COSC 3337 : Data Science I

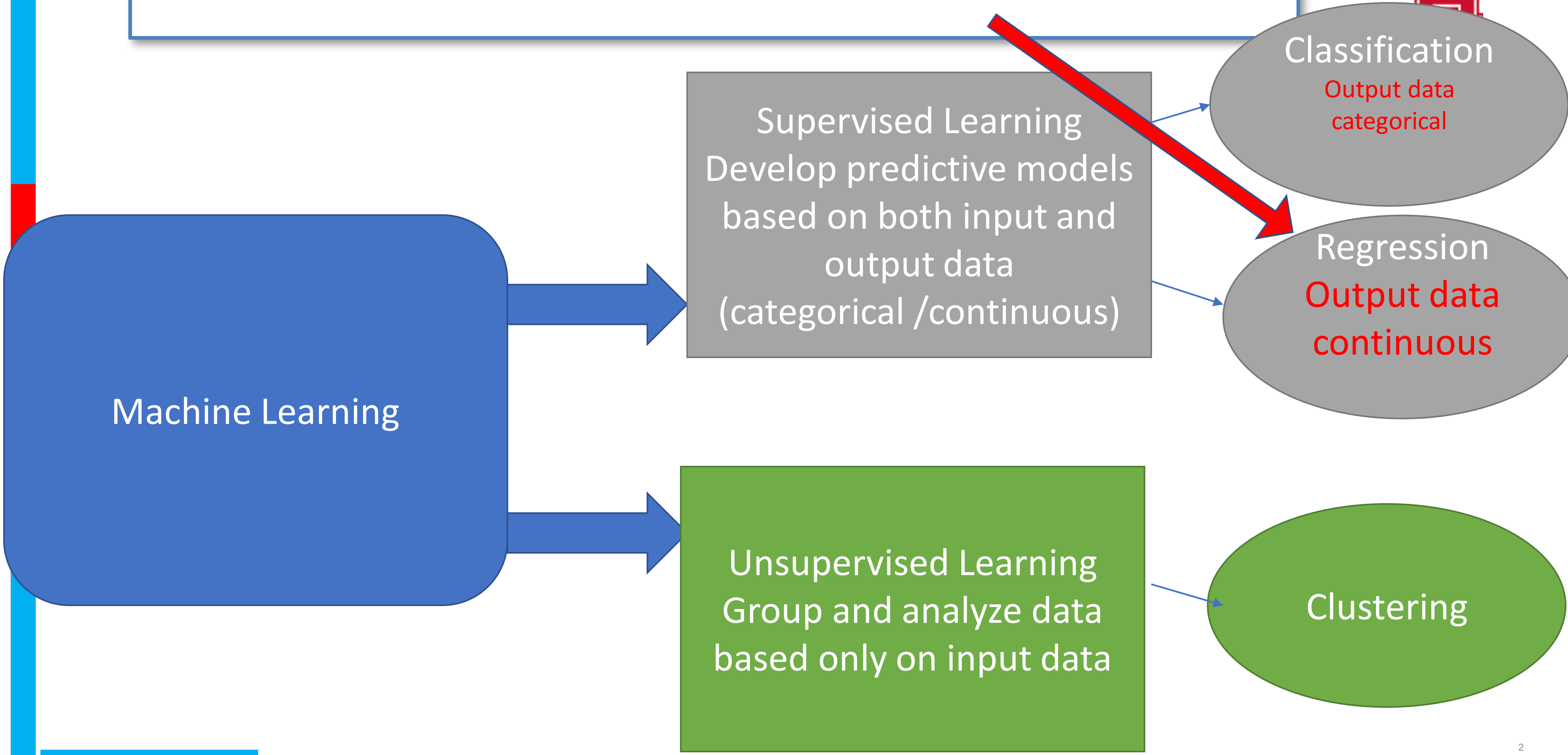
N. Rizk

College of Natural and Applied Sciences

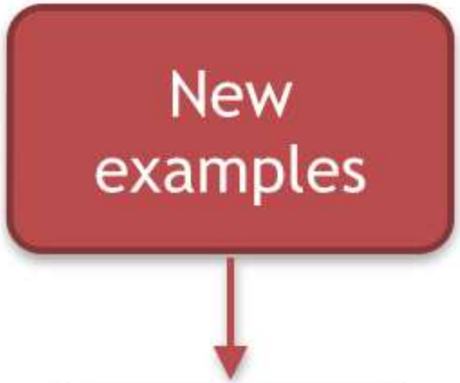
Department of Computer Science

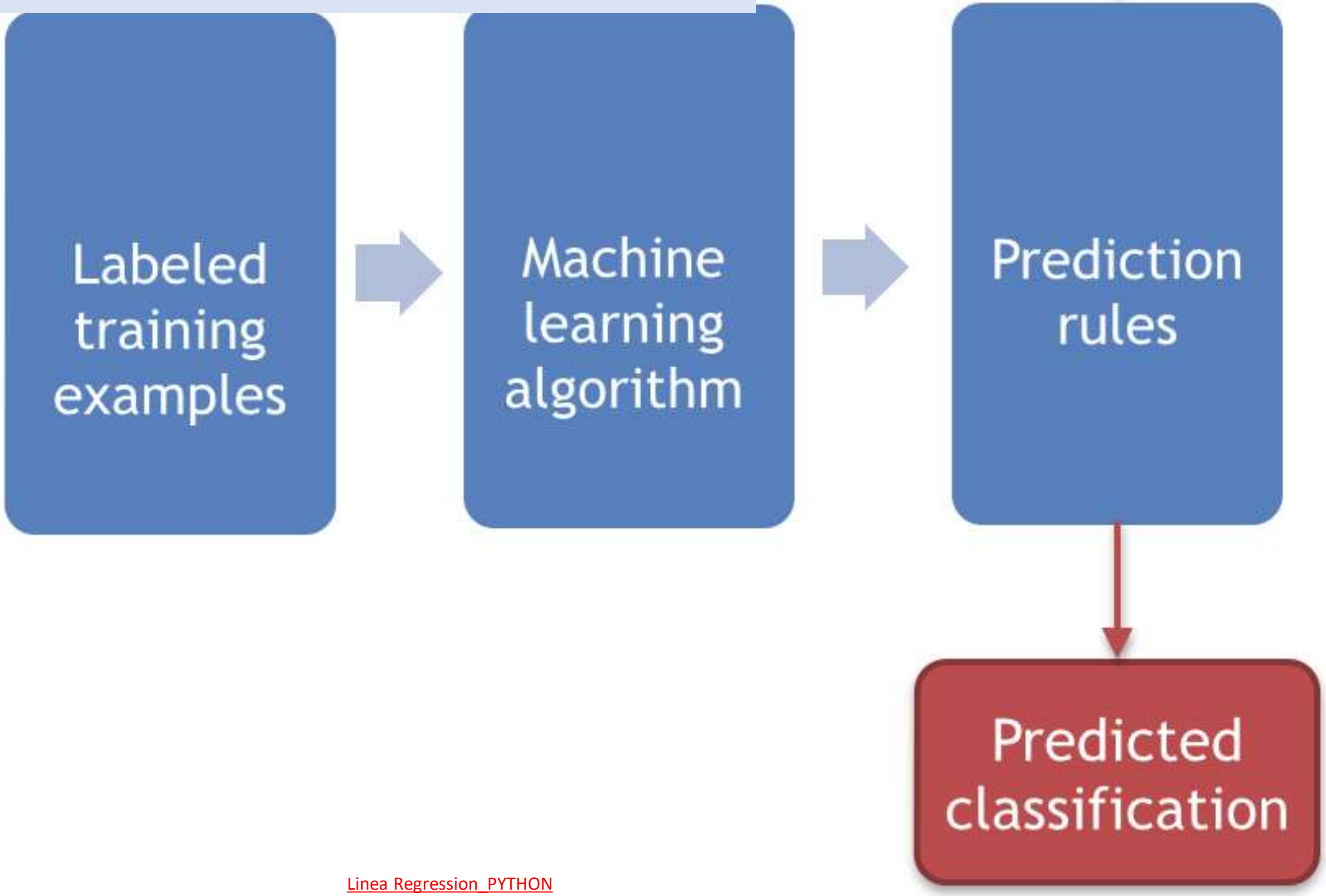
University of Houston

Focus


Machine Learning
Supervised Learning
Develop predictive models
based on both input and
output data
(categorical /continuous)
Classification
Output data
categorical
Regression
Output data
continuous
Unsupervised Learning
Group and analyze data
based only on input data
Clustering

Supervised Learning: Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as supervised learning problems





Introduction

- Regression problems are supervised learning problems in which the response is continuous
- Linear regression is a technique that is useful for regression problems.
- Classification problems are supervised learning problems in which the response is categorical
- Benefits of linear regression
  - widely used
  - runs fast
  - easy to use (not a lot of tuning required)
  - highly interpretable
  - basis for many other methods

Libraries

- Statsmodels
- scikit-learn

```
# imports
import pandas as pd
import seaborn as sns
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.cross_validation import
train_test_split
import numpy as np

# allow plots to appear directly in the notebook
%matplotlib inline
```

```
# read data into a DataFrame
data = pd.read_csv('advertising.csv', index_col=0)
data.head()

# shape of the DataFrame
data.shape

# visualize the relationship between the
features and the response using scatterplots
sns.pairplot(data,
x_vars=['TV','Radio','Newspaper'],
y_vars='Sales', size=7, aspect=0.7)

sns.pairplot(data)

sns.pairplot(data.dropna())

sns.pairplot(data, diag_kind="kde")
```

```python
sns.pairplot(data, diag_kind="kde", markers="+",
             plot_kws=dict(s=50, edgecolor="b", linewidth=1),
            diag_kws=dict(shade=True))
```

```python
sns.pairplot(data,
                 x_vars=["TV", "radio","newspaper"],
                y_vars=[ "sales"],size=7,aspect=0.7)
```

```python
sns.pairplot(data,
                 x_vars=["TV", "radio"],
                y_vars=["newspaper", "sales"])
```

Simple Linear Regression

- Simple linear regression is an approach for predicting a quantitative response using a single feature (or "predictor" or "input variable")
- It takes the following form:
- $y=\beta 0+\beta 1x$
- What does each term represent?

- y is the response
- x is the feature
- $\beta 0$ is the intercept
- $\beta 1$ is the coefficient for x

- $\beta 0$ and $\beta 1$ are called the model coefficients

- To create your model, you must "learn" the values of these coefficients. Once we've learned these coefficients, we can use the model to predict Sales.

Estimating ("Learning") Model Coefficients

- Coefficients are estimated using the least squares criterion
- In other words, we find the line (mathematically) which minimizes the sum of squared residuals (or "sum of squared errors"):

What elements are present in the diagram?

The black dots are the observed values of x and y
The blue line is our least squares line
The red lines are the residuals, which are the distances between the observed values and the least squares line
How do the model coefficients relate to the least squares line?

$\beta 0$ is the intercept (the value of y when x =0)
$\beta 1$ is the slope (the change in y divided by change in x)
Here is a graphical depiction of those calculations:

EXAMPLE

- Boston House Prices dataset
- ===========================
- Data Set Characteristics:
- :Number of Instances: 506
- :Number of Attributes: 13 numeric/categorical predictive
- :Median Value (attribute 14) is usually the target
- :Attribute Information (in order):
- - CRIM per capita crime rate by town
- - ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- - INDUS proportion of non-retail business acres per town
- - CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- - NOX nitric oxides concentration (parts per 10 million)
- - RM average number of rooms per dwelling
- - AGE proportion of owner-occupied units built prior to 1940
- - DIS weighted distances to five Boston employment centres
- - RAD index of accessibility to radial highways
- - TAX full-value property-tax rate per $10,000
- - PTRATIO pupil-teacher ratio by town
- - B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- - LSTAT % lower status of the population
- - MEDV Median value of owner-occupied homes in $1000's

- :Missing Attribute Values: None
- :Creator: Harrison, D. and Rubinfeld, D.L.

In [1]:
```
import statsmodels.api as sm
```

In [2]:
```
from sklearn import datasets ## imports datasets from scikit-learn
data = datasets.load_boston() ## loads Boston dataset from datasets library
```

In [3]:
```
import numpy as np
import pandas as pd
# define the data/predictors as the pre-set feature names
df = pd.DataFrame(data.data, columns=data.feature_names)

# Put the target (housing value -- MEDV) in another DataFrame
target = pd.DataFrame(data.target, columns=["MEDV"])
```

data.feature_names and data.target would print the column names of the independent variables and the dependent variable, respectively

In [4]:
```
## Without a constant

import statsmodels.api as sm

X = df["RM"]
y = target["MEDV"]

# Note the difference in argument order
model = sm.OLS(y, X).fit()
predictions = model.predict(X) # make the predictions by the model

# Print out the statistics
model.summary()
```

RM average number of rooms per dwelling

Out[4]:

OLS Regression Results

Dep. Variable:	MEDV	R-squared:	0.901
Model:	OLS	Adj. R-squared:	0.901
Method:	Least Squares	F-statistic:	4615.
Date:	Sun, 02 Sep 2018	Prob (F-statistic):	3.74e-256
Time:	21:29:59	Log-Likelihood:	-1747.1
No. Observations:	506	AIC:	3496.
Df Residuals:	505	BIC:	3500.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
RM	3.6534	0.054	67.930	0.000	3.548	3.759

Omnibus:	83.295	Durbin-Watson:	0.493
Prob(Omnibus):	0.000	Jarque-Bera (JB):	152.507
Skew:	0.955	Prob(JB):	7.65e-34
Kurtosis:	4.894	Cond. No.	1.00

MEDV Median value of owner-occupied homes in $1000's

house value/price data as a target variable and 13 other variables are set as predictors.

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

Linea Regression_PYTHON

OLS Regression Results

Dep. Variable:	MEDV	R-squared:	0.901
Model:	OLS	Adj. R-squared:	0.901
Method:	Least Squares	F-statistic:	4615.
Date:	Sun, 02 Sep 2018	Prob (F-statistic):	3.74e-256
Time:	21:29:59	Log-Likelihood:	-1747.1
No. Observations:	506	AIC:	3496.
Df Residuals:	505	BIC:	3500.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
RM	3.6534	0.054	67.930	0.000	3.548	3.759

Omnibus:	83.295	Durbin-Watson:	0.493
Prob(Omnibus):	0.000	Jarque-Bera (JB):	152.507
Skew:	0.955	Prob(JB):	7.65e-34
Kurtosis:	4.894	Cond. No.	1.00

1-what's the dependent variable and the model and the method.
2-OLS stands for Ordinary Least Squares and the method "Least Squares" means that we're trying to fit a regression line that woul minimize the square of distance from the regression line
3- number of observations.
4-Df of residuals and models relates to the degrees of freedom — "the number of values in the final calculation of a statistic that are free to vary."
5-The coefficient of 3.634 means that as the RM variable increase the predicted value of MDEV increases by 3.634.
6-the R-squared — the percentage of variance the model explains,
7-the standard error (is the standard deviation of the sampling distribution of a statistic, most commonly of the mean);
8- The t scores and p-values, for hypothesis test — the RM has statistically significant p-value; there is a 95% confidence intervals for the RM (meaning we predict at a 95% percent confidence that the value of RM is between 3.548 to 3.759).

```python
import statsmodels.api as sm # import statsmodels

X = df["RM"] ## X usually means our input variables (or independent variables)
y = target["MEDV"] ## Y usually means our output/dependent variable
X = sm.add_constant(X) ## let's add an intercept (beta_0) to our model

# Note the difference in argument order
model = sm.OLS(y, X).fit() ## sm.OLS(output, input)
predictions = model.predict(X)

# Print out the statistics
model.summary()
```

OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.484
Model:	OLS	**Adj. R-squared:**	0.483
Method:	Least Squares	**F-statistic:**	471.8
Date:	Sun, 02 Sep 2018	**Prob (F-statistic):**	2.49e-74
Time:	21:42:24	**Log-Likelihood:**	-1673.1
No. Observations:	506	**AIC:**	3350.
Df Residuals:	504	**BIC:**	3359.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-34.6706	2.650	-13.084	0.000	-39.877	-29.465
RM	9.1021	0.419	21.722	0.000	8.279	9.925

Omnibus:	102.585	**Durbin-Watson:**	0.684
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	612.449
Skew:	0.726	**Prob(JB):**	1.02e-133
Kurtosis:	8.190	**Cond. No.**	58.4

OLS Regression Results

Dep. Variable:	MEDV	R-squared:	0.484
Model:	OLS	Adj. R-squared:	0.483
Method:	Least Squares	F-statistic:	471.8
Date:	Sun, 02 Sep 2018	Prob (F-statistic):	2.49e-74
Time:	21:42:24	Log-Likelihood:	-1673.1
No. Observations:	506	AIC:	3350.
Df Residuals:	504	BIC:	3359.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-34.6706	2.650	-13.084	0.000	-39.877	-29.465
RM	9.1021	0.419	21.722	0.000	8.279	9.925

Omnibus:	102.585	Durbin-Watson:	0.684
Prob(Omnibus):	0.000	Jarque-Bera (JB):	612.449
Skew:	0.726	Prob(JB):	1.02e-133
Kurtosis:	8.190	Cond. No.	58.4

Interpreting the Table — With the constant term the coefficients are different. Without a constant we are forcing our model to go through the origin, but now we have a y-intercept at -34.67. We also changed the slope of the RM predictor from 3.634 to 9.1021.

fitting a regression model with more than one variable — (using RM and LSTAT)

```
X = df[["RM", "LSTAT"]]
y = target["MEDV"]
model = sm.OLS(y, X).fit()
predictions = model.predict(X)
model.summary()
```

OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.948
Model:	OLS	**Adj. R-squared:**	0.948
Method:	Least Squares	**F-statistic:**	4637.
Date:	Sun, 02 Sep 2018	**Prob (F-statistic):**	0.00
Time:	21:46:45	**Log-Likelihood:**	-1582.9
No. Observations:	506	**AIC:**	3170.
Df Residuals:	504	**BIC:**	3178.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
RM	4.9069	0.070	69.906	0.000	4.769	5.045
LSTAT	-0.6557	0.031	-21.458	0.000	-0.716	-0.596

Omnibus:	145.153	**Durbin-Watson:**	0.834
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	442.157
Skew:	1.351	**Prob(JB):**	9.70e-97
Kurtosis:	6.698	**Cond. No.**	4.72

Interpreting the Output

1-higher R-squared value — 0.948, meaning that this model explains 94.8% of the variance in our dependent variable. Whenever we add variables to a regression model, R^2 will be higher.

2- Both RM and LSTAT are statistically significant in predicting (or estimating) the median house value;

3- as RM increases by 1, MEDV will increase by 4.9069 and when LSTAT increases by 1, MEDV will decrease by -0.6557. (LSTAT is the percentage of lower status of the population)

Linea Regression_PYTHON

Linear Regression in SKLearn

```
from sklearn import linear_model
```

```
from sklearn import datasets ## imports datasets from scikit-learn
data = datasets.load_boston() ## loads Boston dataset from datasets library
```

```
# define the data/predictors as the pre-set feature names
df = pd.DataFrame(data.data, columns=data.feature_names)

# Put the target (housing value -- MEDV) in another DataFrame
target = pd.DataFrame(data.target, columns=["MEDV"])
```

```
X = df
y = target["MEDV"]
```

```
lm = linear_model.LinearRegression()
model = lm.fit(X,y)
```

```
predictions = lm.predict(X)
print(predictions)[0:5]
```

```
[30.00821269 25.0298606  30.5702317  28.60814055 27.94288232 25.25940048
```

The R^2 score of the model is 0.740(the percentage of explained variance of the predictions)

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

```
0.7406077428649428
```

```
lm.coef_
```

```
array([-1.07170557e-01,  4.63952195e-02,  2.08602395e-02,  2.68856140e+00,
       -1.77957587e+01,  3.80475246e+00,  7.51061703e-04, -1.47575880e+00,
        3.05655038e-01, -1.23293463e-02, -9.53463555e-01,  9.39251272e-03,
       -5.25466633e-01])
```

```
lm.intercept_
```

```
36.4911032803614
```

```
import pandas as pd
import numpy as np
import itertools
from itertools import chain, combinations
import statsmodels.formula.api as smf
import scipy.stats as scipystats
import statsmodels.api as sm
import statsmodels.stats.stattools as stools
import statsmodels.stats as stats
from statsmodels.graphics.regressionplots import *
import matplotlib.pyplot as plt
import seaborn as sns
import copy
from sklearn.cross_validation import train_test_split
import math
import time
```

```
elemapi = pd.read_csv('elemapi.csv')

print (elemapi[['api00', 'acs_k3', 'meals',  'full']].head())
```

	api00	acs_k3	meals	full
0	693	16.0	67.0	76.0
1	570	15.0	92.0	79.0
2	546	17.0	97.0	68.0
3	571	20.0	90.0	87.0
4	478	18.0	89.0	87.0

Let's dive right in and perform a regression analysis using the variables api00, acs_k3, meals and full.

These measure:

1- the academic performance of the school (api00)

2- the average class size in kindergarten through 3rd grade (acs_k3)

3- the percentage of students receiving free meals (meals) - which is an indicator of poverty, and the percentage of teachers who have full teaching credentials (full).

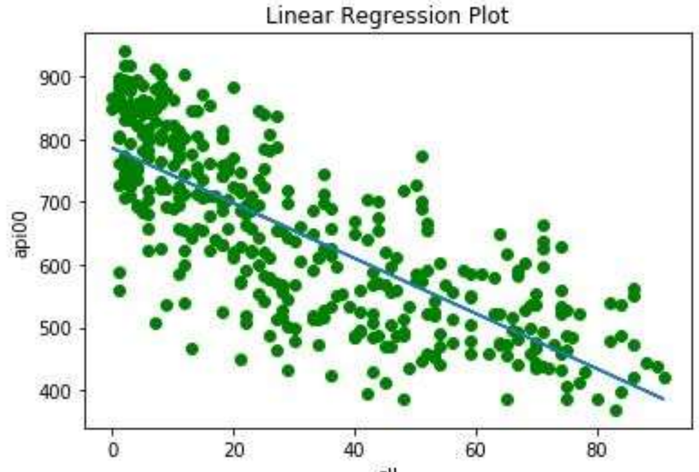
We expect that better academic performance would be associated with lower class size, fewer students receiving free meals, and a higher percentage of teachers having full teaching credentials.

1-linear regression using formulas

```
print ('-'*40 + ' smf.ols in R formula ' + '-'*40 + '\n')
lm = smf.ols(formula = 'api00 ~ ell', data = elemapi).fit()
print( lm.summary())
plt.figure()
plt.scatter(elemapi.ell, elemapi.api00, c = 'g')
plt.plot(elemapi.ell, lm.params[0] + lm.params[1] * elemapi.ell)
plt.xlabel('ell')
plt.ylabel('api00')
plt.title("Linear Regression Plot")

print (elemapi[['api00', 'acs_k3', 'meals',  'full']].head())
```

```
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.589
Model:                            OLS   Adj. R-squared:                  0.588
Method:                 Least Squares   F-statistic:                     571.0
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           6.54e-79
Time:                        12:04:50   Log-Likelihood:                -2372.1
No. Observations:                 400   AIC:                             4748.
Df Residuals:                     398   BIC:                             4756.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    785.8903      7.370    106.638      0.000     771.402     800.379
ell           -4.3961      0.184    -23.895      0.000      -4.758      -4.034
==============================================================================
Omnibus:                        7.668   Durbin-Watson:                   1.461
Prob(Omnibus):                  0.022   Jarque-Bera (JB):                7.264
Skew:                          -0.283   Prob(JB):                       0.0265
Kurtosis:                       2.660   Cond. No.                         64.7
==============================================================================

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

	api00	acs_k3	meals	full
0	693	16.0	67.0	76.0
1	570	15.0	92.0	79.0
2	546	17.0	97.0	68.0
3	571	20.0	90.0	87.0
4	478	18.0	89.0	87.0



1-linear regression using formulas

Linea Regression_PYTHON

2-Simple linear regression using input directly

```
print ('-'*40 + ' sm.OLS with direct input data ' + '-'*40 + '\n')
lm2 = sm.OLS(elemapi['api00'], sm.add_constant(elemapi[['ell']])).fit()
print (lm2.summary())
```

```
---------------------------------------- sm.OLS with direct input data ----------------------------------------

                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.589
Model:                            OLS   Adj. R-squared:                  0.588
Method:                 Least Squares   F-statistic:                     571.0
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           6.54e-79
Time:                        12:08:58   Log-Likelihood:                -2372.1
No. Observations:                 400   AIC:                             4748.
Df Residuals:                     398   BIC:                             4756.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        785.8903      7.370    106.638      0.000     771.402     800.379
ell           -4.3961      0.184    -23.895      0.000      -4.758      -4.034
==============================================================================
Omnibus:                        7.668   Durbin-Watson:                   1.461
Prob(Omnibus):                  0.022   Jarque-Bera (JB):                7.264
Skew:                          -0.283   Prob(JB):                       0.0265
Kurtosis:                       2.660   Cond. No.                         64.7
==============================================================================
```

1-Multiple linear regression using formulas

```
lm = smf.ols(formula = 'api00 ~ acs_k3 + meals + full', data = elemapi).fit()
print (lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.674
Model:                            OLS   Adj. R-squared:                  0.671
Method:                 Least Squares   F-statistic:                     213.4
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           5.73e-75
Time:                        12:12:57   Log-Likelihood:                -1744.6
No. Observations:                 313   AIC:                             3497.
Df Residuals:                     309   BIC:                             3512.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    906.7392     28.265     32.080      0.000     851.123     962.355
acs_k3        -2.6815      1.394     -1.924      0.055      -5.424       0.061
meals         -3.7024      0.154    -24.038      0.000      -4.005      -3.399
full           0.1086      0.091      1.197      0.232      -0.070       0.287
==============================================================================
Omnibus:                        2.012   Durbin-Watson:                   1.467
Prob(Omnibus):                  0.366   Jarque-Bera (JB):                2.070
Skew:                           0.162   Prob(JB):                        0.355
Kurtosis:                       2.767   Cond. No.                         769.
==============================================================================
```

1-Multiple linear regression using input (drop missing values)

```
data = elemapi[['api00', 'acs_k3', 'meals',  'full']]
data = data.dropna(axis = 0, how = 'any')

lm2 = sm.OLS(data['api00'], sm.add_constant(data[['acs_k3', 'meals',  'full']])).fit()
print( lm2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.674
Model:                            OLS   Adj. R-squared:                  0.671
Method:                 Least Squares   F-statistic:                     213.4
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           5.73e-75
Time:                        12:19:50   Log-Likelihood:                -1744.6
No. Observations:                 313   AIC:                             3497.
Df Residuals:                     309   BIC:                             3512.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        906.7392     28.265     32.080      0.000     851.123     962.355
acs_k3        -2.6815      1.394     -1.924      0.055      -5.424       0.061
meals         -3.7024      0.154    -24.038      0.000      -4.005      -3.399
full           0.1086      0.091      1.197      0.232      -0.070       0.287
==============================================================================
Omnibus:                        2.012   Durbin-Watson:                   1.467
Prob(Omnibus):                  0.366   Jarque-Bera (JB):                2.070
Skew:                           0.162   Prob(JB):                        0.355
Kurtosis:                       2.767   Cond. No.                         769.
```

Data Analysis

Numeric Data Analysis

1. if there is any missing data
2. what is the distribution of the data, and its visualization like histogram and box-plot
3. what is the five numbers: min, 25 percentile, median, 75 percentile, and the max
4. mean, stdev, length
5. the correlation between the data
6. futuremore, is there any outliers in the data?
7. other plots: pairwise scatter plot, kernal density plot

Categorical Data Analysis

1. is there any missing data?
2. how many unique values of the data? what is their frequency?

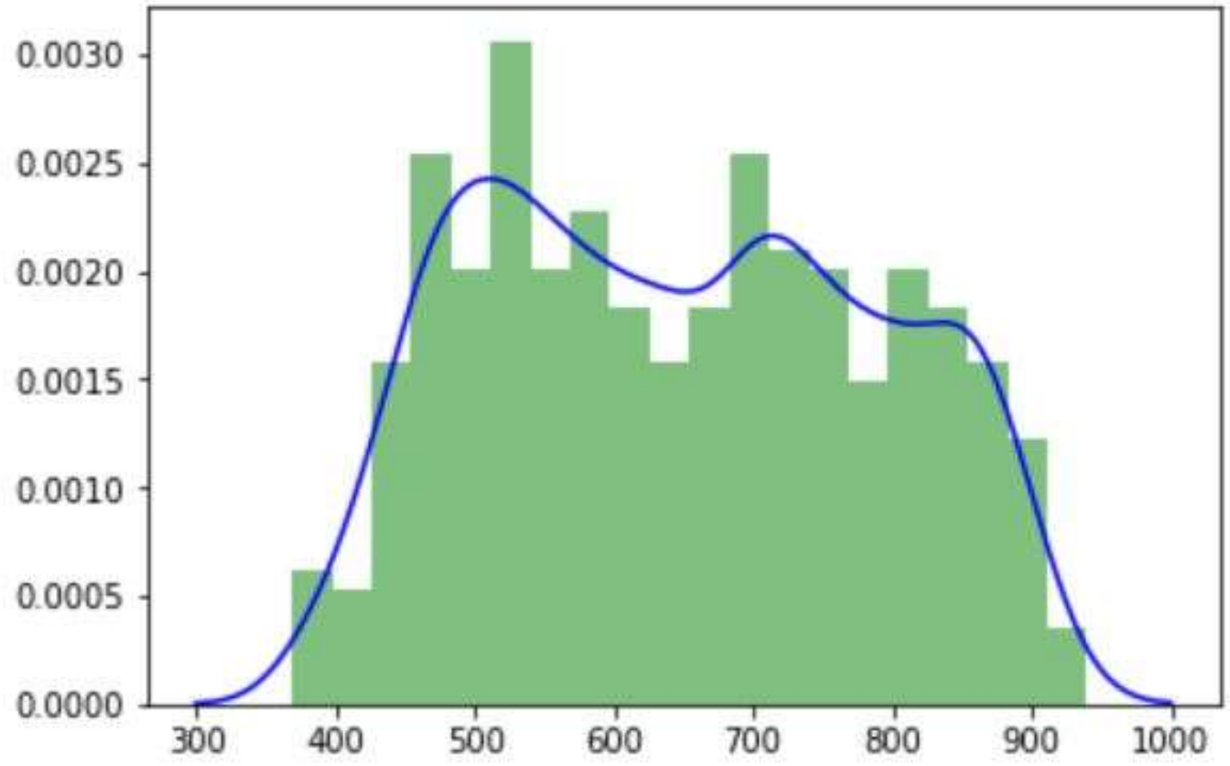
Feature selection !?

```python
sample_data = elemapi[['api00', 'acs_k3', 'meals',  'full']]
print (sample_data.describe())
```

	api00	acs_k3	meals	full
count	400.000000	398.000000	315.000000	400.000000
mean	647.622500	18.547739	71.993651	66.056800
std	142.248961	5.004933	24.385570	40.297926
min	369.000000	-21.000000	6.000000	0.420000
25%	523.750000	18.000000	57.000000	0.950000
50%	643.000000	19.000000	77.000000	87.000000
75%	762.250000	20.000000	93.000000	97.000000
max	940.000000	25.000000	100.000000	100.000000

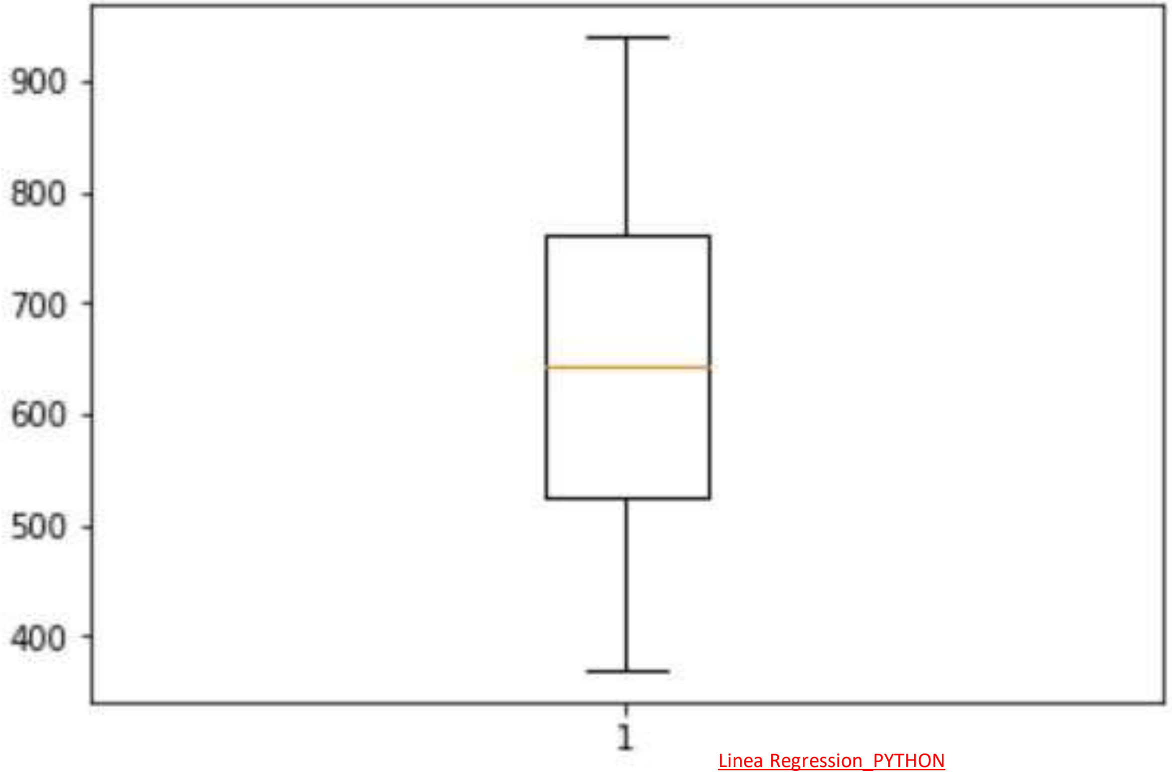
1. api00 and full does not have missing values and their length is 400.
2. acs_k3 has 398 non-missing values, so it has two missing data. meals has 315 non-missing values so it has 85 missings

```python
from scipy.stats import gaussian_kde

plt.hist(elemapi.api00, 20, normed = 1, facecolor = 'g', alpha = 0.5)

# add density plot
density = gaussian_kde(elemapi.api00)
xs = np.linspace(300, 1000, 500)
density.covariance_factor = lambda : .2
density._compute_covariance()
plt.plot(xs, density(xs), color = "b")
plt.show()
```



```python
plt.boxplot(elemapi.api00, 0, 'gD')
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1c070566dd8>,
  <matplotlib.lines.Line2D at 0x1c07056f2b0>],
 'caps': [<matplotlib.lines.Line2D at 0x1c07056f6d8>,
  <matplotlib.lines.Line2D at 0x1c07056fb00>],
 'boxes': [<matplotlib.lines.Line2D at 0x1c070566c88>],
 'medians': [<matplotlib.lines.Line2D at 0x1c07056ff28>],
 'fliers': [<matplotlib.lines.Line2D at 0x1c070576390>],
 'means': []}
```

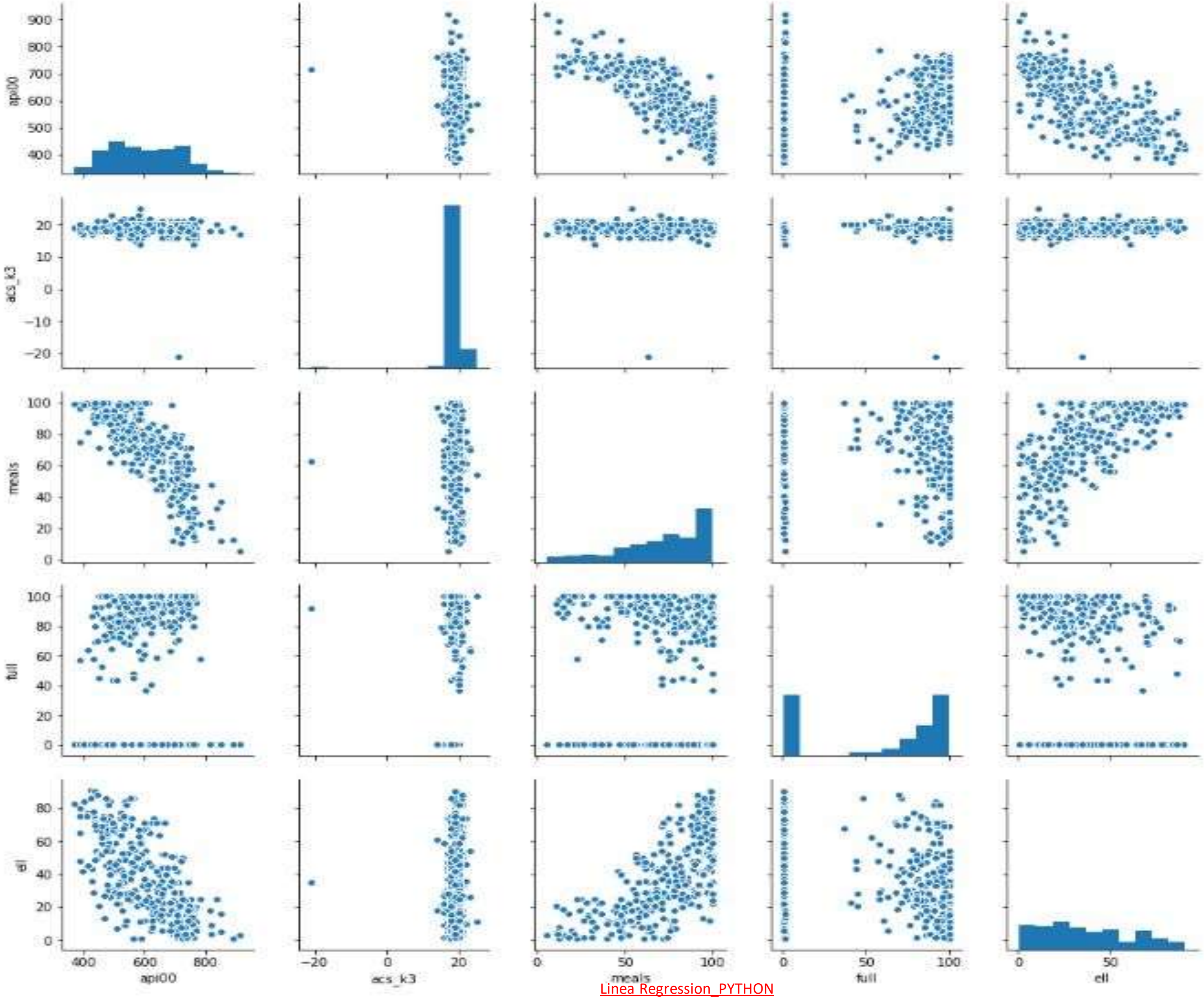


```python
# check correlation between each variable and api00
print (elemapi.corr().loc['api00', :].sort_values())
```

```
mealcat     -0.867260
meals       -0.819300
ell         -0.767634
not_hsg     -0.683255
emer        -0.582731
yr_rnd      -0.475440
hsg         -0.355809
enroll      -0.318172
mobility    -0.206410
growth      -0.108158
acs_k3      -0.095546
dnum        -0.011383
snum         0.216457
acs_46       0.232912
some_col     0.261527
full         0.411125
col_grad     0.527301
grad_sch     0.633241
avg_ed       0.792954
api99        0.985343
api00        1.000000
Name: api00, dtype: float64
```

```
sns.pairplot(elemapi[['api00', 'acs_k3', 'meals',  'full', 'ell']].dropna(how = 'any', axis = 0))
```

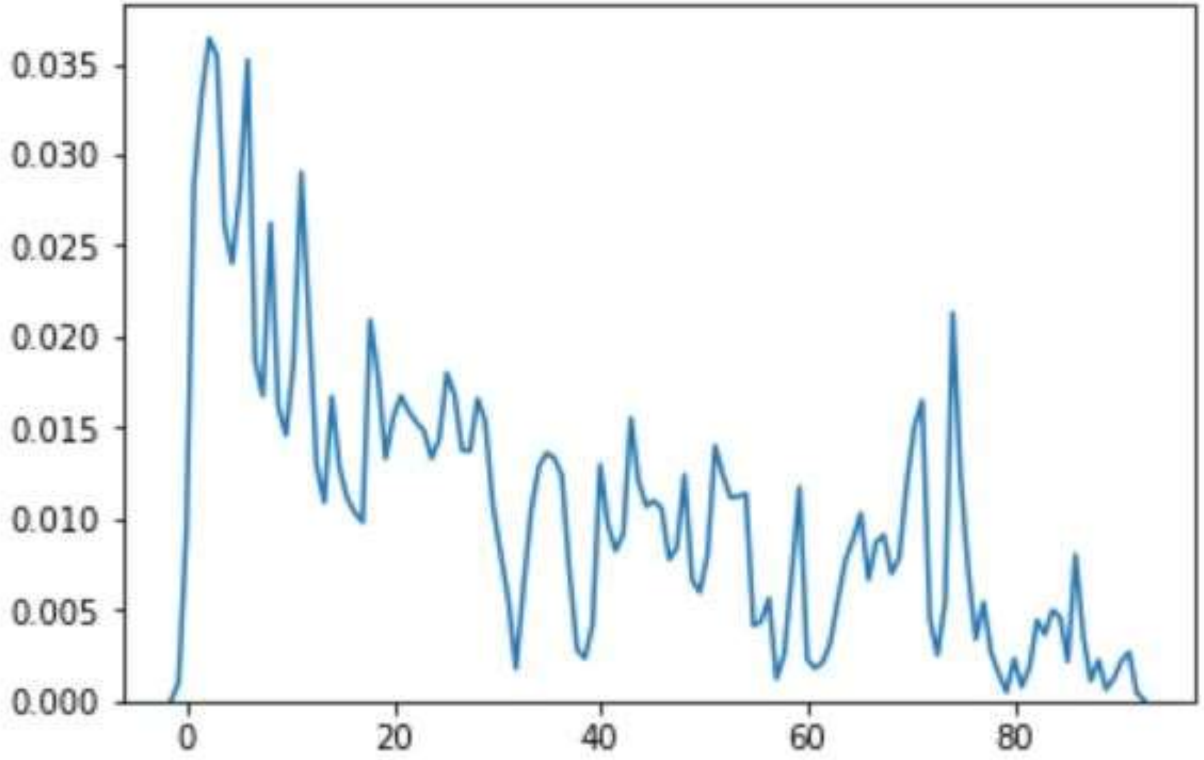
<seaborn.axisgrid.PairGrid at 0x1c070584fd0>



kernel density function =>the ell variable skewed to the right.

```
sns.kdeplot(np.array(elemapi.ell), bw=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0710cb390>
```



Categorical variable frequency

```
elemapi.acs_k3.value_counts(dropna = False).sort_index()
```

```
-21.0      3
-20.0      2
-19.0      1
 14.0      2
 15.0      1
 16.0     14
 17.0     20
 18.0     64
 19.0    143
 20.0     97
 21.0     40
 22.0      7
 23.0      3
 25.0      1
NaN        2
Name: acs_k3, dtype: int64
```

for acs_k3, there are about 143 data points equal to 19.

Regression Diagnostics

verifying that the data have met the regression assumptions

Issues that can arise during the analysis

- Linearity - the relationships between the predictors and the outcome variable should be linear
- Normality - the errors should be normally distributed - technically normality is necessary only for the t-tests to be valid, estimation of the coefficients only requires that the errors be identically and independently distributed
- Homogeneity of variance (homoscedasticity) - the error variance should be constant
- Independence - the errors associated with one observation are not correlated with the errors of any other observation
- Errors in variables - predictor variables are measured without error
- Model specification - the model should be properly specified (including all relevant variables, and excluding irrelevant variables)

Unusual and influential data

1. Outliers: In linear regression, an outlier is an observation with large residual. In other words, it is an observation whose dependent-variable value is unusual given its values on the predictor variables. An outlier may indicate a sample peculiarity or may indicate a data entry error or other problem.
2. Leverage: An observation with an extreme value on a predictor variable is called a point with high leverage. Leverage is a measure of how far an observation deviates from the mean of that variable. These leverage points can have an effect on the estimate of regression coefficients.
3. Influence: An observation is said to be influential if removing the observation substantially changes the estimate of coefficients. Influence can be thought of as the product of leverage and outlierness.

Data set crime

```python
import pandas as pd
import numpy as np
import itertools
from itertools import chain, combinations
import statsmodels.formula.api as smf
import scipy.stats as scipystats
import statsmodels.api as sm
import statsmodels.stats.stattools as stools
import statsmodels.stats as stats
from statsmodels.graphics.regressionplots import *
import matplotlib.pyplot as plt
import seaborn as sns
import copy
from sklearn.cross_validation import train_test_split
import math
import time

%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)


crime = pd.read_csv('crime.csv')
```

```python
crime.describe()
```

	low	murder	tc2009
count	48.000000	48.000000	48.000000
mean	-40.104167	4.495833	3388.716667
std	17.786248	2.354530	749.441028
min	-80.000000	0.900000	2026.200000
25%	-50.250000	2.675000	2778.825000
50%	-40.000000	4.650000	3442.850000
75%	-29.750000	5.925000	4018.575000
max	12.000000	12.300000	4562.200000

Advanced

Regression with Categorical Predictors

Regression with categorical predictors

1 Regression with a 0/1 variable

2 Regression with a 1/2 variable

3 Regression with a 1/2/3 variable

4 Regression with multiple categorical predictors

5 Categorical predictor with interactions

6 Continuous and categorical variables

7 Interactions of continuous by 0/1 categorical variables

8 Continuous and categorical variables, interaction with 1/2/3 variable

Regression with a 0/1 variable

```
import warnings

with warnings.catch_warnings():
    warnings.filterwarnings("ignore",category=DeprecationWarning)
```

```
import pandas as pd
import numpy as np
import itertools
from itertools import chain, combinations
import statsmodels.formula.api as smf
import scipy.stats as scipystats
import statsmodels.api as sm
import statsmodels.stats.stattools as stools
import statsmodels.stats as stats
from statsmodels.graphics.regressionplots import *
import matplotlib.pyplot as plt
import seaborn as sns
import copy
from sklearn.cross_validation import train_test_split
import math
import time

%matplotlib inline


elemapi2 = pd.read_csv('elemapi.csv')
```

```
elemapi2_sel = elemapi2.loc[:, ["api00", "some_col", "yr_rnd", "mealcat"]]
```

```python
print(elemapi2_sel.describe())

def cv_desc(df, var):
    return df[var].value_counts(dropna = False)

print ('\n'  )
print( cv_desc(elemapi2_sel, 'mealcat'))
print ('\n'  )
print (cv_desc(elemapi2_sel, 'yr_rnd'))
```

```
            api00    some_col     yr_rnd     mealcat
count  400.000000  400.000000  400.00000  400.000000
mean   647.622500   19.712500    0.23000    2.015000
std    142.248961   11.336938    0.42136    0.819423
min    369.000000    0.000000    0.00000    1.000000
25%    523.750000   12.000000    0.00000    1.000000
50%    643.000000   19.000000    0.00000    2.000000
75%    762.250000   28.000000    0.00000    3.000000
max    940.000000   67.000000    1.00000    3.000000


3    137
2    132
1    131
Name: mealcat, dtype: int64


0    308
1     92
Name: yr_rnd, dtype: int64
```

1. The variable api00 is a measure of the performance of the students.
2. The variable some_col is a continuous variable that measures the percentage of the parents in the school who have attended college.
3. The variable yr_rnd is a categorical variable that is coded 0 if the school is not year round, and 1 if year round.
4. The variable meals is the percentage of students who are receiving state sponsored free meals and can be used as an indicator of poverty. This was broken into 3 categories (to make equally sized groups) creating the variable mealcat. e.

Macro function gives codebook type information on a specific variable.

```
def codebook(df, var):
    title = "Codebook for " + str(var)
    unique_values = len(df[var].unique())
    max_v = df[var].max()
    min_v = df[var].min()
    n_miss = sum(pd.isnull(df[var]))
    mean = df[var].mean()
    stdev = df[var].std()
    print(pd.DataFrame({'title': title, 'unique values': unique_values, 'max value' : ma
    return

codebook(elemapi2_sel, 'api00')
```

```
                title  unique values  max value  min value  num of missing  \
0  Codebook for api00            271        940        369               0

       mean       stdev
0  647.6225  142.248961
```

```python
reg = smf.ols(formula = "api00 ~ yr_rnd", data = elemapi2_sel).fit()
reg.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.226
Model:	OLS	**Adj. R-squared:**	0.224
Method:	Least Squares	**F-statistic:**	116.2
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	5.96e-24
Time:	09:19:25	**Log-Likelihood:**	-2498.9
No. Observations:	400	**AIC:**	5002.
Df Residuals:	398	**BIC:**	5010.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	684.5390	7.140	95.878	0.000	670.503	698.575
yr_rnd	-160.5064	14.887	-10.782	0.000	-189.774	-131.239

Omnibus:	45.748	**Durbin-Watson:**	1.499
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13.162
Skew:	0.006	**Prob(JB):**	0.00139
Kurtosis:	2.111	**Cond. No.**	2.53

Regression with a 0/1 variable (categorical dummy variable)

api00 = Intercept + Byr_rnd * yr_rnd

api00 = 684.539 + -160.5064 * yr_rnd

yr_rnd is 0

api00 = constant + 0 * Byr_rnd

api00 = 684.539 + 0 * -160.5064 api00

= 684.539

yr_rnd is 1

api00 = constant + 1 * Byr_rnd

api00 = 684.539 + 1 * -160.5064

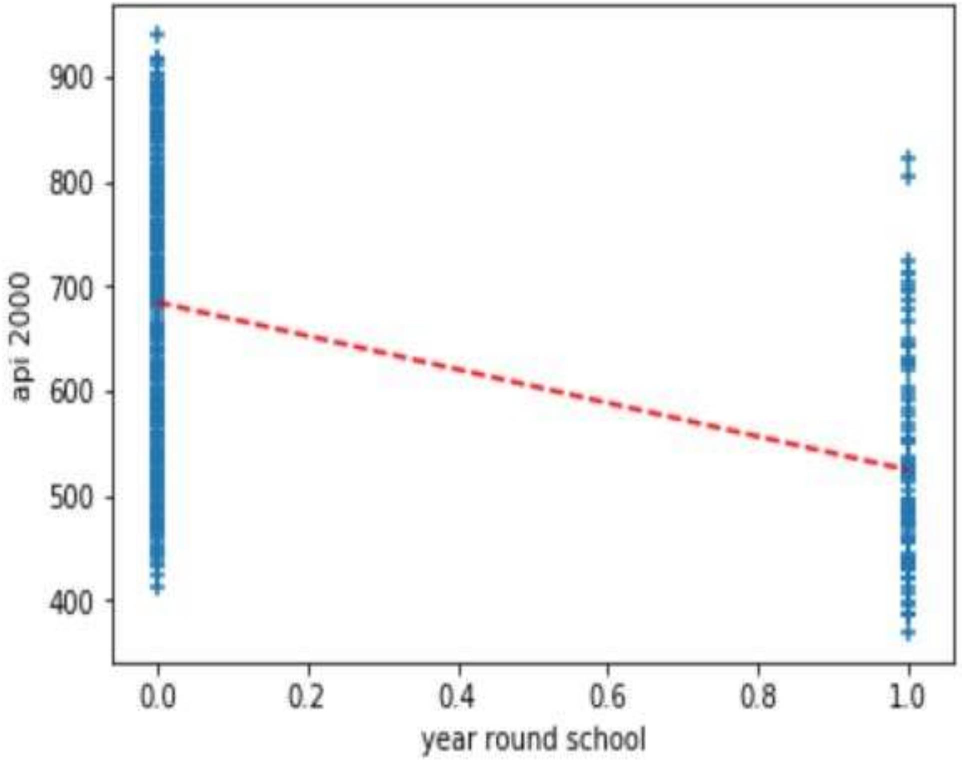
api00 = 524.0326

predicted value for the year round schools is 524.032

Linea Regression_PYTHON

Predicted values to the mean api00 scores for the year-round and non-year-round students

```
plt.scatter(elemapi2_sel.yr_rnd, elemapi2_sel.api00, marker = "+")
plt.plot([0, 1], [np.mean(elemapi2_sel.query('yr_rnd == 0').api00), np.mean(elemapi2_sel.query('yr_rnd == 1').api00)], 'r--')
plt.ylabel("api 2000")
plt.xlabel("year round school")
plt.show()
```



```
elemapi2_sel["yr_rnd_c"] = elemapi2_sel.yr_rnd.map({0: "No", 1: "Yes"})
elemapi2_sel["mealcat_c"] = elemapi2_sel.mealcat.map({1: "0-46% free meals", 2: "47-80% free meals", 3: "1-100% free meals"})

elemapi2_sel_group = elemapi2_sel.groupby("yr_rnd_c")
elemapi2_sel_group.api00.agg([np.mean, np.std])
```

yr_rnd_c	mean	std
No	684.538961	132.112534
Yes	524.032609	98.916043

For the non-year-round schools, their mean is the same as the intercept (684.539). The coefficient for yr_rnd is the amount we need to add to get the mean for the year-round schools, i.e., we need to add -160.5064 to get 524.0326, the mean for the non year-round schools.

➔Byr_rnd is the mean api00 score for the year-round schools minus the mean api00 score for the non year-round schools, i.e., mean(year-round) - mean(non year-round).

The t value below is the same as the t value for yr_rnd in the regression.

```
# pooled ttest, assume equal population variance
print( scipystats.ttest_ind(elemapi2_sel.query('yr_rnd == 0').api00,
elemapi2_sel.query('yr_rnd == 1').api00))

# does not assume equal variance
print (scipystats.ttest_ind(elemapi2_sel.query('yr_rnd == 0').api00,
elemapi2_sel.query('yr_rnd == 1').api00, equal_var = False))
```

Ttest_indResult(statistic=10.781500136400451, pvalue=5.964708112788805 6e-24)
Ttest_indResult(statistic=12.57105956566846, pvalue=5.29731480664924e-27)

Since a t-test is the same as doing an ANOVA (analysis of variance:test whether a number of me

```
print (scipystats.f_oneway(elemapi2_sel.query('yr_rnd == 0').api00, elemapi2_sel.query('yr_rnd
== 1').api00))
```

F_onewayResult(statistic=116.2407451912029, pvalue=5.96470811279079 9e-24)

If we square the t-value from the t-test, we get the same value as the F-value from ANOVA: 10.78^2=116.21 (with a little rounding error.)

```
elemapi2_sel["yr_rnd2"] = elemapi2_sel["yr_rnd"] + 1
reg = smf.ols("api00 ~ yr_rnd2", data = elemapi2_sel).fit()
reg.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.226
Model:	OLS	**Adj. R-squared:**	0.224
Method:	Least Squares	**F-statistic:**	116.2
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	5.96e-24
Time:	10:02:17	**Log-Likelihood:**	-2498.9
No. Observations:	400	**AIC:**	5002.
Df Residuals:	398	**BIC:**	5010.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	845.0453	19.353	43.664	0.000	806.998	883.093
yr_rnd2	-160.5064	14.887	-10.782	0.000	-189.774	-131.239

Omnibus:	45.748	**Durbin-Watson:**	1.499
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13.162
Skew:	0.006	**Prob(JB):**	0.00139
Kurtosis:	2.111	**Cond. No.**	6.23

Regression with a 1/2 variable

a copy of the variable yr_rnd called yr_rnd2 that is coded 1/2, 1=non year-round and 2=year-round.

Note that the coefficient for yr_rnd is the same as yr_rnd2. So, you can see that if you code yr_rnd as 0/1 or as 1/2, the regression coefficient works out to be the same. However the intercept (Intercept) is a bit less intuitive. When we used yr_rnd, the intercept was the mean for the non year-rounds. When using yr_rnd2, the intercept is the mean for the non year-rounds minus Byr_rnd2, i.e., 684.539 - (-160.506) = 845.045

The relationship between the amount of poverty and api scores?

```
elemapi2_sel_group = elemapi2_sel.groupby("mealcat_c")
elemapi2_sel_group.api00.agg([lambda x: x.shape[0], np.mean, np.std])
```

	<lambda>	mean	std
mealcat_c			
0-46% free meals	131	805.717557	65.668664
1-100% free meals	137	504.379562	62.727015
47-80% free meals	132	639.393939	82.135130

Regression with a 1/2/3 variable

use mealcat as a proxy for a measure of poverty. Mealcat (has three unique values on free meals. We can associate a value label to variable mealcat to make it more meaningful for us when we run python regression with mealcat.

```python
lm = smf.ols('api00 ~ mealcat', data = elemapi2_sel).fit()
lm.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.752
Model:	OLS	**Adj. R-squared:**	0.752
Method:	Least Squares	**F-statistic:**	1208.
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	1.29e-122
Time:	10:10:41	**Log-Likelihood:**	-2271.1
No. Observations:	400	**AIC:**	4546.
Df Residuals:	398	**BIC:**	4554.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	950.9874	9.422	100.935	0.000	932.465	969.510
mealcat	-150.5533	4.332	-34.753	0.000	-159.070	-142.037

Omnibus:	3.106	**Durbin-Watson:**	1.516
Prob(Omnibus):	0.212	**Jarque-Bera (JB):**	3.112
Skew:	-0.214	**Prob(JB):**	0.211
Kurtosis:	2.943	**Cond. No.**	6.86

mealcat is not an interval variable. (need to create dummy variables). For example, in orde to create dummy variables for mealcat, we car do the following using sklearn to create dumm variables.

```
from sklearn import preprocessing
le_mealcat = preprocessing.LabelEncoder()
elemapi2_sel['mealcat_dummy'] =
le_mealcat.fit_transform(elemapi2_sel.mealcat)

elemapi2_sel.groupby('mealcat_dummy').size()

ohe = preprocessing.OneHotEncoder()
dummy = pd.DataFrame(ohe.fit_transform(elemapi2_sel.mealcat.reshape(-
1,1)).toarray(), columns = ["mealcat1", "mealcat2", "mealcat3"])
elemapi2_sel = pd.concat([elemapi2_sel, dummy], axis = 1)

lm = smf.ols('api00 ~ mealcat2 + mealcat3', data = elemapi2_sel).fit()
lm.summary()
```

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
Intercept	805.7176	6.169	130.599	0.000	793.589 817.846
mealcat2[0]	-83.1618	4.354	-19.099	0.000	-91.722 - 74.602
mealcat2[1]	-83.1618	4.354	-19.099	0.000	-91.722 - 74.602
mealcat3[0]	-150.6690	4.314	-34.922	0.000	-159.151 - 142.187
mealcat3[1]	-150.6690	4.314	-34.922	0.000	-159.151 - 142.187

```
lm = lm = smf.ols('api00 ~ C(mealcat)', data = elemapi2_sel).fit()
lm.summary()
```

Run regression with categorical variable directly

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.755
Model:	OLS	**Adj. R-squared:**	0.754
Method:	Least Squares	**F-statistic:**	611.1
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	6.48e-122
Time:	10:25:35	**Log-Likelihood:**	-2269.0
No. Observations:	400	**AIC:**	4544.
Df Residuals:	397	**BIC:**	4556.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	805.7176	6.169	130.599	0.000	793.589	817.846
C(mealcat)[T.2]	-166.3236	8.708	-19.099	0.000	-183.444	-149.203
C(mealcat)[T.3]	-301.3380	8.629	-34.922	0.000	-318.302	-284.374

Omnibus:	1.593	**Durbin-Watson:**	1.541
Prob(Omnibus):	0.451	**Jarque-Bera (JB):**	1.684
Skew:	-0.139	**Prob(JB):**	0.431
Kurtosis:	2.847	**Cond. No.**	3.76

Regression with two categorical predictors

```
lm = smf.ols('api00 ~ yr_rnd', data = elemapi2_sel).fit()
print (lm.summary())

print ('\n')

lm = smf.ols('api00 ~ mealcat1 + mealcat2', data = elemapi2_sel).fit()
print( lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.226
Model:                            OLS   Adj. R-squared:                  0.224
Method:                 Least Squares   F-statistic:                     116.2
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           5.96e-24
Time:                        10:28:57   Log-Likelihood:                -2498.9
No. Observations:                 400   AIC:                             5002.
Df Residuals:                     398   BIC:                             5010.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    684.5390      7.140     95.878      0.000     670.503     698.575
yr_rnd      -160.5064     14.887    -10.782      0.000    -189.774    -131.239
==============================================================================
Omnibus:                       45.748   Durbin-Watson:                   1.499
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               13.162
```

```python
lm = smf.ols('api00 ~ yr_rnd + C(mealcat)', data = elemapi2_sel).fit()
print( lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.767
Model:                            OLS   Adj. R-squared:                  0.765
Method:                 Least Squares   F-statistic:                     435.0
Date:                Mon, 03 Sep 2018   Prob (F-statistic):          6.40e-125
Time:                        10:34:47   Log-Likelihood:                -2258.6
No. Observations:                 400   AIC:                             4525.
Df Residuals:                     396   BIC:                             4541.
Df Model:                           3
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept         808.0131      6.040    133.777      0.000     796.139     819.888
C(mealcat)[T.2]  -163.7374      8.515    -19.229      0.000    -180.478    -146.997
C(mealcat)[T.3]  -281.6832      9.446    -29.821      0.000    -300.253    -263.113
yr_rnd            -42.9601      9.362     -4.589      0.000     -61.365     -24.555
===================================================================================
Omnibus:                        1.210   Durbin-Watson:                   1.584
Prob(Omnibus):                  0.546   Jarque-Bera (JB):                1.279
Skew:                          -0.086   Prob(JB):                        0.528
Kurtosis:                       2.783   Cond. No.                         4.17
==============================================================================
```

The effect of yr_rnd at each of the three levels of mealcat.

```
lm = smf.ols('api00 ~ C(yr_rnd) * C(mealcat)', data = elemapi2_sel).fit()
print (lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.769
Model:                            OLS   Adj. R-squared:                  0.766
Method:                 Least Squares   F-statistic:                     261.6
Date:                Mon, 03 Sep 2018   Prob (F-statistic):          9.19e-123
Time:                        10:45:35   Log-Likelihood:                -2257.5
No. Observations:                 400   AIC:                             4527.
Df Residuals:                     394   BIC:                             4551.
Df Model:                           5
Covariance Type:            nonrobust
====================================================================================================
                                       coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------------
Intercept                          809.6855      6.185    130.911      0.000     797.526     821.845
C(yr_rnd)[T.1]                     -74.2569     26.756     -2.775      0.006    -126.860     -21.654
C(mealcat)[T.2]                   -164.4120      8.877    -18.522      0.000    -181.864    -146.960
C(mealcat)[T.3]                   -288.1929     10.443    -27.597      0.000    -308.724    -267.662
C(yr_rnd)[T.1]:C(mealcat)[T.2]      22.5167     32.752      0.687      0.492     -41.873      86.907
C(yr_rnd)[T.1]:C(mealcat)[T.3]      40.7644     29.231      1.395      0.164     -16.704      98.233
==============================================================================
Omnibus:                        1.439   Durbin-Watson:                   1.583
Prob(Omnibus):                  0.487   Jarque-Bera (JB):                1.484
Skew:                          -0.096   Prob(JB):                        0.476
Kurtosis:                       2.771   Cond. No.                         16.4
==============================================================================
```

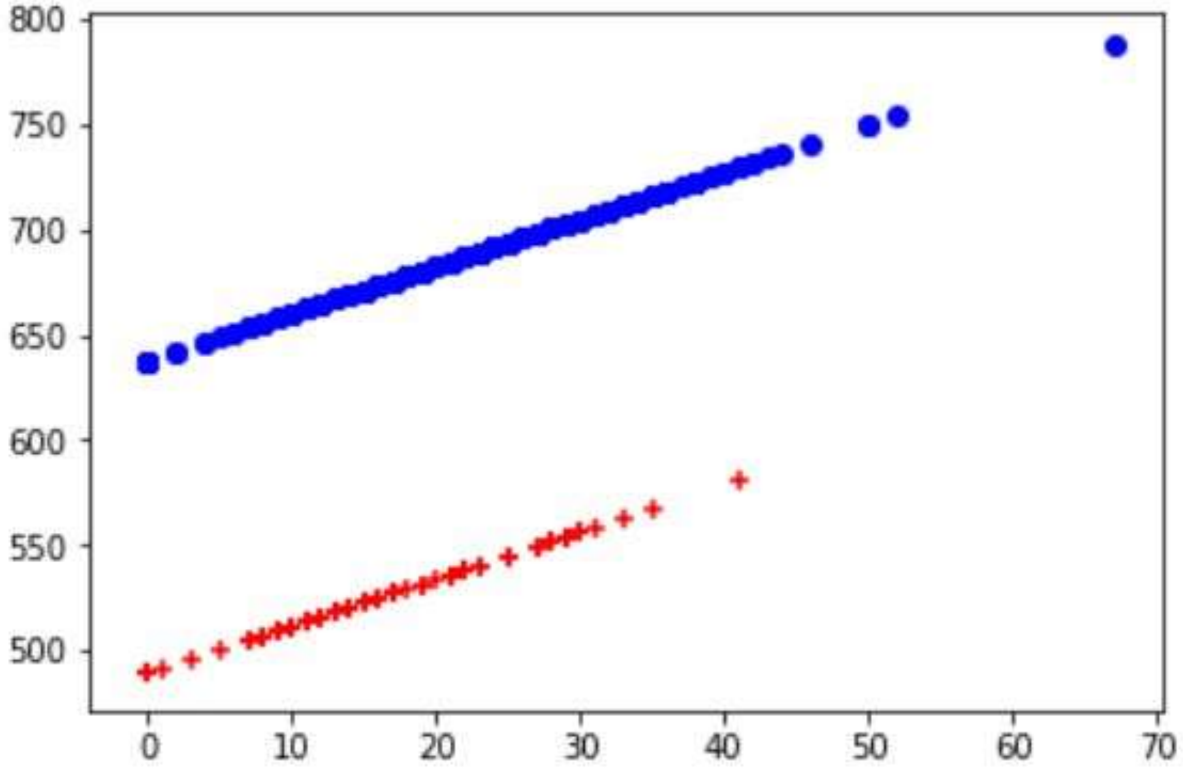
results show no indication of interaction

Continuous and categorical variables

```
elemapi2_ = elemapi2.copy()

lm = smf.ols('api00 ~ yr_rnd + some_col', data = elemapi2_).fit()
print (lm.summary())

elemapi2_['pred'] = lm.predict()

plt.scatter(elemapi2_.query('yr_rnd == 0').some_col, elemapi2_.query('yr_rnd == 0').pred, c = "b", marker = "o")
plt.scatter(elemapi2_.query('yr_rnd == 1').some_col, elemapi2_.query('yr_rnd == 1').pred, c = "r", marker = "+")
plt.plot()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.257
Model:                            OLS   Adj. R-squared:                  0.253
Method:                 Least Squares   F-statistic:                     68.54
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           2.69e-26
Time:                        10:49:35   Log-Likelihood:                -2490.8
No. Observations:                 400   AIC:                             4988.
Df Residuals:                     397   BIC:                             5000.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    637.8581     13.503     47.237      0.000     611.311     664.405
yr_rnd      -149.1591     14.875    -10.027      0.000    -178.403    -119.915
some_col       2.2357      0.553      4.044      0.000       1.149       3.323
==============================================================================
Omnibus:                       23.070   Durbin-Watson:                   1.565
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                9.935
Skew:                           0.125   Prob(JB):                      0.00696
Kurtosis:                       2.269   Cond. No.                         62.5
```

Linea Regression_PYTHON



Relationship between some_col and api00 but there were two regression lines, one higher than the other but with equal slope. Such a model assumed that the slope was the same for the two groups. Perhaps the slope might be different!!

	coef
Intercept	637.8581
yr_rnd	-149.1591
some_col	2.2357

- The coefficient for some_col indicates that for every unit increase in some_col the api00 score is predicted to increase by 2.23 units.
- The graph has two lines, one for the year round schools and one for the non-year round schools.
- The coefficient for yr_rnd is -149.16, indicating that as yr_rnd increases by 1 unit, the api00 score is expected to decrease by about 149 units.
- the top line is about 150 units higher than the lower line.
- the intercept is 637 and that is where the upper line crosses the Y axis when X is 0. The lower line crosses the line about 150 units lower at about 487.

Using categorical variable directly

```
lm = smf.ols('api00 ~ C(yr_rnd) + some_col', data = elemapi2_).fit()
print (lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.257
Model:                            OLS   Adj. R-squared:                  0.253
Method:                 Least Squares   F-statistic:                     68.54
Date:                Mon, 03 Sep 2018   Prob (F-statistic):           2.69e-26
Time:                        10:57:12   Log-Likelihood:                -2490.8
No. Observations:                 400   AIC:                             4988.
Df Residuals:                     397   BIC:                             5000.
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept        637.8581     13.503     47.237      0.000     611.311     664.405
C(yr_rnd)[T.1]  -149.1591     14.875    -10.027      0.000    -178.403    -119.915
some_col           2.2357      0.553      4.044      0.000       1.149       3.323
==============================================================================
Omnibus:                       23.070   Durbin-Watson:                   1.565
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                9.935
Skew:                           0.125   Prob(JB):                      0.00696
Kurtosis:                       2.269   Cond. No.                         62.5
```

```python
lm_0 = smf.ols(formula = "api00 ~ some_col", data = elemapi2.query('yr_rnd == 0')).fit()
lm_0.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.016
Model:	OLS	**Adj. R-squared:**	0.013
Method:	Least Squares	**F-statistic:**	4.915
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	0.0274
Time:	11:01:38	**Log-Likelihood:**	-1938.2
No. Observations:	308	**AIC:**	3880.
Df Residuals:	306	**BIC:**	3888.
Df Model:	1		
Covariance Type:	nonrobust		

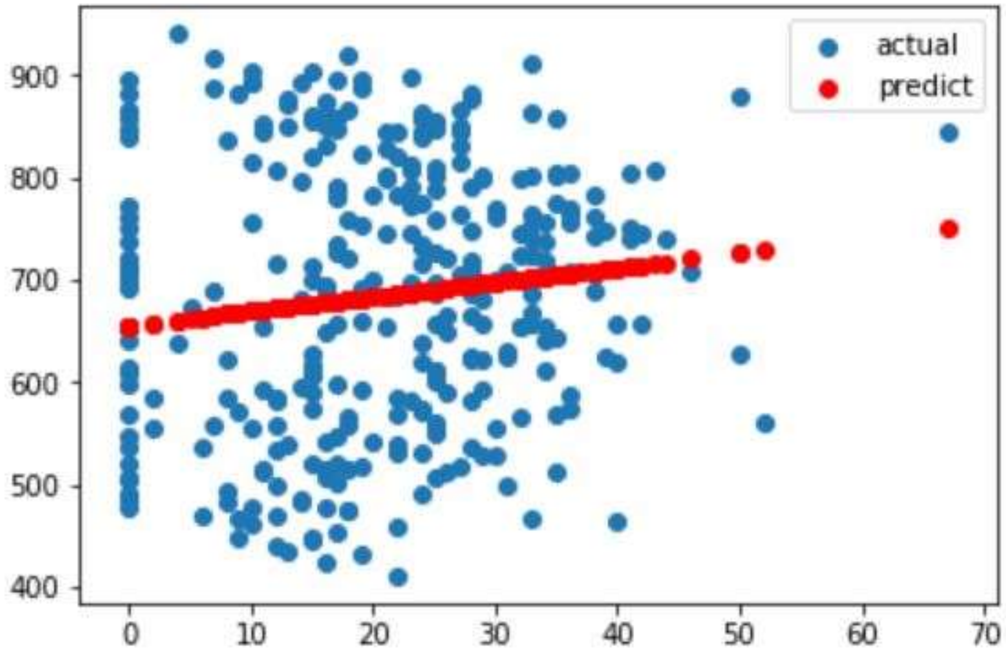
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	655.1103	15.237	42.995	0.000	625.128	685.093
some_col	1.4094	0.636	2.217	0.027	0.158	2.660

Omnibus:	63.461	**Durbin-Watson:**	1.531
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13.387
Skew:	-0.003	**Prob(JB):**	0.00124
Kurtosis:	1.979	**Cond. No.**	48.9

Some_col only

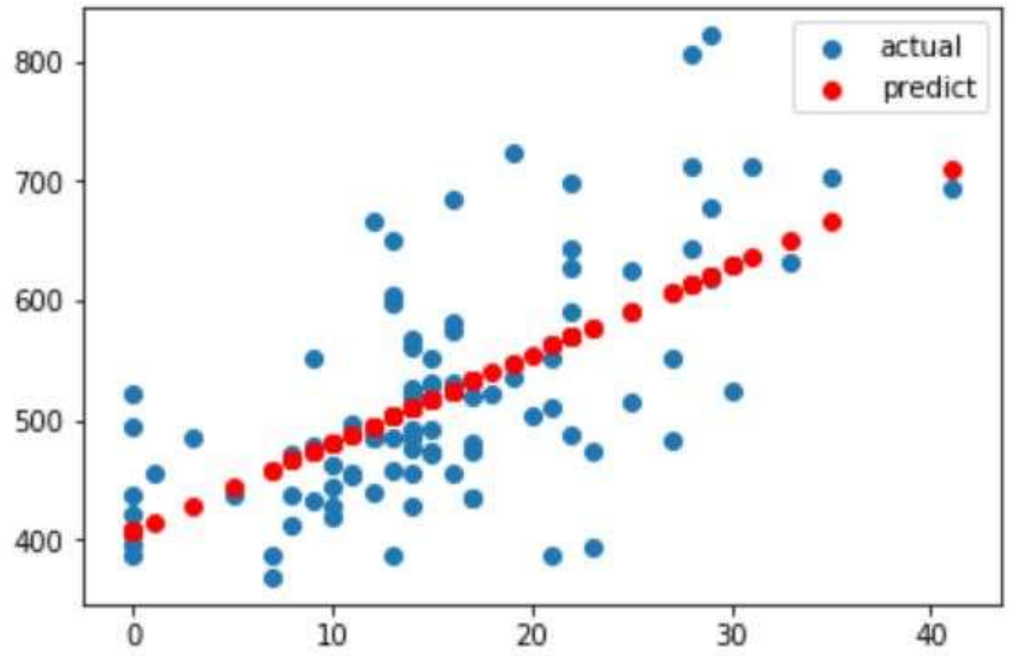
Interaction continuous categorical using yr-rnd 0

```
plt.scatter(elemapi2.query('yr_rnd == 0').some_col.values, elemapi2.query('yr_rnd == 0').api00.values, label = "actual")
plt.scatter(elemapi2.query('yr_rnd == 0').some_col.values, lm_0.predict(), c = "r", label = "predict")
plt.legend()
plt.show()
```



Interaction continuous categorical using yr-end 1

```
lm_1 = smf.ols(formula = "api00 ~ some_col", data = elemapi2.query('yr_rnd == 1')).fit()
plt.scatter(elemapi2.query('yr_rnd == 1').some_col.values, elemapi2.query('yr_rnd == 1').api00.values, label = "actual")
plt.scatter(elemapi2.query('yr_rnd == 1').some_col.values, lm_1.predict(), c = "r", label = "predict")
plt.legend()
plt.show()
```



The slope of the regression looks much steeper for the year round schools than for the non-year round schools.
(slope for the year round schools to be higher (6.55) than non-year round schools (1.4)). We can compare these to see if these are significantly different from each other by including the

```
yrxsome_elemapi = elemapi2
yrxsome_elemapi["yrxsome"] = yrxsome_elemapi.yr_rnd * yrxsome_elemapi.some_col
```

```
lm = smf.ols("api00 ~ some_col + yr_rnd + yrxsome", data = yrxsome_elemapi).fit()
lm.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.283
Model:	OLS	**Adj. R-squared:**	0.277
Method:	Least Squares	**F-statistic:**	52.05
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	2.21e-28
Time:	11:12:21	**Log-Likelihood:**	-2483.6
No. Observations:	400	**AIC:**	4975.
Df Residuals:	396	**BIC:**	4991.
Df Model:	3		
Covariance Type:	nonrobust		

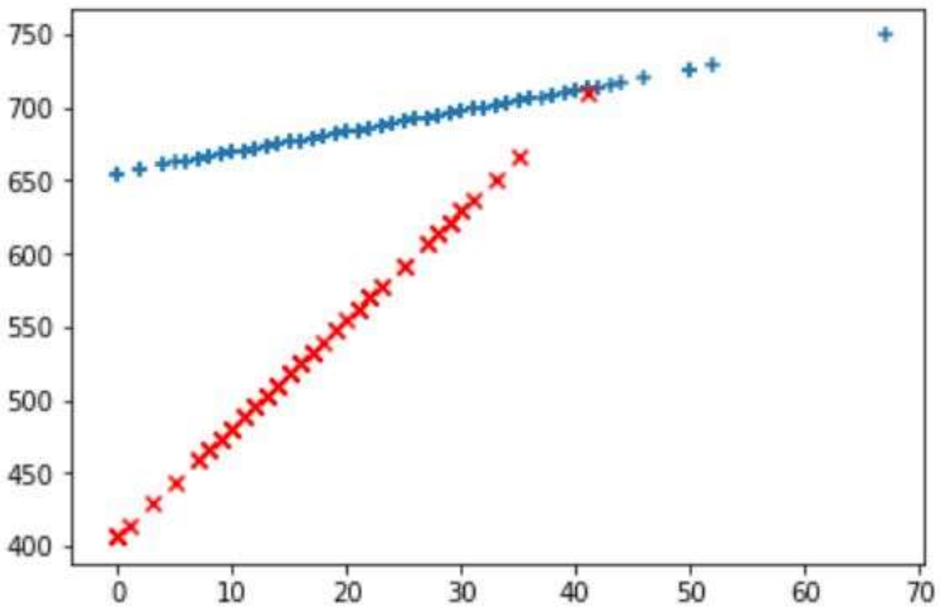
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	655.1103	14.035	46.677	0.000	627.518	682.703
some_col	1.4094	0.586	2.407	0.017	0.258	2.561
yr_rnd	-248.0712	29.859	-8.308	0.000	-306.773	-189.369
yrxsome	5.9932	1.577	3.800	0.000	2.893	9.094

Interaction of some_col by yr_rnd (interaction of a continuous variable by a categorical variable)

Computing interactions manually
(variable that is the interaction of some college (some_col) and year round schools (yr_rnd) called yrxsome

```
lt.scatter(yrxsome_elemapi.query('yr_rnd == 0').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 0], marker = "+")
lt.scatter(yrxsome_elemapi.query('yr_rnd == 1').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 1], c = "r", marker = "x
```

```
<matplotlib.collections.PathCollection at 0x1fec843e198>
```

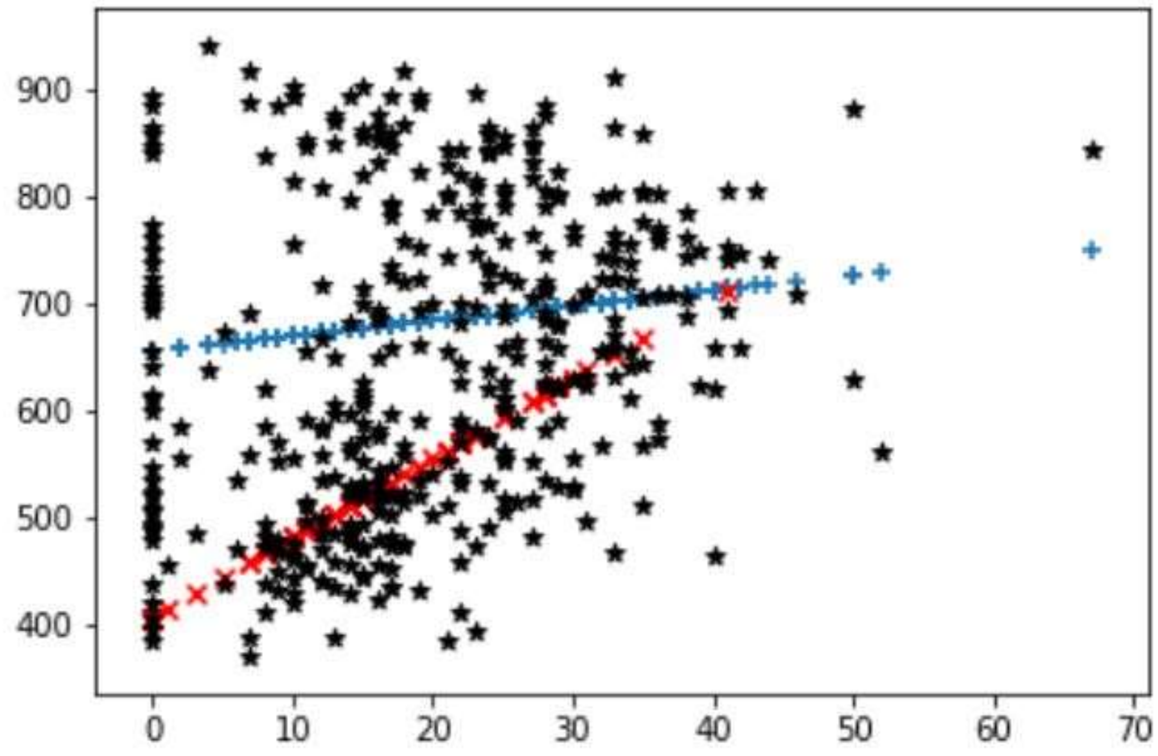


The two lines have quite different slopes, consistent with the fact that the yrxsome interaction was significant.

create a plot including the data points

```
i.query('yr_rnd == 0').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 0], marker = "+")
i.query('yr_rnd == 1').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 1], c = "r", marker = "x"]
i.some_col, yrxsome_elemapi.api00, c = "black", marker = "*")
```

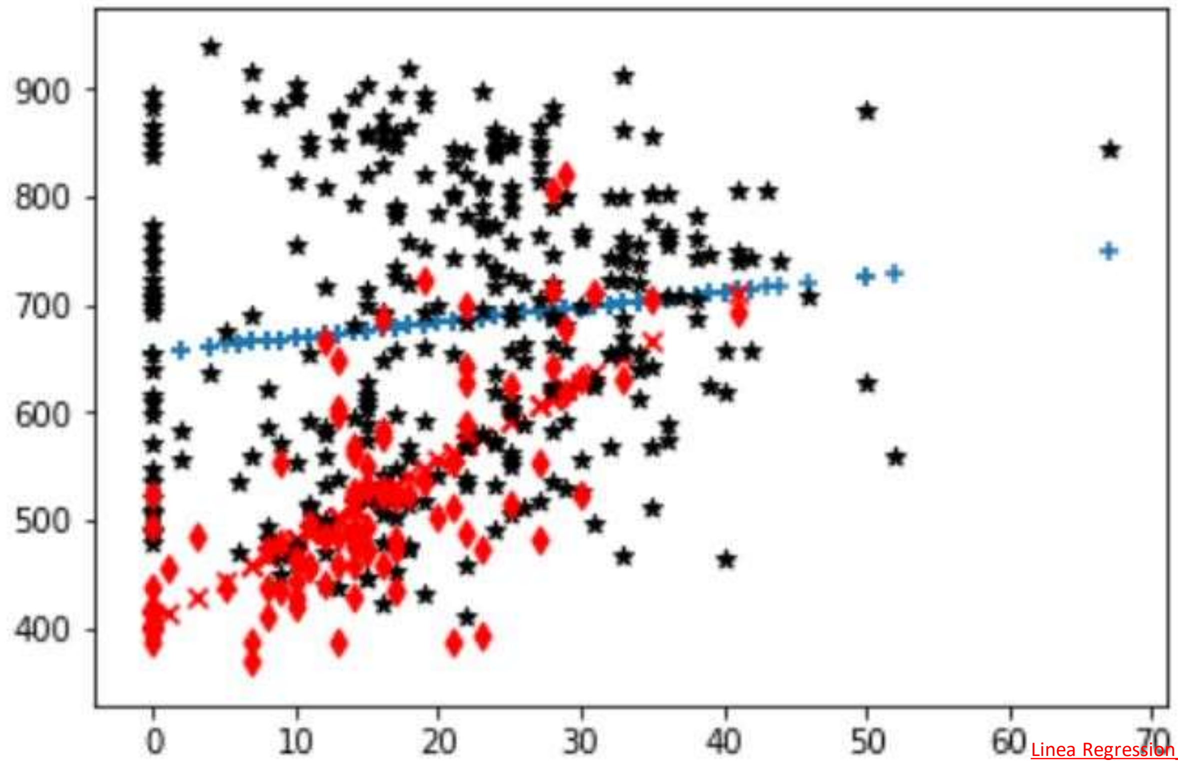
```
<matplotlib.collections.PathCollection at 0x1fec84a72e8>
```



Make separate variables for the api00 scores for the two types of schools called api0 for the non-year round schools and api1 for the year round schools.

```
..query('yr_rnd == 0').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 0], marker = "+")
..query('yr_rnd == 1').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 1], c = "r", marker = "x"
..query('yr_rnd == 0').some_col, yrxsome_elemapi.query('yr_rnd == 0').api00, c = "black", marker = "*")
..query('yr_rnd == 1').some_col, yrxsome_elemapi.query('yr_rnd == 1').api00, c = "r", marker = "d")
```

```
<matplotlib.collections.PathCollection at 0x1fec850f2e8>
```



Linea Regression_PYTHON

split data to yr_rnd = 0 group and yr_rnd = 1 group. Then run regression of api00 to some col in each group seperately.

```
yrxsome_elemapi_0 = yrxsome_elemapi.query('yr_rnd == 0')
yrxsome_elemapi_1 = yrxsome_elemapi.query('yr_rnd == 1')
lm_0 = smf.ols("api00 ~ some_col", data = yrxsome_elemapi_0).fit()
print (lm_0.summary())
print ('\n')
lm_1 = smf.ols("api00 ~ some_col", data = yrxsome_elemapi_1).fit()
print (lm_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  api00   R-squared:                       0.016
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     4.915
Date:                Mon, 03 Sep 2018   Prob (F-statistic):             0.0274
Time:                        11:23:31   Log-Likelihood:                -1938.2
No. Observations:                 308   AIC:                             3880.
Df Residuals:                     306   BIC:                             3888.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    655.1103     15.237     42.995      0.000     625.128     685.093
some_col       1.4094      0.636      2.217      0.027       0.158       2.660
==============================================================================
Omnibus:                       63.461   Durbin-Watson:                   1.531
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               13.387
Skew:                          -0.003   Prob(JB):                      0.00124
Kurtosis:                       1.979   Cond. No.                         48.9
```

```
lm = smf.ols("api00 ~ some_col + yr_rnd + yrxsome", data = yrxsome_elemapi).fit()
lm.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.283
Model:	OLS	**Adj. R-squared:**	0.277
Method:	Least Squares	**F-statistic:**	52.05
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	2.21e-28
Time:	11:25:56	**Log-Likelihood:**	-2483.6
No. Observations:	400	**AIC:**	4975.
Df Residuals:	396	**BIC:**	4991.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	655.1103	14.035	46.677	0.000	627.518	682.703
some_col	1.4094	0.586	2.407	0.017	0.258	2.561
yr_rnd	-248.0712	29.859	-8.308	0.000	-306.773	-189.369
yrxsome	5.9932	1.577	3.800	0.000	2.893	9.094

Omnibus:	23.863	**Durbin-Watson:**	1.593
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9.350

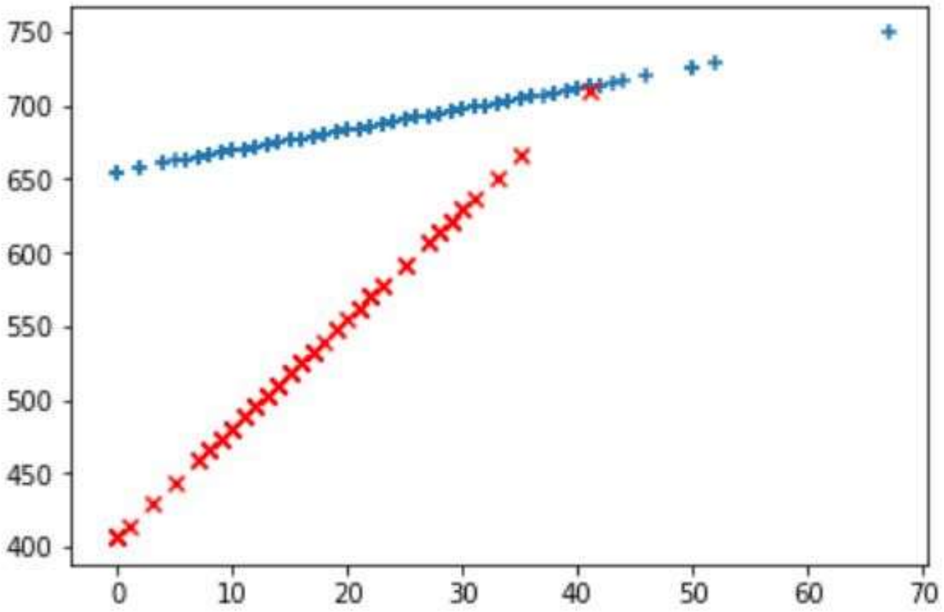
The coefficient for some_col in the combined analysis is the same as the coefficient for some_col for the non-year round schools?

This is because non-year round schools are the reference group. Then, the coefficient for the yrxsome interaction in the combined analysis is the Bsome_col for the year round schools (7.4) minus Bsome_col for the non year round schools (1.41) yielding 5.99

Show the regression for both types of schools with the interaction term

```
plt.scatter(yrxsome_elemapi.query('yr_rnd == 0').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 0], marker = "+")
plt.scatter(yrxsome_elemapi.query('yr_rnd == 1').some_col, lm.predict()[yrxsome_elemapi.yr_rnd.values == 1], c = "r", marker = "x
```

<matplotlib.collections.PathCollection at 0x1fec8587198>



```
lm = smf.ols("api00 ~ some_col + yr_rnd + yr_rnd * some_col ", data = yrxsome_elemapi).fit()
lm.summary()
```

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.283
Model:	OLS	**Adj. R-squared:**	0.277
Method:	Least Squares	**F-statistic:**	52.05
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	2.21e-28
Time:	11:32:19	**Log-Likelihood:**	-2483.6
No. Observations:	400	**AIC:**	4975.
Df Residuals:	396	**BIC:**	4991.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	655.1103	14.035	46.677	0.000	627.518	682.703
some_col	1.4094	0.586	2.407	0.017	0.258	2.561
yr_rnd	-248.0712	29.859	-8.308	0.000	-306.773	-189.369
yr_rnd:some_col	5.9932	1.577	3.800	0.000	2.893	9.094

Omnibus:	23.863	**Durbin-Watson:**	1.593

the relationship between some_col and api00 was significantly stronger than for those from non-year round schools. In general, this type of analysis allows you to test whether the strength of the relationship between two continuous

```
lm = smf.ols("api00 ~ some_col + C(mealcat) + some_col * C(mealcat)", data = elemapi2).fit()
lm.summary()
```

The prior examples showed how to do regressions with a continuous variable and a categorical variable that has two levels. How about using a categorical variable with three levels, mealcat

OLS Regression Results

Dep. Variable:	api00	**R-squared:**	0.769
Model:	OLS	**Adj. R-squared:**	0.767
Method:	Least Squares	**F-statistic:**	263.0
Date:	Mon, 03 Sep 2018	**Prob (F-statistic):**	4.13e-123
Time:	11:39:42	**Log-Likelihood:**	-2256.6
No. Observations:	400	**AIC:**	4525.
Df Residuals:	394	**BIC:**	4549.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	825.8937	11.992	68.871	0.000	802.318	849.470
C(mealcat)[T.2]	-239.0300	18.665	-12.806	0.000	-275.725	-202.334
C(mealcat)[T.3]	-344.9476	17.057	-20.223	0.000	-378.483	-311.413
some_col	-0.9473	0.487	-1.944	0.053	-1.906	0.011
some_col:C(mealcat)[T.2]	3.1409	0.729	4.307	0.000	1.707	4.575
some_col:C(mealcat)[T.3]	2.6073	0.896	2.910	0.004	0.846	4.369

The relationship between some_col and api00 varied, depending on the level of mealcat. In comparing group 1 with group 2, the coefficient for some_col was significantly different, but there was no difference in the coefficient for some_col in comparing groups 2 and 3.