In [14]:
```
# dummy variables

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [15]:
```
#Example: automobile data
#https://www.kaggle.com/toramky/automobile-dataset

carsdata=pd.read_csv('Automobile_data.csv')
carsdata.sample(5)
```

Out[15]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	en loc
141	0	102	subaru	gas	std	four	sedan	fwd	
37	0	106	honda	gas	std	two	hatchback	fwd	
97	1	103	nissan	gas	std	four	wagon	fwd	
39	0	85	honda	gas	std	four	sedan	fwd	
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	

5 rows × 26 columns

In [16]:
```
# categorical variables: fuel-type
carsdata['city-mpg'].replace('?', np.nan, inplace= True)
carsdata['fuel-type'].replace('?', np.nan, inplace= True)
cars=carsdata.dropna()

#check the summary of 'fuel-type': two levels: gas and diesel
print(cars['fuel-type'].value_counts())
```

```
gas       185
diesel     20
Name: fuel-type, dtype: int64
```

In [19]:
```
# regression city-mpg~fuel-type, names wouldn't be recogonized in smf.ols

cars['citympg']=cars['city-mpg']
cars['fueltype']=cars['fuel-type']
reg = smf.ols('citympg ~ fueltype', data=cars).fit()
reg.summary()

# in the summary it only showed 1 level: gas
```

Out[19]:

OLS Regression Results

Dep. Variable:	citympg	**R-squared:**	0.066
Model:	OLS	**Adj. R-squared:**	0.061
Method:	Least Squares	**F-statistic:**	14.23
Date:	Mon, 20 Sep 2021	**Prob (F-statistic):**	0.000212
Time:	14:11:04	**Log-Likelihood:**	-668.48
No. Observations:	205	**AIC:**	1341.
Df Residuals:	203	**BIC:**	1348.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	30.3000	1.418	21.374	0.000	27.505	33.095
fueltype[T.gas]	-5.6297	1.492	-3.773	0.000	-8.572	-2.687

Omnibus:	17.025	**Durbin-Watson:**	0.846
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18.937
Skew:	0.668	**Prob(JB):**	7.73e-05
Kurtosis:	3.658	**Cond. No.**	6.25

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [17]:

```
#similarly, drivewheels has 3 levels
cars['drivewheels']=cars['drive-wheels']
print(cars['drivewheels'].value_counts())
```

```
fwd    120
rwd     76
4wd      9
Name: drivewheels, dtype: int64
```

In [20]:

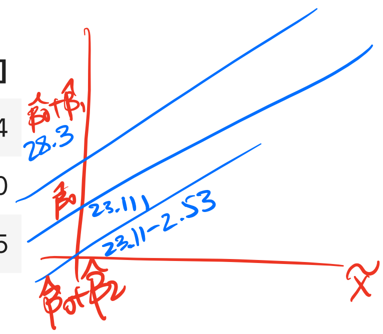
```
reg1 = smf.ols('citympg ~ drivewheels', data=cars).fit()
reg1.summary()

#two levels: fwd, rwd showed up, 4wd as the baseline level doesn't show
```

Out[20]:

OLS Regression Results

Dep. Variable:	citympg	**R-squared:**	0.324
Model:	OLS	**Adj. R-squared:**	0.317
Method:	Least Squares	**F-statistic:**	48.38
Date:	Mon, 20 Sep 2021	**Prob (F-statistic):**	6.81e-18
Time:	14:11:09	**Log-Likelihood:**	-635.31
No. Observations:	205	**AIC:**	1277.
Df Residuals:	202	**BIC:**	1287.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	23.1111	1.802	12.825	0.000	19.558	26.664
drivewheels[T.fwd]	5.2056	1.868	2.786	0.006	1.522	8.890
drivewheels[T.rwd]	-2.5322	1.906	-1.329	0.185	-6.290	1.225



Omnibus:	15.581	**Durbin-Watson:**	1.175
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	19.666
Skew:	0.537	**Prob(JB):**	5.36e-05
Kurtosis:	4.072	**Cond. No.**	10.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [22]:
```
sm.stats.anova_lm(reg1, typ=2) #in anova, it's analysis as a whole
# reduce: null model
# full: y~drivewheels_fwd+ drivewheels_rwd
# notice: df of drivewheels is 2, since there are two parameters to estimate
```

Out[22]:

	sum_sq	**df**	**F**	**PR(>F)**
drivewheels	2827.740080	2.0	48.379345	6.809467e-18
Residual	5903.381871	202.0	NaN	NaN

In [23]:
```
# same for MLR with categorical variables
cars['enginesize']=cars['engine-size']
reg2 = smf.ols('citympg ~ enginesize+ drivewheels + fueltype', data=cars).fi
reg2.summary()
```

Out[23]:

OLS Regression Results

Dep. Variable:	citympg	**R-squared:**	0.599
Model:	OLS	**Adj. R-squared:**	0.591
Method:	Least Squares	**F-statistic:**	74.59
Date:	Mon, 20 Sep 2021	**Prob (F-statistic):**	1.36e-38
Time:	14:16:28	**Log-Likelihood:**	-581.84
No. Observations:	205	**AIC:**	1174.
Df Residuals:	200	**BIC:**	1190.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	38.9139	1.954	19.911	0.000	35.060	42.768
drivewheels[T.fwd]	4.5714	1.449	3.156	0.002	1.715	7.428
drivewheels[T.rwd]	0.2171	1.537	0.141	0.888	-2.813	3.247
fueltype[T.gas]	-7.0471	0.994	-7.090	0.000	-9.007	-5.087
enginesize	-0.0795	0.009	-9.319	0.000	-0.096	-0.063

Omnibus:	33.046	**Durbin-Watson:**	1.095
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	84.717
Skew:	0.689	**Prob(JB):**	4.02e-19
Kurtosis:	5.832	**Cond. No.**	1.19e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.19e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [24]:

```
sm.stats.anova_lm(reg2, typ=2)
```

Out[24]:

	sum_sq	df	F	PR(>F)
drivewheels	699.332669	2.0	19.958355	1.250114e-08
fueltype	880.634747	1.0	50.265123	2.260697e-11
enginesize	1521.368276	1.0	86.837095	2.213074e-17
Residual	3503.959410	200.0	NaN	NaN

In [10]:
```
sm.stats.anova_lm(reg2, typ=2)
```

Out[10]:

	sum_sq	df	F	PR(>F)
drivewheels	699.332669	2.0	19.958355	1.250114e-08
fueltype	880.634747	1.0	50.265123	2.260697e-11
enginesize	1521.368276	1.0	86.837095	2.213074e-17
Residual	3503.959410	200.0	NaN	NaN

In [13]:
```
# when the variable is categorical, but the levels are incidated as numbers
# or if you want to analyze an ordinal variable as categorical
# we can "force" it to be categorical using "C()" in smf.ols

credit = pd.read_csv("Credit.csv")
credit.sample(5)
```

Out[13]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Ma
358	359	30.111	4336	339	1	81	18	Male	No	
165	166	25.383	4527	367	4	46	11	Male	No	
283	284	49.927	6396	485	3	75	17	Female	No	
92	93	30.733	2832	249	4	51	13	Male	No	
184	185	158.889	11589	805	1	62	17	Female	No	

In [12]:
```
# example: number of cards
# without changing anything, it will be analyzed as numeric

reg3= smf.ols('Balance~Cards', data=credit).fit()
reg3.summary()

#not a perfect example since there are too many levels, but you get the idea
# then we see compared to having just one card, if there's significant chang
```

Out[12]:

OLS Regression Results

Dep. Variable:	Balance	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	0.005
Method:	Least Squares	**F-statistic:**	2.997
Date:	Mon, 26 Oct 2020	**Prob (F-statistic):**	0.0842
Time:	16:22:31	**Log-Likelihood:**	-3017.9
No. Observations:	400	**AIC:**	6040.
Df Residuals:	398	**BIC:**	6048.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	434.2861	54.569	7.958	0.000	327.006	541.566
Cards	28.9869	16.743	1.731	0.084	-3.929	61.903

Omnibus:	28.964	**Durbin-Watson:**	1.957
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	26.603
Skew:	0.566	**Prob(JB):**	1.67e-06
Kurtosis:	2.437	**Cond. No.**	8.37

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [13]:

```
#change it to categorical
reg3= smf.ols('Balance~C(Cards)', data=credit).fit()
reg3.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	Balance	**R-squared:**	0.023
Model:	OLS	**Adj. R-squared:**	0.003
Method:	Least Squares	**F-statistic:**	1.144
Date:	Mon, 26 Oct 2020	**Prob (F-statistic):**	0.332
Time:	16:22:31	**Log-Likelihood:**	-3014.7
No. Observations:	400	**AIC:**	6047.
Df Residuals:	391	**BIC:**	6083.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	531.1373	64.286	8.262	0.000	404.748	657.527
C(Cards)[T.2]	-58.1720	77.236	-0.753	0.452	-210.023	93.679
C(Cards)[T.3]	-39.0742	77.663	-0.503	0.615	-191.763	113.615
C(Cards)[T.4]	45.2794	84.024	0.539	0.590	-119.916	210.475
C(Cards)[T.5]	-8.1373	101.645	-0.080	0.936	-207.977	191.702
C(Cards)[T.6]	149.6809	152.622	0.981	0.327	-150.381	449.743
C(Cards)[T.7]	497.6127	238.379	2.087	0.037	28.947	966.278
C(Cards)[T.8]	106.8627	463.574	0.231	0.818	-804.547	1018.272
C(Cards)[T.9]	-149.1373	463.574	-0.322	0.748	-1060.547	762.272

Omnibus:	28.038	**Durbin-Watson:**	1.928
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	26.387
Skew:	0.568	**Prob(JB):**	1.86e-06
Kurtosis:	2.459	**Cond. No.**	22.5

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

Ex: Y = cal in coffee

X = level of sugar: (low, med, high)

Dummy

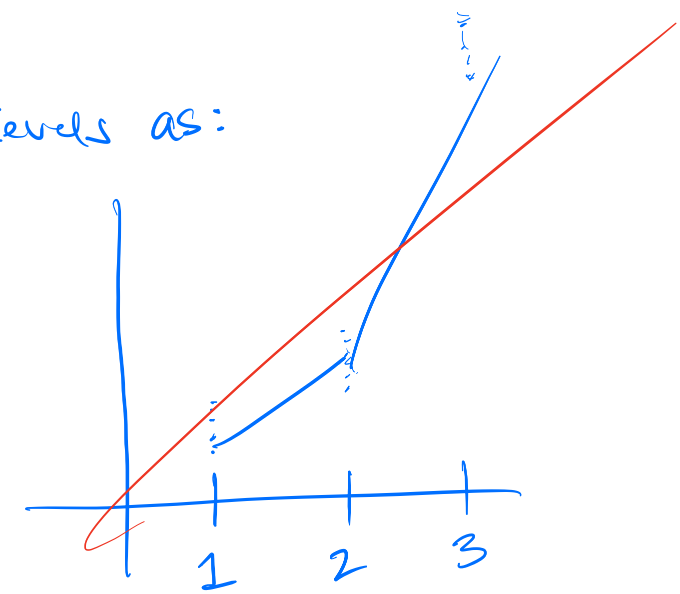
	X_med	X_high
$\Rightarrow X \longrightarrow$	0	1

low is baseline

Alternatively,

what if I encode the levels as:

$$X = \begin{cases} 1 \text{ if low} \\ 2 \text{ if med} \\ 3 \text{ if high} \end{cases}$$



$$y = \beta_0 + \underline{\beta_1} X + \varepsilon_i$$

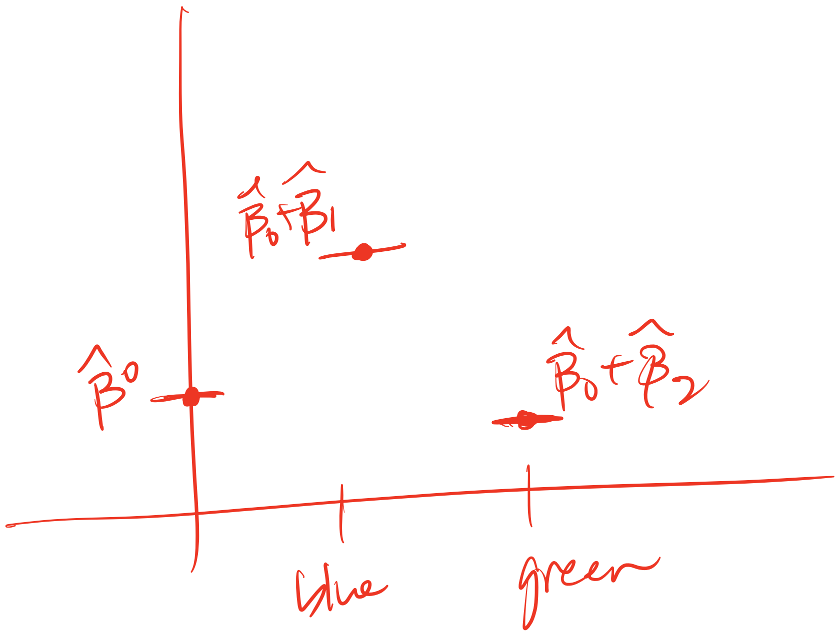
This type of encoding is ok

if you are ok with the assumption

that there is a true linear effect

btw category levels & y.

EX: Eye Color



2 cat:

eye & gender

X_1 X_2

X_{1} — blue

X_{1} — green

X_{2} — M

X_{2} — NB

$$Y_i = \beta_0 + \beta_1 X_{1\,blue} + \beta_2 X_{1\,green} + \beta_3 X_{2-M} + \beta_4 X_{2-NB} + \varepsilon_i$$

only main effects ⇒ no interactions

What is an interaction?

Y = stylishness

X_1 = wearing casual pink shirt

X_2 = wearing business pants

There is potential for mismatch → we need an interaction:

$$Y_i = \beta_0 + \beta_1 X_{1\text{-shirt}} + \beta_2 X_{2\text{-pants}} + \beta_3 X_{1\text{-shirt}} X_{2\text{-pants}} + \varepsilon_i$$

$\hat{\beta}_1 = 2$

$\hat{\beta}_2 = 1$

$\hat{\beta}_3 = -6$