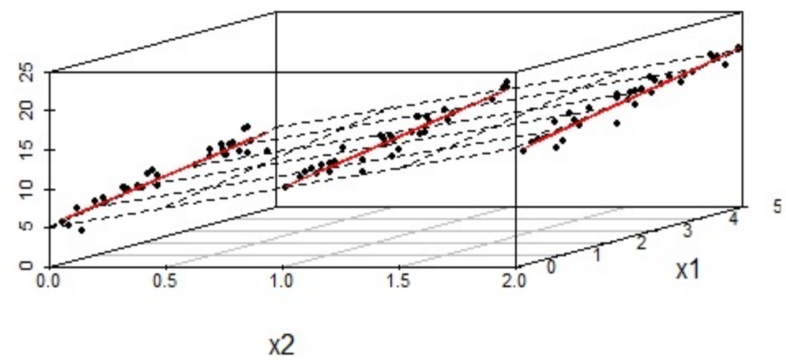
- If the observations are far away from the plane then One-hot encoding results in a better model

Categorical Predictors – EXAMPLE

- With a large number of variables in the model it is not possible to have a display like this

- We may relay on adj-R^2 or cross-validation error to choose the best model





Example 2

Forecasting with categorical variables

Categorical Predictors – EXAMPLE 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import statsmodels.formula.api as smf
```

```
df = pd.read_csv('part2.csv')
```

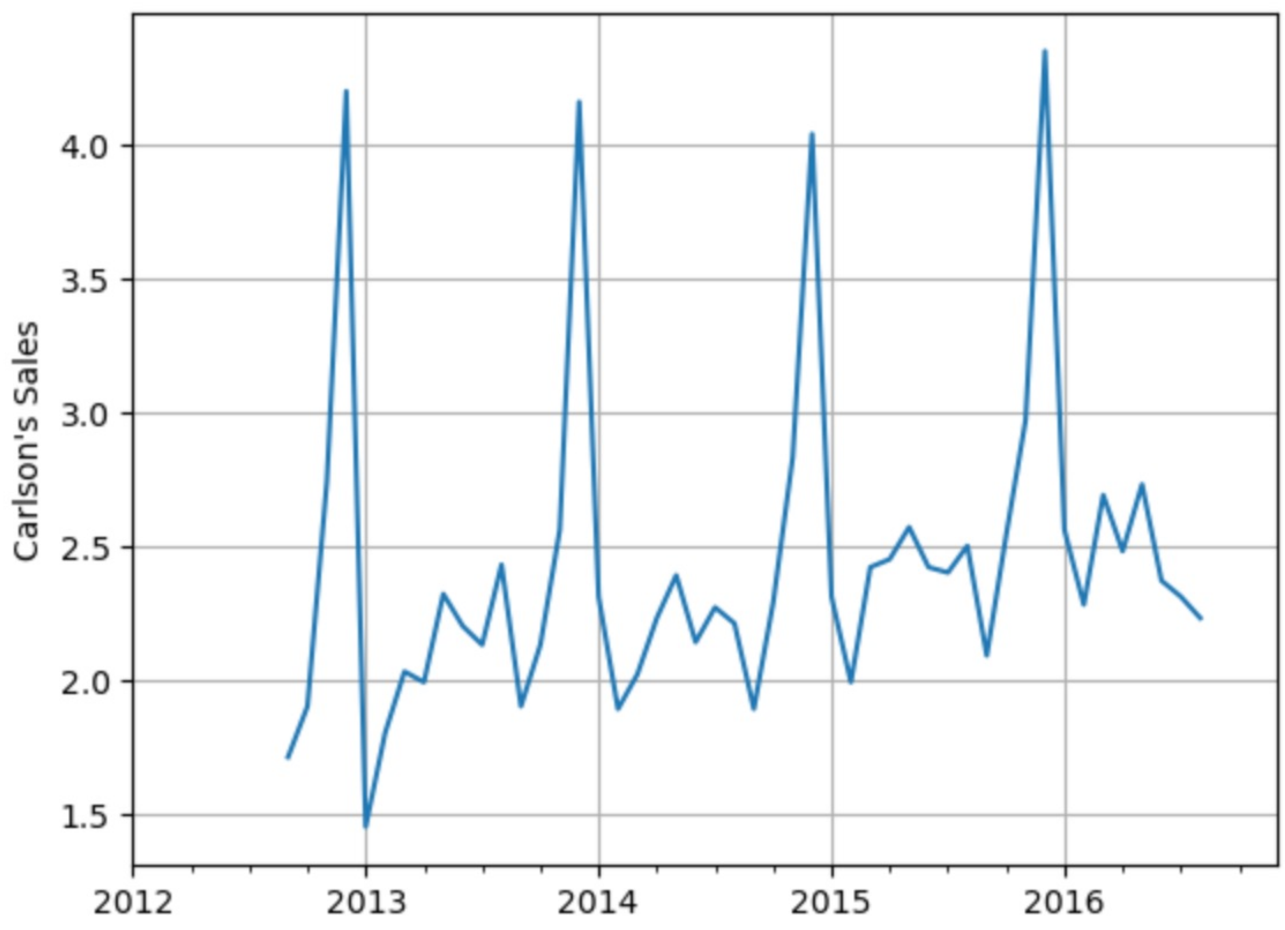
```
start = "2012-01-01"
end = "2016-12-01"

df.index = pd.date_range(start, end, freq='MS')
df[:5]
```

	Month	Year	sales
2012-01-01	January	2012	NaN
2012-02-01	February	2012	NaN
2012-03-01	March	2012	NaN
2012-04-01	April	2012	NaN
2012-05-01	May	2012	NaN

⋮

Categorical Predictors – EXAMPLE 2

```
df['sales'].plot()
plt.xlabel("")
plt.ylabel("Carlson's Sales")
plt.grid();
```



⋮

Categorical Predictors – EXAMPLE 2

How do we incorporate Month into the Model?

Month	Year	sales
October	2012	1.90
November	2012	2.74
December	2012	4.20
January	2013	1.45
February	2013	1.80
March	2013	2.03
April	2013	1.99
May	2013	2.32
June	2013	2.20
July	2013	2.13

Label encoding

for categorical variable Month

EXAMPLE 2 – LABEL ENCODING

predict **sales** using **Year** and **Month**

Month	Year	sales
January	2012	NaN
February	2012	NaN
March	2012	NaN
April	2012	NaN
May	2012	NaN
June	2012	NaN
July	2012	NaN
August	2012	NaN
September	2012	1.71
October	2012	1.90
November	2012	2.74
December	2012	4.20
January	2013	1.45
February	2013	1.80
March	2013	2.03
April	2013	1.99
May	2013	2.32
June	2013	2.20
July	2013	2.13
August	2013	2.43

EXAMPLE 2 – LABEL ENCODING

predict **sales** using **Year** and **Period**

Month	Year	sales	Period
January	2012	NaN	1
February	2012	NaN	2
March	2012	NaN	3
April	2012	NaN	4
May	2012	NaN	5
June	2012	NaN	6
July	2012	NaN	7
August	2012	NaN	8
September	2012	1.71	9
October	2012	1.90	10
November	2012	2.74	11
December	2012	4.20	12
January	2013	1.45	1
February	2013	1.80	2
March	2013	2.03	3
April	2013	1.99	4
May	2013	2.32	5
June	2013	2.20	6
July	2013	2.13	7
August	2013	2.43	8

label encode

EXAMPLE 2 – LABEL ENCODING

```
model2 = smf.ols('sales~ Period + Year',data = df).fit()
model2.summary()
```

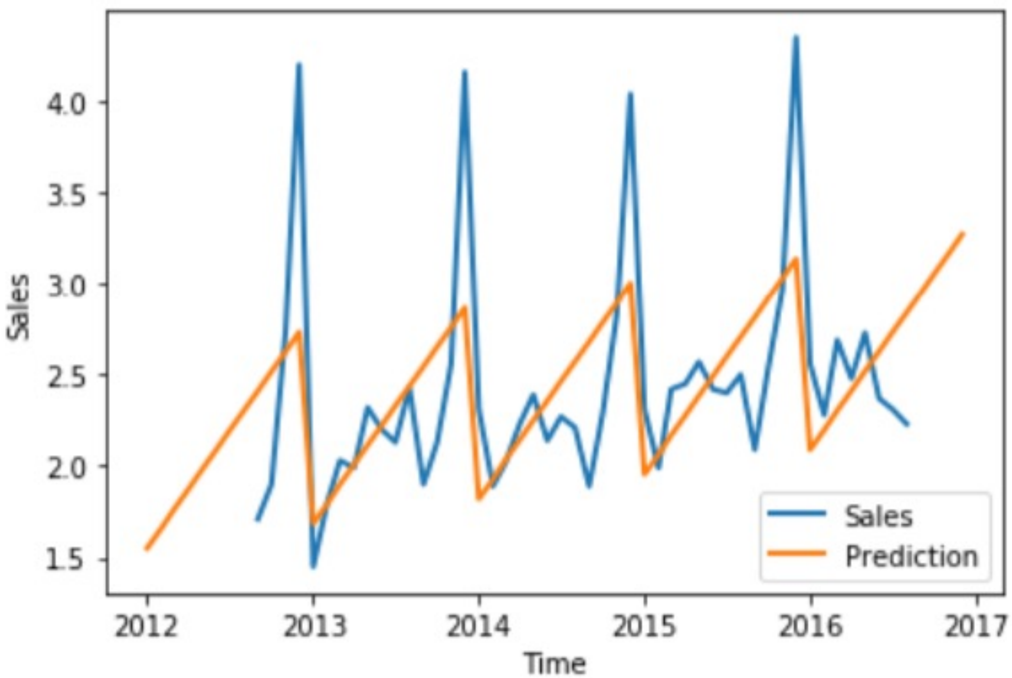
OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.343
Model:	OLS	**Adj. R-squared:**	0.314
Method:	Least Squares	**F-statistic:**	11.75
Date:	Thu, 14 May 2020	**Prob (F-statistic):**	7.86e-05
Time:	17:16:01	**Log-Likelihood:**	-33.960
No. Observations:	48	**AIC:**	73.92
Df Residuals:	45	**BIC:**	79.53
Df Model:	2		

Year	sales	Period
2012	NaN	1
2012	NaN	2
2012	NaN	3
2012	NaN	4
2012	NaN	5
2012	NaN	6
2012	NaN	7
2012	NaN	8
2012	1.71	9
2012	1.90	10
2012	2.74	11
2012	4.20	12
2013	1.45	1
2013	1.80	2
2013	2.03	3
2013	1.99	4
2013	2.32	5
2013	2.20	6
2013	2.13	7
2013	2.43	8

rows with NaN are ignored

EXAMPLE 2 – LABEL ENCODING



predict **sales** using **Year** and **Period**

Month	Year	sales	Period
January	2012	NaN	1
February	2012	NaN	2
March	2012	NaN	3
April	2012	NaN	4
May	2012	NaN	5
June	2012	NaN	6
July	2012	NaN	7
August	2012	NaN	8
September	2012	1.71	9
October	2012	1.90	10
November	2012	2.74	11
December	2012	4.20	12
January	2013	1.45	1
February	2013	1.80	2
March	2013	2.03	3
April	2013	1.99	4
May	2013	2.32	5
June	2013	2.20	6
July	2013	2.13	7
August	2013	2.43	8

One-hot encoding for categorical variable Month (with 12 categories)

EXAMPLE 2 – ONE-HOT ENCODING

predict **sales** using **Year** and **Month**

one-hot encode **Month**

(no coding needed with library smf)

Month	Year	sales
January	2012	NaN
February	2012	NaN
March	2012	NaN
April	2012	NaN
May	2012	NaN
June	2012	NaN
July	2012	NaN
August	2012	NaN
September	2012	1.71
October	2012	1.90
November	2012	2.74
December	2012	4.20
January	2013	1.45
February	2013	1.80
March	2013	2.03
April	2013	1.99
May	2013	2.32
June	2013	2.20
July	2013	2.13
August	2013	2.43

EXAMPLE 2 – ONE-HOT ENCODING

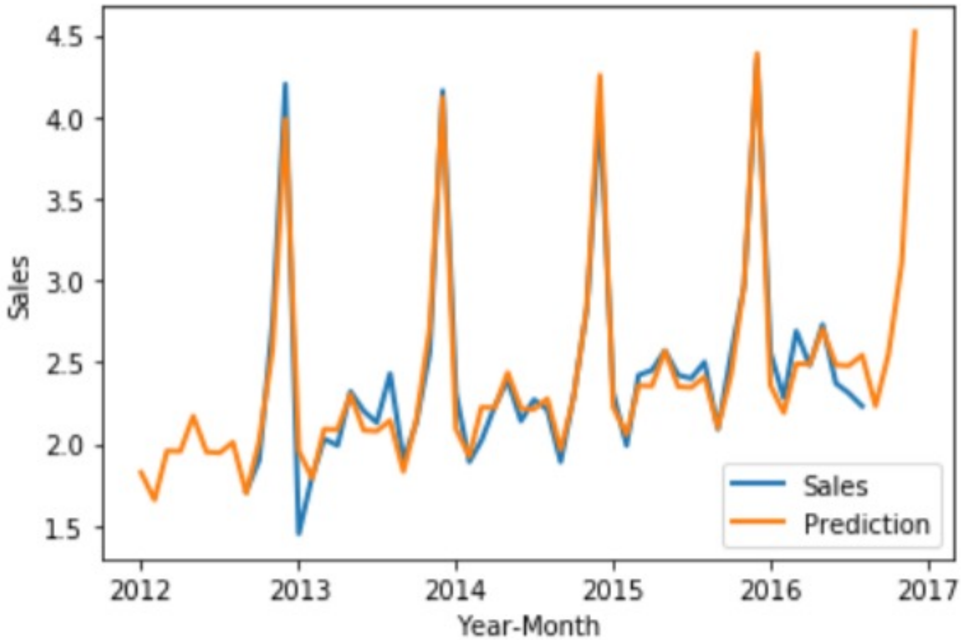
```
model3 = smf.ols('sales~Year+Month',data = df).fit()
model3.summary()
```

OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.947
Model:	OLS	**Adj. R-squared:**	0.929
Method:	Least Squares	**F-statistic:**	52.35
Date:	Thu, 14 May 2020	**Prob (F-statistic):**	1.01e-18
Time:	17:25:10	**Log-Likelihood:**	26.559
No. Observations:	48	**AIC:**	-27.12
Df Residuals:	35	**BIC:**	-2.792
Df Model:	12		

Month	Year	sales
January	2012	NaN
February	2012	NaN
March	2012	NaN
April	2012	NaN
May	2012	NaN
June	2012	NaN
July	2012	NaN
August	2012	NaN
September	2012	1.71
October	2012	1.90
November	2012	2.74
December	2012	4.20
January	2013	1.45
February	2013	1.80
March	2013	2.03
April	2013	1.99
May	2013	2.32
June	2013	2.20
July	2013	2.13
August	2013	2.43

EXAMPLE 2 – ONE-HOT ENCODING

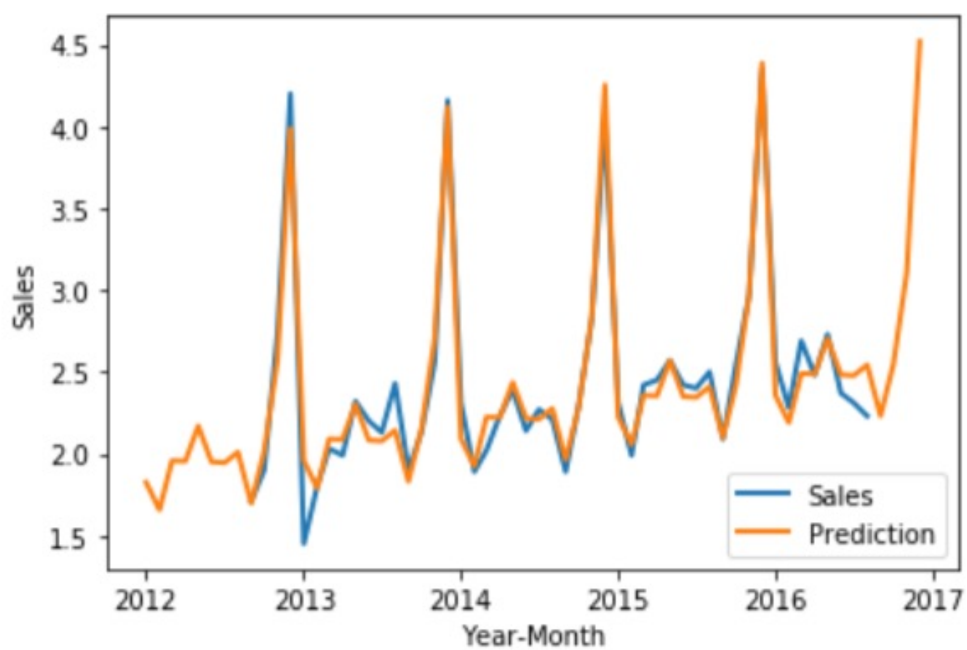


this is *linear* regression

Month	Year	sales
January	2012	NaN
February	2012	NaN
March	2012	NaN
April	2012	NaN
May	2012	NaN
June	2012	NaN
July	2012	NaN
August	2012	NaN
September	2012	1.71
October	2012	1.90
November	2012	2.74
December	2012	4.20
January	2013	1.45
February	2013	1.80
March	2013	2.03
April	2013	1.99
May	2013	2.32
June	2013	2.20
July	2013	2.13
August	2013	2.43

EXAMPLE 2 – ONE-HOT ENCODING

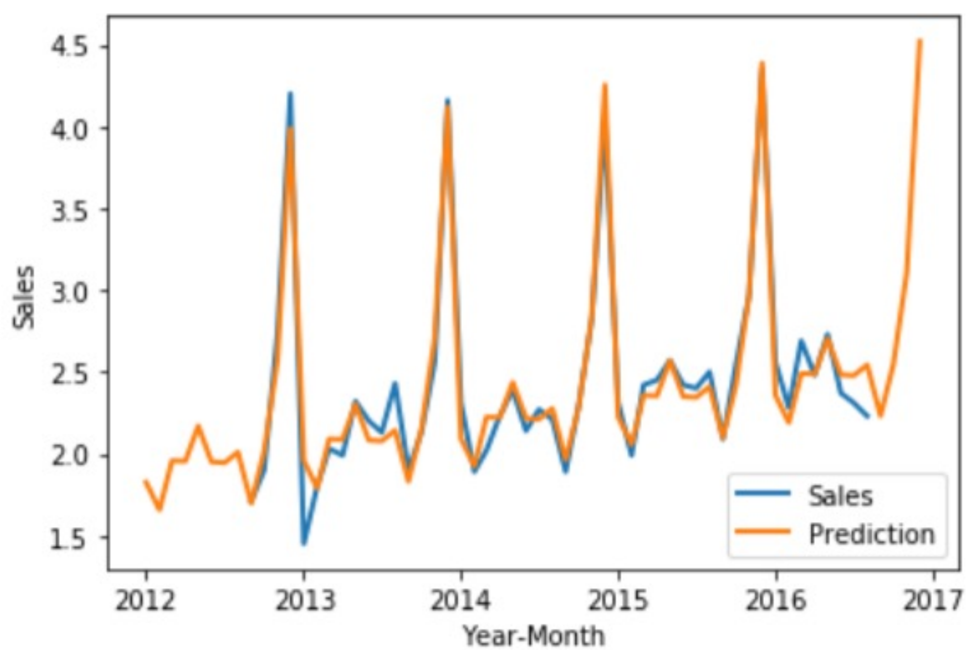
There are 12 regression equations (one for each month)




```
model3.params

Intercept              -266.312500
Month[T.August]           0.055000
Month[T.December]         2.033333
Month[T.February]        -0.297500
Month[T.January]         -0.130000
Month[T.July]            -0.010000
Month[T.June]            -0.005000
Month[T.March]            0.002500
Month[T.May]              0.215000
Month[T.November]         0.620833
Month[T.October]          0.060833
Month[T.September]       -0.256667
Year                      0.133333
dtype: float64
```

EXAMPLE 2 – ONE-HOT ENCODING

There are 12 regression equations (April is base model)

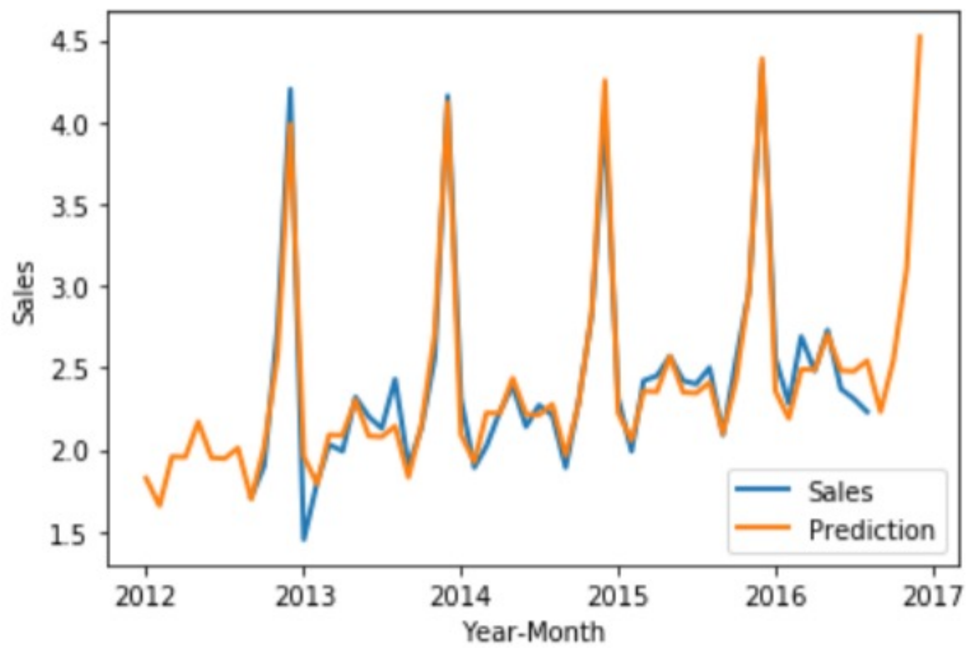



```
model3.params

Intercept              -266.312500
Month[T.August]           0.055000
Month[T.December]         2.033333
Month[T.February]        -0.297500
Month[T.January]         -0.130000
Month[T.July]            -0.010000
Month[T.June]            -0.005000
Month[T.March]            0.002500
Month[T.May]              0.215000
Month[T.November]         0.620833
Month[T.October]          0.060833
Month[T.September]       -0.256667
Year                      0.133333
dtype: float64
```

April prediction = -266.3125 + 0.1333 Year   base model

EXAMPLE 2 – ONE-HOT ENCODING

There are 12 regression equations (April is base model)



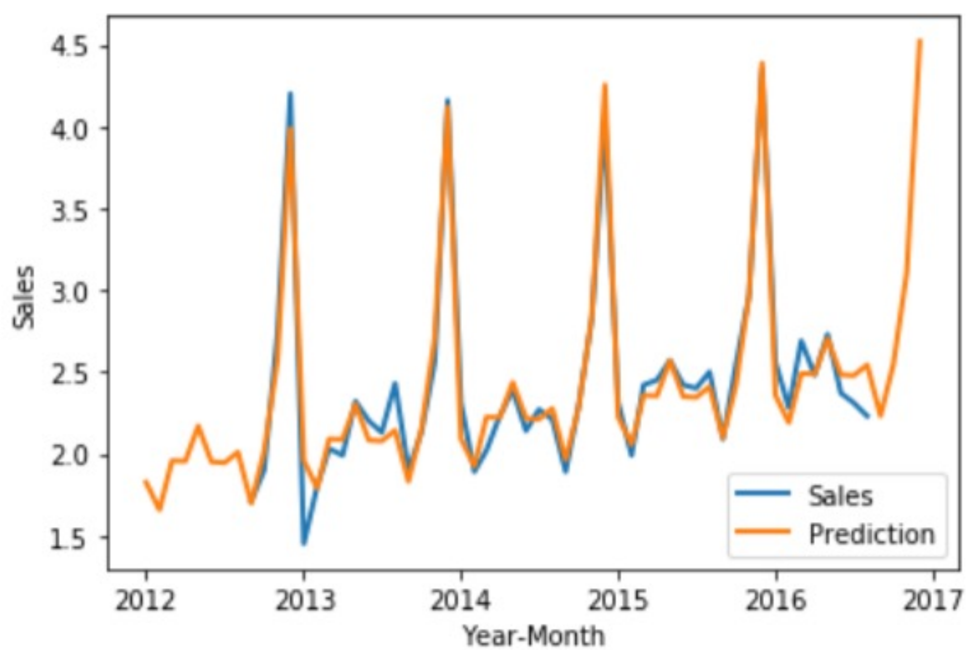
11 Additional intercepts

```
model3.params

Intercept              -266.312500
Month[T.August]           0.055000
Month[T.December]         2.033333
Month[T.February]        -0.297500
Month[T.January]         -0.130000
Month[T.July]            -0.010000
Month[T.June]            -0.005000
Month[T.March]            0.002500
Month[T.May]              0.215000
Month[T.November]         0.620833
Month[T.October]          0.060833
Month[T.September]       -0.256667
Year                      0.133333
dtype: float64
```

April prediction = -266.3125 + 0.1333 Year   base model

EXAMPLE 2 – ONE-HOT ENCODING

There are 12 regression equations (April is base model)

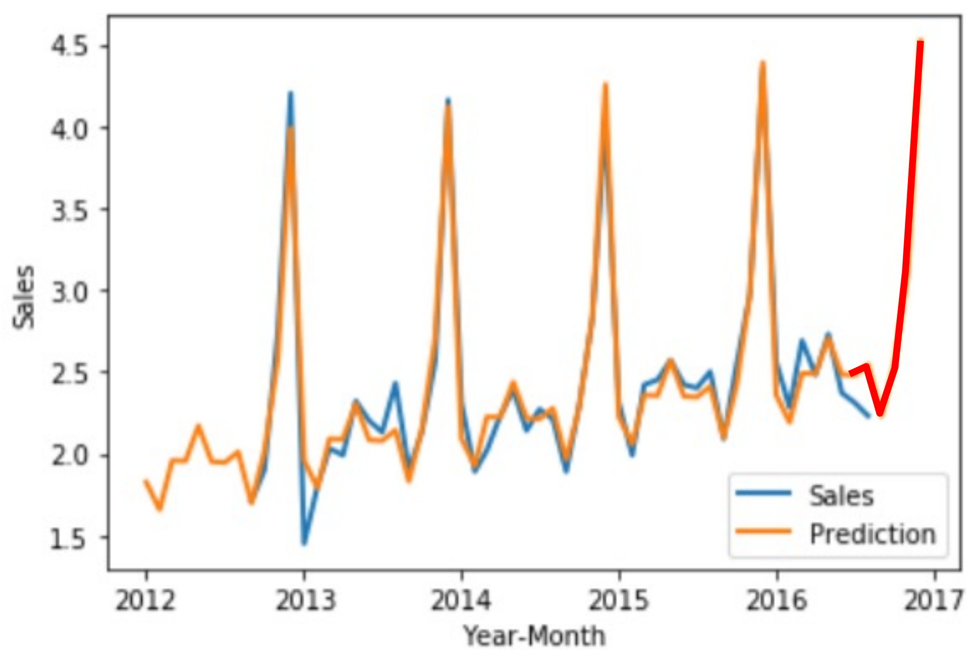



```
model3.params

Intercept               -266.312500
Month[T.August]            0.055000
Month[T.December]          2.033333
Month[T.February]         -0.297500
Month[T.January]          -0.130000
Month[T.July]             -0.010000
Month[T.June]             -0.005000
Month[T.March]             0.002500
Month[T.May]               0.215000
Month[T.November]          0.620833
Month[T.October]           0.060833
Month[T.September]        -0.256667
Year                       0.133333
dtype: float64
```

March prediction = (-266.3125 + 0.0025) + 0.1333 Year

EXAMPLE 2 – MODEL PREDICTIONS

.

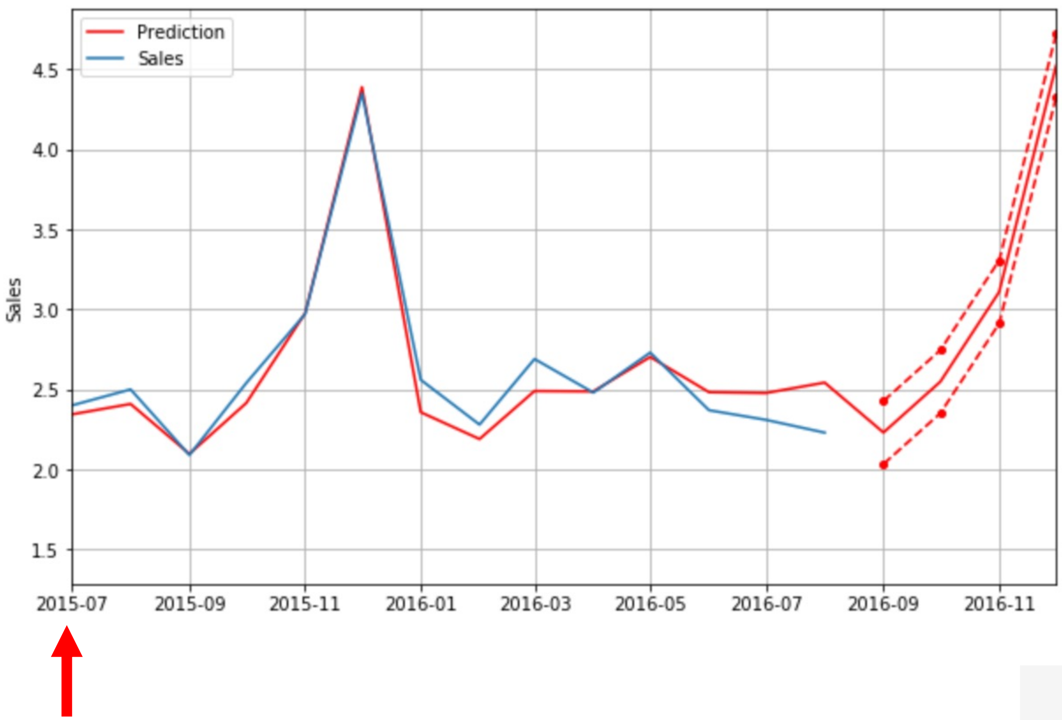



```
df['prediction'] = model3.predict(df)
df.tail()
```

	Month	Year	sales	prediction
2016-08-01	August	2016	2.23	2.542500
2016-09-01	September	2016	NaN	2.230833
2016-10-01	October	2016	NaN	2.548333
2016-11-01	November	2016	NaN	3.108333
2016-12-01	December	2016	NaN	4.520833

EXAMPLE 2 – MODEL PREDICTIONS

Predictions with CIs



Month	Year	sales	Date	prediction	lower	upper
June	2016	2.37	2016-06-01	2.482500	2.305126	2.659874
July	2016	2.31	2016-07-01	2.477500	2.300126	2.654874
August	2016	2.23	2016-08-01	2.542500	2.365126	2.719874
September	2016	NaN	2016-09-01	2.230833	2.033966	2.427701
October	2016	NaN	2016-10-01	2.548333	2.351466	2.745201
November	2016	NaN	2016-11-01	3.108333	2.911466	3.305201
December	2016	NaN	2016-12-01	4.520833	4.323966	4.717701

One-hot encoding for categorical variables **Year** and Month

EXAMPLE 2 – BUILD MODEL4

Build a regression model with **Year** and Month as categorical variables

```
model4 = smf.ols('sales ~ C(Year) + C(Month)',data = df).fit()
model4.summary()
```

OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.951
Model:	OLS	**Adj. R-squared:**	0.928
Method:	Least Squares	**F-statistic:**	41.27
Date:	Mon, 27 Sep 2021	**Prob (F-statistic):**	1.17e-16
Time:	12:45:20	**Log-Likelihood:**	28.265
No. Observations:	48	**AIC:**	-24.53
Df Residuals:	32	**BIC:**	5.409
Df Model:	15		

EXAMPLE 2 – MODEL4 COEFFICIENTS WHEN ONE-HOT ENCODING *YEAR*

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.0492	0.127	16.170	0.000	1.791	2.307
C(Year)[T.2013]	0.0272	0.103	0.265	0.792	-0.182	0.236
C(Year)[T.2014]	0.1447	0.103	1.411	0.168	-0.064	0.354
C(Year)[T.2015]	0.3531	0.103	3.442	0.002	0.144	0.562
C(Year)[T.2016]	0.4283	0.116	3.683	0.001	0.191	0.665
C(Month)[T.August]	0.0550	0.116	0.473	0.639	-0.182	0.292
C(Month)[T.December]	2.0071	0.120	16.743	0.000	1.763	2.251
C(Month)[T.February]	-0.2975	0.116	-2.558	0.015	-0.534	-0.061
C(Month)[T.January]	-0.1300	0.116	-1.118	0.272	-0.367	0.107
C(Month)[T.July]	-0.0100	0.116	-0.086	0.932	-0.247	0.227
C(Month)[T.June]	-0.0050	0.116	-0.043	0.966	-0.242	0.232
C(Month)[T.March]	0.0025	0.116	0.021	0.983	-0.234	0.239
C(Month)[T.May]	0.2150	0.116	1.849	0.074	-0.022	0.452
C(Month)[T.November]	0.5946	0.120	4.960	0.000	0.350	0.839
C(Month)[T.October]	0.0346	0.120	0.288	0.775	-0.210	0.279
C(Month)[T.September]	-0.2829	0.120	-2.360	0.025	-0.527	-0.039

EXAMPLE 2 – PREPARE NEW DATAFRAME

Collect year 2016 rows from original DataFrame

```
df4 = df.iloc[-12:].copy()
df4.drop(['prediction','Date'],axis = 1,
        inplace=True)
df4
```

	Month	Year	sales
48	January	2016	2.56
49	February	2016	2.28
50	March	2016	2.69
51	April	2016	2.48
52	May	2016	2.73
53	June	2016	2.37
54	July	2016	2.31
55	August	2016	2.23
56	September	2016	NaN
57	October	2016	NaN
58	November	2016	NaN
59	December	2016	NaN

EXAMPLE 2 – PREPARE NEW DATAFRAME TO PREDICT 2017 SALES

Collect year 2016 rows from original DataFrame to create a new one for 2017

```
df4.Year = 2017
df4.sales = np.nan
df4
```

	Month	Year	sales
48	January	2017	NaN
49	February	2017	NaN
50	March	2017	NaN
51	April	2017	NaN
52	May	2017	NaN
53	June	2017	NaN
54	July	2017	NaN
55	August	2017	NaN
56	September	2017	NaN
57	October	2017	NaN
58	November	2017	NaN
59	December	2017	NaN

EXAMPLE 2 – PREDICT *YEAR* 2017 SALES

```
df4['prediction'] = model4.predict(df4)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/opt/anaconda3/lib/python3.7/site-packages/patsy/categorical.py in categorical
)
    345             try:
--> 346                 out[i] = level_to_int[value]
    347             except KeyError:

KeyError: 2017

During handling of the above exception, another exception occurred:

Error converting data to categorical: observation with value 2017
does not match any of the expected levels
(expected: [2012, 2013, ..., 2015, 2016])

    sales ~ C(Year) + C(Month)
            ^^^^^^^
```

Model has not been trained with category 2017

Example 3

MLR with categorical vars

statsmodels.formula.api

EXAMPLE 3

- Use the *homes.csv* dataset to fit a full model for houses with two to four bedrooms.
- Find 95% PI for the price of a house with the following attributes

  - two bedrooms
  - three bathrooms
  - garage for two cars
  - high quality
  - built in 1996
  - area 3150 square feet
  - size 26250 square feet
  - with AC and pool
  - not close to a highway

EXAMPLE 3 SOLUTION

```
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
```

```
df0 = pd.read_csv('homes.csv')
df0[:3]
```

	price	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	360000	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO
1	340000	2058	4	2	2	1976	1	22912	YES	NO	MEDIUM	NO
2	250000	1780	4	3	2	1980	1	21345	YES	NO	MEDIUM	NO

EXAMPLE 3 SOLUTION

```
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
```

```
df0 = pd.read_csv('homes.csv')
df0[:3]
```

	price	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	360000	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO
1	340000	2058	4	2	2	1976	1	22912	YES	NO	MEDIUM	NO
2	250000	1780	4	3	2	1980	1	21345	YES	NO	MEDIUM	NO

convert style to a categorical variable

EXAMPLE 3 SOLUTION

```
df0['style'] = df0['style'].astype(object)
df0.dtypes
```

```
price        int64
area         int64
beds         int64
baths        int64
garage       int64
year         int64
style       object
lotsize      int64
ac          object
pool        object
quality     object
highway     object
```

```
df = df0.copy()
```

```
df = df[(df.beds > 1) & (df.beds < 5)]
```

We will build a MLR Model with five categorical variables and six numerical variables

EXAMPLE 3 SOLUTION

```
model1 = smf.ols(formula = 'price ~ area+beds+baths+garage+year+\
                              C(style)+lotsize+C(ac)+C(pool)+\
                              C(quality)+C(highway)',
                 data = df).fit()
```

EXAMPLE 3 SOLUTION

```
model1 = smf.ols(formula = 'price ~ area+beds+baths+garage+year+\
                          style+lotsize+ac+pool+quality+highway',
                 data = df).fit()
model1.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.852
Model:	OLS	**Adj. R-squared:**	0.845
Method:	Least Squares	**F-statistic:**	135.9

EXAMPLE 3 SOLUTION – MODEL COEFFICIENTS

.

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.473e+06	3.93e+05	-6.285	0.000	-3.25e+06	-1.7e+06
C(style)[T.2]	-2.006e+04	8638.441	-2.322	0.021	-3.7e+04	-3080.425
C(style)[T.3]	-1.151e+04	8251.632	-1.395	0.164	-2.77e+04	4707.107
C(style)[T.4]	2.31e+04	1.72e+04	1.346	0.179	-1.06e+04	5.68e+04
C(style)[T.5]	-7407.6304	1.55e+04	-0.479	0.632	-3.78e+04	2.3e+04
C(style)[T.6]	-3.06e+04	1.51e+04	-2.029	0.043	-6.02e+04	-951.905
C(style)[T.7]	-4.664e+04	8514.836	-5.477	0.000	-6.34e+04	-2.99e+04
C(style)[T.9]	-9.094e+04	5.26e+04	-1.728	0.085	-1.94e+05	1.25e+04

categorical variable style. Base level is style 1

↑ T means category

EXAMPLE 3 SOLUTION

C(ac)[T.YES]	-1075.0761	7501.411	-0.143	0.886	-1.58e+04	1.37e+04	categorical variables
C(pool)[T.YES]	2.163e+04	1.06e+04	2.042	0.042	813.263	4.25e+04	
C(quality)[T.LOW]	-1.368e+05	1.45e+04	-9.433	0.000	-1.65e+05	-1.08e+05	
C(quality)[T.MEDIUM]	-1.363e+05	1.08e+04	-12.640	0.000	-1.58e+05	-1.15e+05	
C(highway)[T.YES]	-3.18e+04	1.69e+04	-1.884	0.060	-6.5e+04	1380.295	
area	117.2176	7.739	15.146	0.000	102.006	132.429	numerical variables
beds	-2222.9820	4148.594	-0.536	0.592	-1.04e+04	5931.279	
baths	9246.5631	4285.513	2.158	0.032	823.180	1.77e+04	
garage	7423.7368	4914.563	1.511	0.132	-2236.074	1.71e+04	
year	1300.6190	199.782	6.510	0.000	907.938	1693.300	
lotsize	1.3283	0.238	5.584	0.000	0.861	1.796	

EXAMPLE 3 SOLUTION – BASE CATEGORY FOR *QUALITY* IS *HIGH*

C(ac)[T.YES]	-1075.0761	7501.411	-0.143	0.886	-1.58e+04	1.37e+04
C(pool)[T.YES]	2.163e+04	1.06e+04	2.042	0.042	813.263	4.25e+04
C(quality)[T.LOW]	-1.368e+05	1.45e+04	-9.433	0.000	-1.65e+05	-1.08e+05
C(quality)[T.MEDIUM]	-1.363e+05	1.08e+04	-12.640	0.000	-1.58e+05	-1.15e+05
C(highway)[T.YES]	-3.18e+04	1.69e+04	-1.884	0.060	-6.5e+04	1380.295
area	117.2176	7.739	15.146	0.000	102.006	132.429
beds	-2222.9820	4148.594	-0.536	0.592	-1.04e+04	5931.279
baths	9246.5631	4285.513	2.158	0.032	823.180	1.77e+04
garage	7423.7368	4914.563	1.511	0.132	-2236.074	1.71e+04
year	1300.6190	199.782	6.510	0.000	907.938	1693.300
lotsize	1.3283	0.238	5.584	0.000	0.861	1.796

categorical variables

numerical variables

EXAMPLE 3 SOLUTION – Set the Base level for *quality*

```
model1 = smf.ols(formula = 'price ~ area+beds+baths+garage+year+\
                                    C(style)+lotsize+C(ac)+C(pool)+\
                                    C(quality,Treatment(reference="LOW"))+\
                                    C(highway)',
                 data = df).fit()
model1.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.852
Model:	OLS	**Adj. R-squared:**	0.845

EXAMPLE 3 SOLUTION – BASE LEVEL FOR *QUALITY* IS *LOW*

additional intercept for quality HIGH

categorical variables

C(ac)[T.YES]	-1075.0761	7501.411	-0.143	0.886	-1.58e+04	1.37e+04
C(pool)[T.YES]	2.163e+04	1.06e+04	2.042	0.042	813.263	4.25e+04
C(quality, Treatment(reference="LOW"))[T.HIGH]	1.368e+05	1.45e+04	9.433	0.000	1.08e+05	1.65e+05
C(quality, Treatment(reference="LOW"))[T.MEDIUM]	437.0813	7899.182	0.055	0.956	-1.51e+04	1.6e+04
C(highway)[T.YES]	-3.18e+04	1.69e+04	-1.884	0.060	-6.5e+04	1380.295
area	117.2176	7.739	15.146	0.000	102.006	132.429
beds	-2222.9820	4148.594	-0.536	0.592	-1.04e+04	5931.279
baths	9246.5631	4285.513	2.158	0.032	823.180	1.77e+04
garage	7423.7368	4914.563	1.511	0.132	-2236.074	1.71e+04
year	1300.6190	199.782	6.510	0.000	907.938	1693.300
lotsize	1.3283	0.238	5.584	0.000	0.861	1.796

numerical variables

additional intercept for quality MEDIUM

EXAMPLE 3 SOLUTION – BASE LEVEL FOR *QUALITY* IS *LOW*

additional price for quality HIGH

categorical variables

C(ac)[T.YES]	-1075.0761	7501.411	-0.143	0.886	-1.58e+04	1.37e+04
C(pool)[T.YES]	2.163e+04	1.06e+04	2.042	0.042	813.263	4.25e+04
C(quality, Treatment(reference="LOW"))[T.HIGH]	1.368e+05	1.45e+04	9.433	0.000	1.08e+05	1.65e+05
C(quality, Treatment(reference="LOW"))[T.MEDIUM]	437.0813	7899.182	0.055	0.956	-1.51e+04	1.6e+04
C(highway)[T.YES]	-3.18e+04	1.69e+04	-1.884	0.060	-6.5e+04	1380.295
area	117.2176	7.739	15.146	0.000	102.006	132.429
beds	-2222.9820	4148.594	-0.536	0.592	-1.04e+04	5931.279
baths	9246.5631	4285.513	2.158	0.032	823.180	1.77e+04
garage	7423.7368	4914.563	1.511	0.132	-2236.074	1.71e+04
year	1300.6190	199.782	6.510	0.000	907.938	1693.300
lotsize	1.3283	0.238	5.584	0.000	0.861	1.796

numerical variables

additional price for quality MEDIUM

EXAMPLE 3 SOLUTION – NEW DATAFRAME FOR PREDICTION

```
newvalue = df[:1].copy()
del newvalue['price']
```

copy 1st row only

```
newvalue
```

	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO

EXAMPLE 3 SOLUTION – NEW DATAFRAME FOR PREDICTION

```
newvalue = df[:1].copy()
del newvalue['price']
```

copy 1st row only

```
newvalue
```

	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO

```
newvalue.area = 3150
newvalue.beds = 2
newvalue.baths = 3
newvalue.garage = 2
newvalue.year = 1996
newvalue.style = 1
newvalue.lotsize = 26250
newvalue.ac = 'YES'
newvalue.pool = 'YES'
newvalue.quality = 'HIGH'
newvalue.highway = 'NO'
```

replace entries with those of the house whose price is to be predicted

Modified dataframe newvalue in the next slide

EXAMPLE 3 SOLUTION

```
newvalue
```

	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	3150	2	3	2	1996	1	26250	YES	YES	HIGH	NO

```
model1.predict(newvalue)
```

```
0     585922.526796
```

EXAMPLE 3 SOLUTION – CI and PI

```
newvalue
```

	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	3150	2	3	2	1996	1	26250	YES	YES	HIGH	NO

```
model1.predict(newvalue)
```

```
0     585922.526796
```

```
df2 = model1.get_prediction(newvalue)
df2.summary_frame()
```

alpha = 0.05

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	585922.526796	15323.494616	555803.458271	616041.595321	479849.693761	691995.359831

prediction — confidence interval — prediction interval

EXAMPLE 3 SOLUTION

- two bedrooms
- three bathrooms
- garage for two cars
- high quality
- built in 1996
- area 3150 square feet
- size 26250 square feet
- with AC and pool
- not close to a highway

What is the estimated price of a house with this description?

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	585922.526796	15323.494616	555803.458271	616041.595321	479849.693761	691995.359831

EXAMPLE 3 SOLUTION

- two bedrooms
- three bathrooms
- garage for two cars
- high quality
- built in 1996
- area 3150 square feet
- size 26250 square feet
- with AC and pool
- not close to a highway

What is the estimated price of a house with this description?

What is a 95% range for this estimated price?

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	585922.526796	15323.494616	555803.458271	616041.595321	479849.693761	691995.359831

prediction interval

EXAMPLE 3

- two bedrooms
- three bathrooms
- garage for two cars
- high quality
- built in 1996
- area 3150 square feet
- size 26250 square feet
- with AC and pool
- not close to a highway

What is the estimated average price of all houses with this description?

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	585922.526796	15323.494616	555803.458271	616041.595321	479849.693761	691995.359831

EXAMPLE 3

- two bedrooms
- three bathrooms
- garage for two cars
- high quality
- built in 1996
- area 3150 square feet
- size 26250 square feet
- with AC and pool
- not close to a highway

What is the estimated average price of all houses with this description?

What is a 95% range for this estimated average?

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	585922.526796	15323.494616	555803.458271	616041.595321	479849.693761	691995.359831

confidence interval

Example 3

MLR with categorical vars

sklearn

EXAMPLE 3 – ONE-HOT ENCODING

- With statsmodels.formula.api, one-hot encoding is the default. The user does not need to create binary columns
- With sklearn the user must transform categorical columns into binary columns using **pd.get_dummies( )**

EXAMPLE 3 with SKLEARN

```
import numpy as np
import pandas as pd
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
df0 = pd.read_csv('homes.csv')
df0[:3]
```

	price	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	360000	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO
1	340000	2058	4	2	2	1976	1	22912	YES	NO	MEDIUM	NO
2	250000	1780	4	3	2	1980	1	21345	YES	NO	MEDIUM	NO

EXAMPLE 3 with SKLEARN

```
# Change style to categorical
```

```
df0['style'] = df0['style'].astype(object)
df0.dtypes
```

```
price        int64
area         int64
beds         int64
baths        int64
garage       int64
year         int64
style       object
lotsize      int64
ac          object
pool        object
quality     object
highway     object
```

ONE-HOT ENCODING – Create binary columns for categoricals

```
df = df0.copy()
df = df[(df.beds > 1) & (df.beds < 5)]
df2 = pd.get_dummies(df,
                     columns = ['style','ac','pool',
                                'quality','highway'],
                     drop_first = True)
```

ONE-HOT ENCODING – Create binary columns for all categoricals

```
df = df0.copy()
df = df[(df.beds > 1) & (df.beds < 5)]
df2 = pd.get_dummies(df,
                     columns = ['style','ac','pool',
                                'quality','highway'],
                     drop_first = True)
```

← remove binary column of the first category in the data

ONE-HOT ENCODING – Create binary columns for all categoricals

```
df = df0.copy()
df = df[(df.beds > 1) & (df.beds < 5)]
df2 = pd.get_dummies(df,
                     columns = ['style','ac','pool',
                                'quality','highway'],
                     drop_first = True)
df2[:5]
```

remove binary column of the first category in the data

	price	area	beds	baths	garage	year	lotsize	style_2	style_3	style_4	style_5	style_6
0	360000	3032	4	4	2	1972	22221	0	0	0	0	0
1	340000	2058	4	2	2	1976	22912	0	0	0	0	0
2	250000	1780	4	3	2	1980	21345	0	0	0	0	0
3	205500	1638	4	2	2	1963	17342	0	0	0	0	0
4	275500	2196	4	3	2	1968	21786	0	0	0	0	0

…

EXAMPLE 3 with SKLEARN – Create binary columns for categoricals

df

	price	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	360000	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO
1	340000	2058	4	2	2	1976	1	22912	YES	NO	MEDIUM	NO
2	250000	1780	4	3	2	1980	1	21345	YES	NO	MEDIUM	NO

df2

	price	area	beds	baths	garage	year	lotsize	style_2	style_3	style_4	style_5	style_6
0	360000	3032	4	4	2	1972	22221	0	0	0	0	0
1	340000	2058	4	2	2	1976	22912	0	0	0	0	0
2	250000	1780	4	3	2	1980	21345	0	0	0	0	0
3	205500	1638	4	2	2	1963	17342	0	0	0	0	0
4	275500	2196	4	3	2	1968	21786	0	0	0	0	0

EXAMPLE 3 with SKLEARN – Create binary columns for categoricals

	price	area	beds	baths	garage	year	style	lotsize	ac	pool	quality	highway
0	360000	3032	4	4	2	1972	1	22221	YES	NO	MEDIUM	NO
1	340000	2058	4	2	2	1976	1	22912	YES	NO	MEDIUM	NO
2	250000	1780	4	3	2	1980	1	21345	YES	NO	MEDIUM	NO

	price	area	beds	baths	garage	year	lotsize	style_2	style_3	style_4	style_5	style_6
0	360000	3032	4	4	2	1972	22221	0	0	0	0	0
1	340000	2058	4	2	2	1976	22912	0	0	0	0	0
2	250000	1780	4	3	2	1980	21345	0	0	0	0	0
3	205500	1638	4	2	2	1963	17342	0	0	0	0	0
4	275500	2196	4	3	2	1968	21786	0	0	0	0	0

when style = 1 all other styles are equal to 0

EXAMPLE 3 with SKLEARN – Get the Regression model

```
y = df2.price
X = df2.drop(columns = 'price',axis = 1)
```

split df2 into response and predictors

```
model2 = LinearRegression().fit(X,y)
```

```
# R-squared
```

```
model2.score(X,y)
```

```
0.8516357572716687
```

```
# MSE
```

```
yhat = model2.predict(X)
```

```
MSE = mean_squared_error(y,yhat)
MSE
```

```
2563201465.1350064
```

EXAMPLE 3 with SKLEARN

```
model2.intercept_
```

```
-2472913.948993484
```

```
model2.coef_
```

```
array([    117.21757024,   -2222.98200076,    9246.56308256,
          7423.73675513,    1300.61897791,       1.32829835,
        -20059.69719175,  -11511.8737492 ,   23097.27289768,
         -7407.63042457,  -30595.69840355,  -46636.80287504,
        -90937.90731062,   -1075.0761224 ,   21631.69537831,
       -136786.65107443, -136349.56975042,  -31800.51432548])
```

EXAMPLE 3 with SKLEARN – Display regression coefficients in a dataframe df3

```
df3 = pd.DataFrame(model2.coef_.round(2),
                   columns = ['coef'],
                   index = X.columns)
df3
```

	coef
area	117.22
beds	-2222.98
baths	9246.56
garage	7423.74
year	1300.62
lotsize	1.33
style_2	-20059.70
style_3	-11511.87

	coef
style_4	23097.27
style_5	-7407.63
style_6	-30595.70
style_7	-46636.80
style_9	-90937.91
ac_YES	-1075.08
pool_YES	21631.70
quality_LOW	-136786.65
quality_MEDIUM	-136349.57
highway_YES	-31800.51

EXAMPLE 3 with SKLEARN - PREDICTION

```
newvalue = df2[:1].copy()
del newvalue['price']
newvalue
```

	area	beds	baths	garage	year	lotsize	style_2	style_3	style_4	style_5
0	3032	4	4	2	1972	22221	0	0	0	0

style_6	style_7	style_9	ac_YES	pool_YES	quality_LOW	quality_MEDIUM
0	0	0	1	0	0	1

EXAMPLE 3 with SKLEARN - PREDICTION

```
newvalue.area = 3150
newvalue.beds = 2
newvalue.baths = 3
newvalue.garage = 2
newvalue.year = 1996
newvalue.lotsize = 26250
newvalue.ac_YES = 1
newvalue.pool_YES = 1
newvalue.quality_LOW = 0
newvalue.quality_MEDIUM = 0
newvalue.highway_YES = 0
```

```
model2.predict(newvalue)
```

```
array([585922.52679621])
```