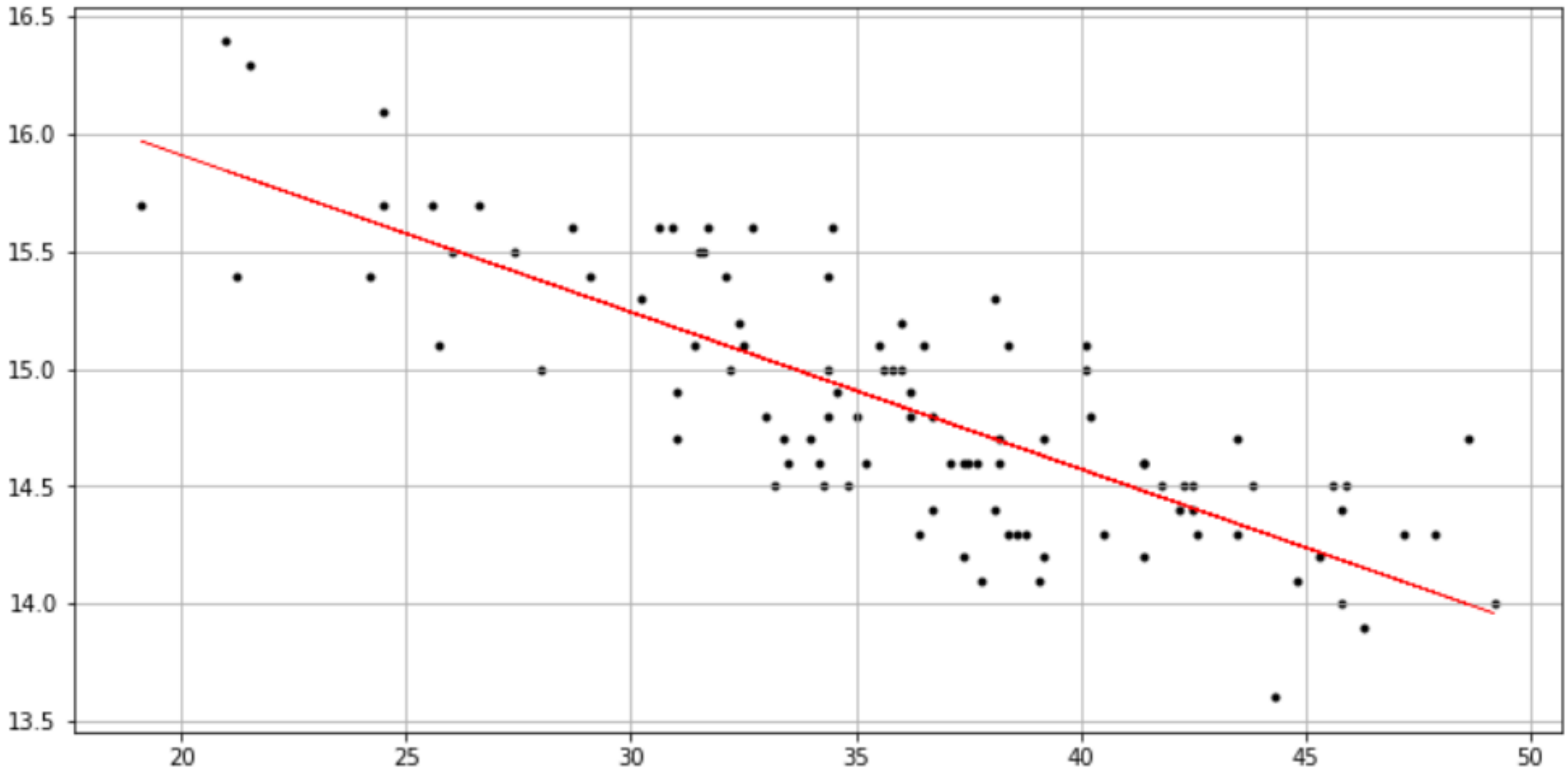
Multiple Linear Regression

OVERVIEW

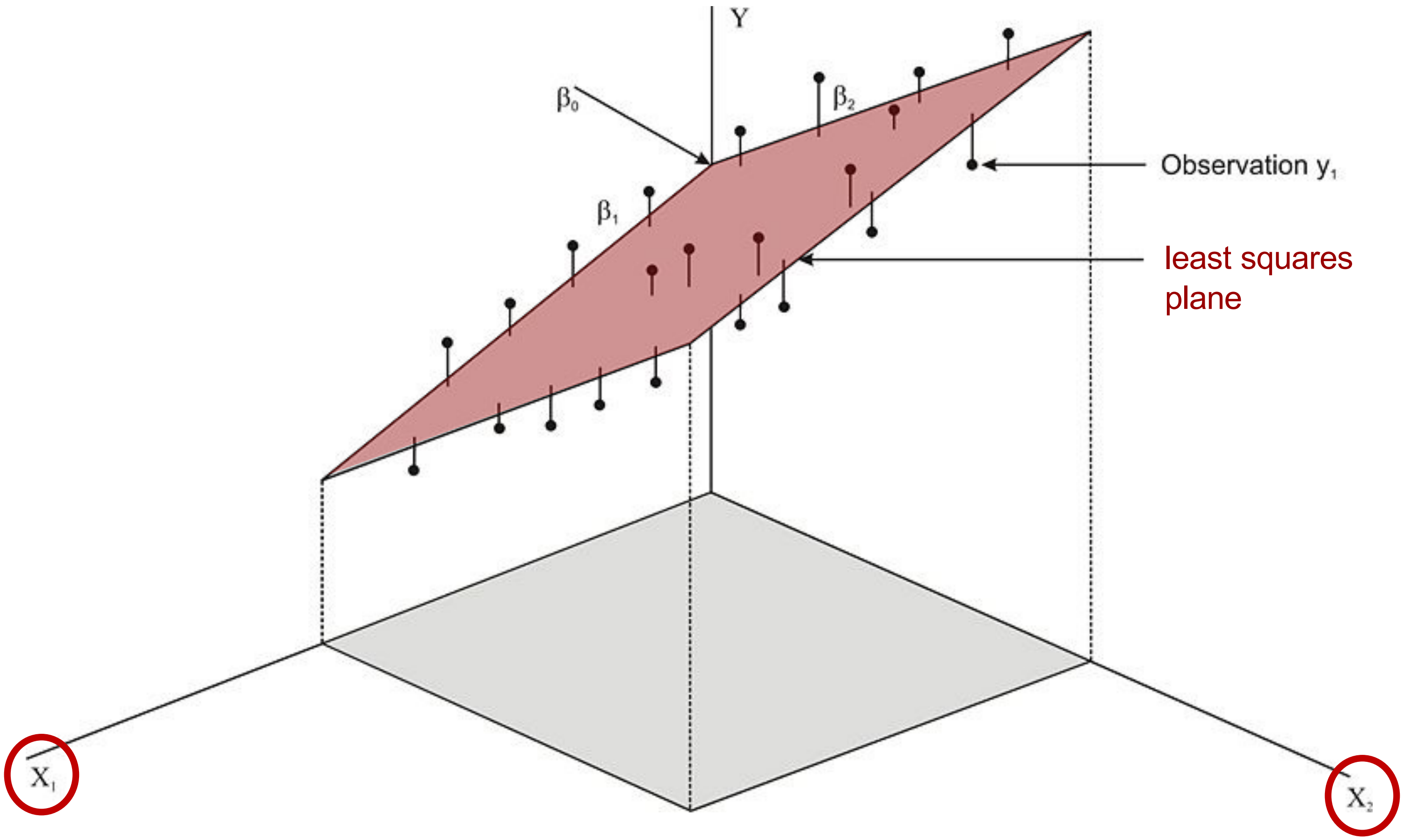
- *Introduction*
- *How good is the Regression Model?*
- *R-square*
- *Comparing Regression Models (Adj R^2, AIC)*
- *Libraries sklearn, statsmodels*

OLS – one predictor X

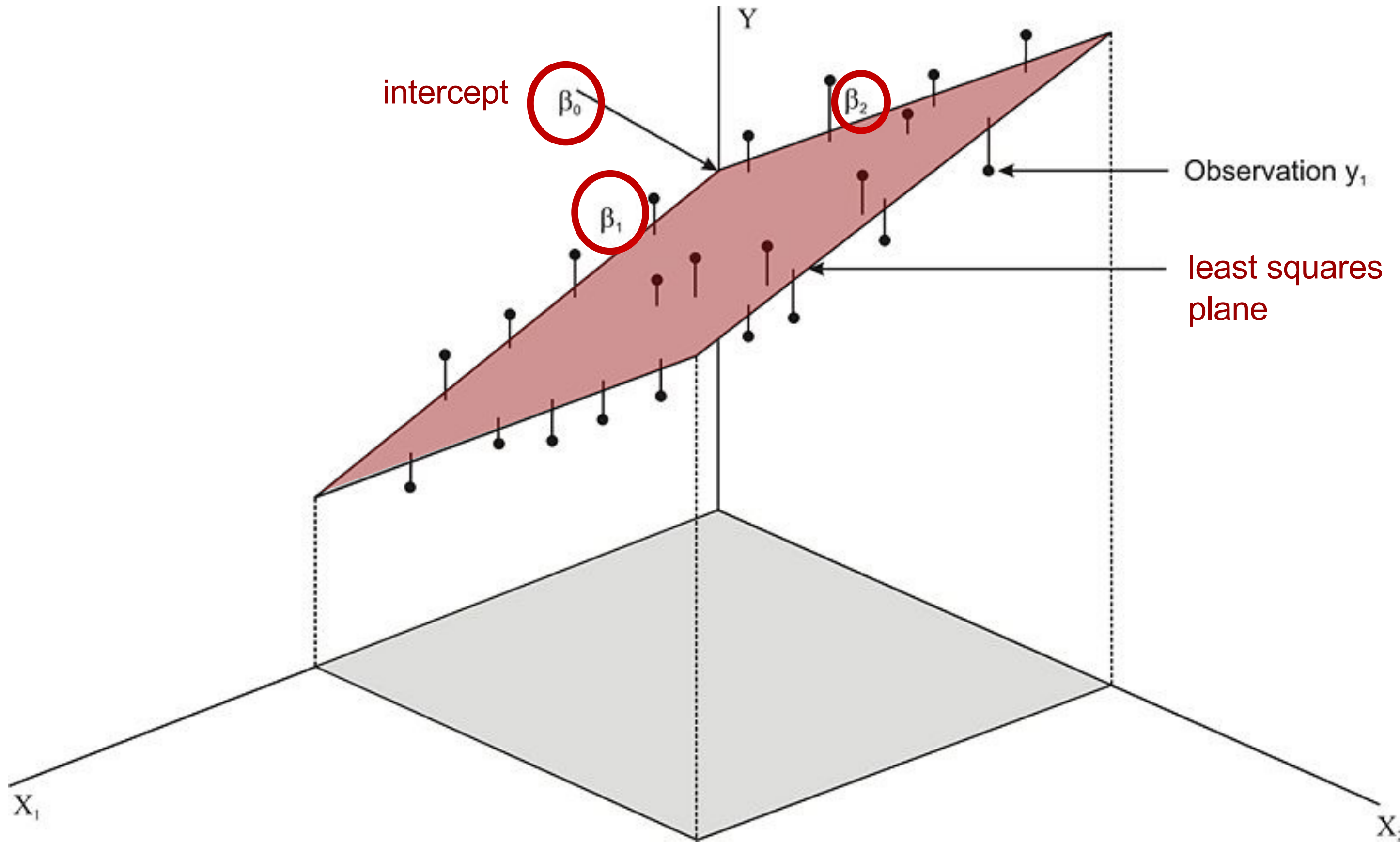


X

OLS - Two predictors X_1 and X_2



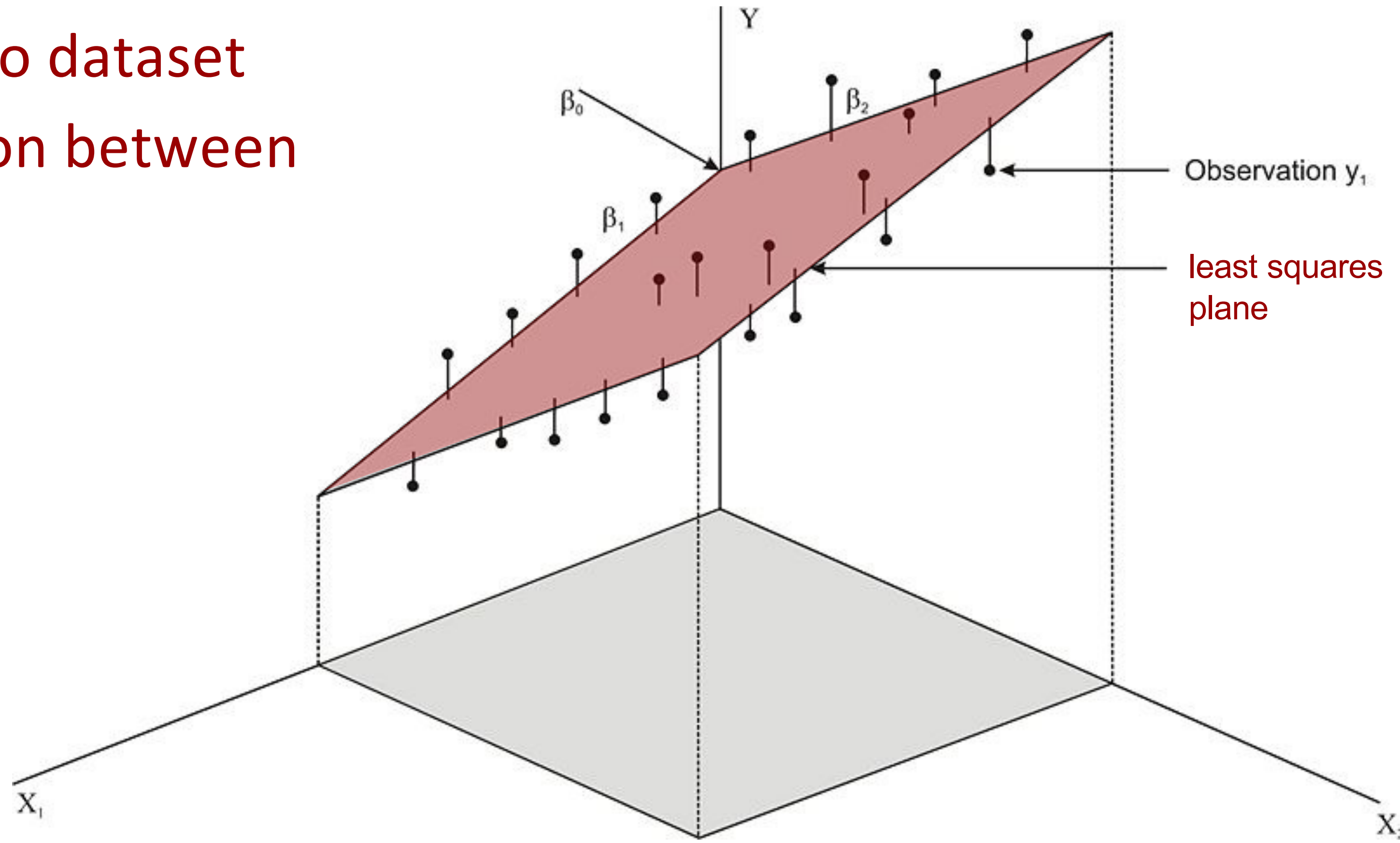
Two predictors X_1 and X_2



What is the OLS plane?

OLS plane

- Closest plane to dataset
- Average relation between (X_1,X_2) and Y



Multiple Linear Regression (MLR)

Consider *p* predictors $X_1, X_2,\ldots,X_p$

Regression plane (unknown) $$E[Y] = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

Multiple Linear Regression

Consider *p* predictors $X_1, X_2, \ldots, X_p$

Regression plane (unknown) $$E[Y] = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

OLS plane $$\hat{Y} = b_0 + b_1 X_1 + \cdots + b_p X_p$$

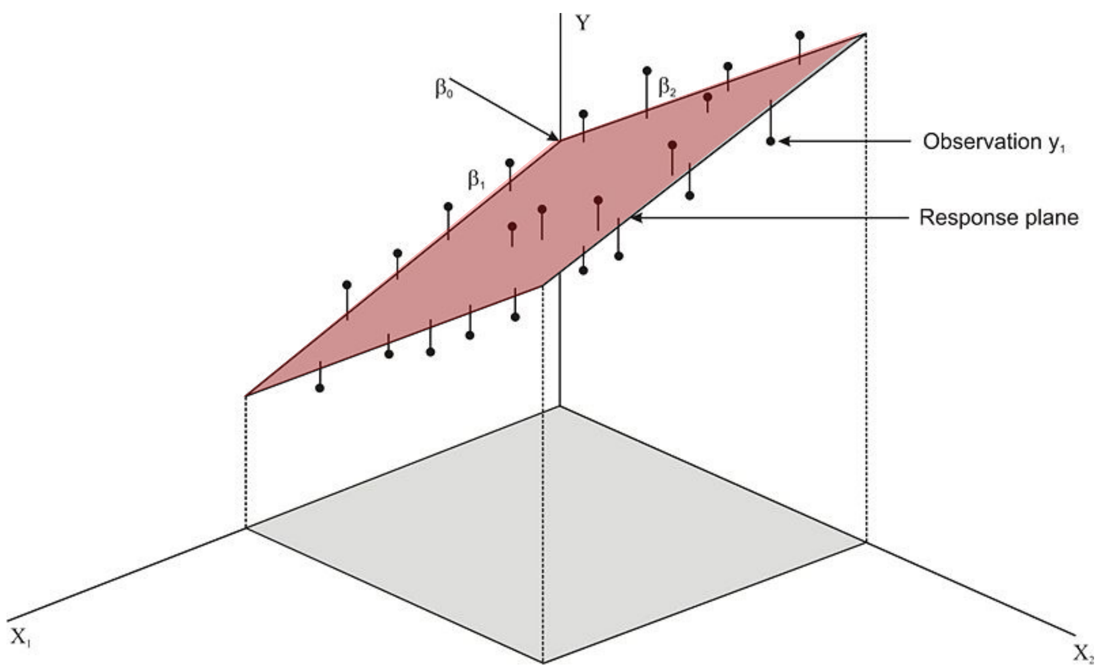
Model performance

How good is the regression model?

- *How well the model fits the data?*
- *How well the model predicts the data?*

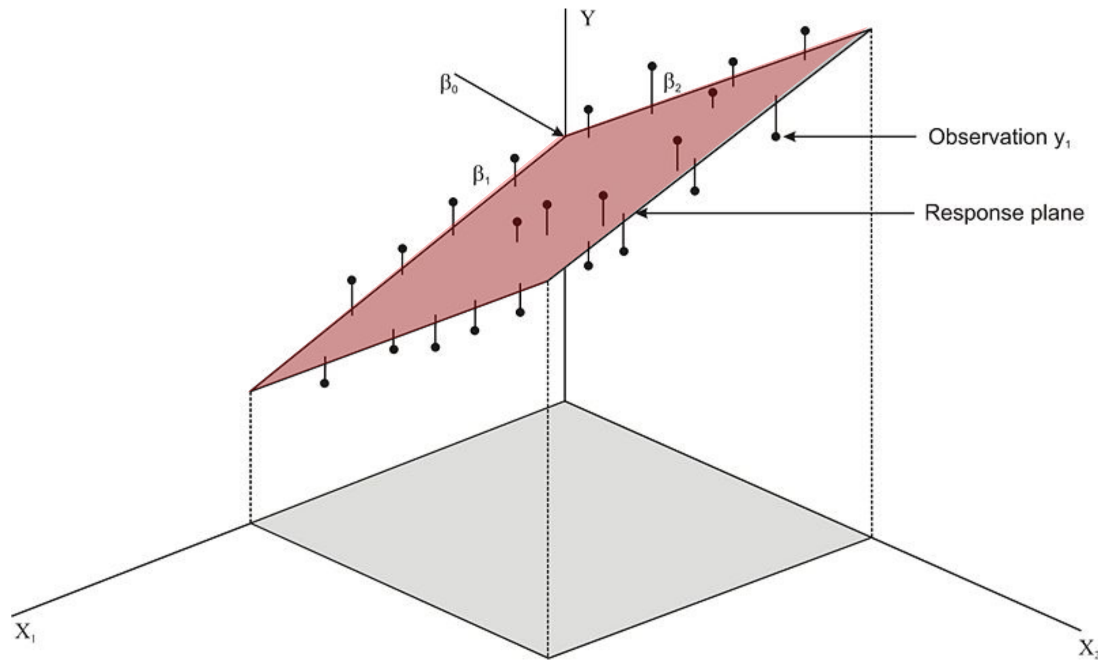
How well the model fits the data?

Model fits the data well if the regression plane is close to the data underlying pattern



How well the model predicts the data?

Model is a good predictor if it accurately predicts new data



How good is the regression model?

- *How well the model **fits** the data?*

  Model adequacy measures: SSE, R^2

- *How well the model **predicts** the data?*

  Cross Validation measures: MSPE

What is R-squared?

Model adequacy

R^2 is the proportion of the variation in Y
that is explained by
a linear model with predictors $X_1, X_2,\dots,X_p$

The larger the better

R^2 is the fraction of changes in Y that is explained by X

R^2 is always between 0 and 1

0 means that no changes in Y are explained by X

1 means all changes in Y are explained by X

(a perfect fit to the data)

R^2 is also called

Coefficient of multiple determination,

Coefficient determination,

Multiple R-squared

How is R-squared computed?

R-squared is the result of ANOVA decomposition

ANOVA DECOMPOSITION

- *SST* *Total Sum of Squares*
- *SSE* *Residuals Sum of Squares or Error Sum of Squares*
- *SSR* *Regression Sum of Squares*

$$SST = SSE + SSR$$

← ANOVA Decomposition

TOTAL SUM OF SQUARES

$$SST = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

This quantity is the same for all possible models

TOTAL SUM OF SQUARES

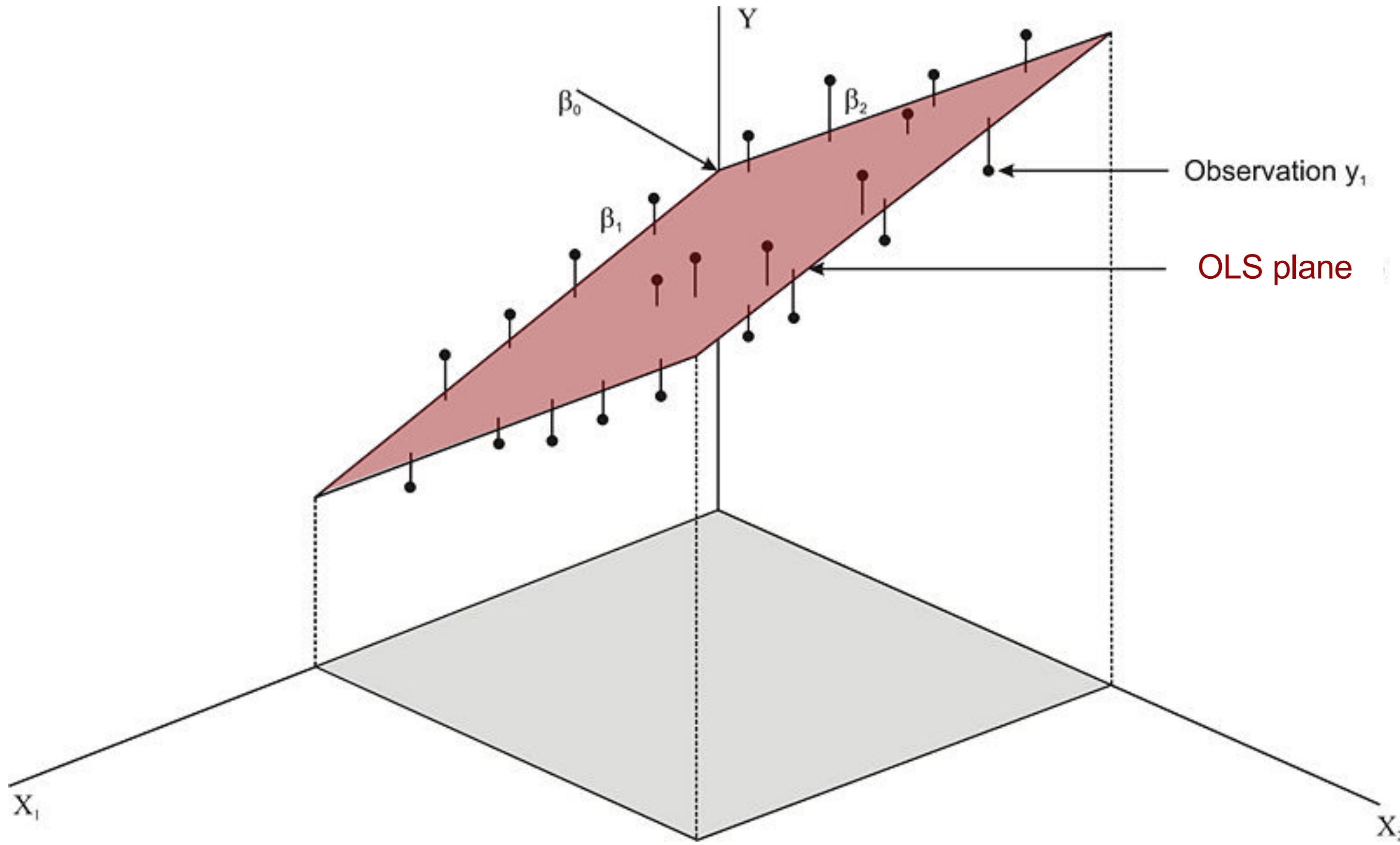
	X1	X2	Y	(Y-Ybar)^2
1				
2				
.				
.				
.				
N				
			Ybar	SST

SST = $$\sum_{i=1}^{n}(y_i - \overline{y})^2$$

- No model is needed to compute this quantity
- It is a constant for the dataset

OLS plane

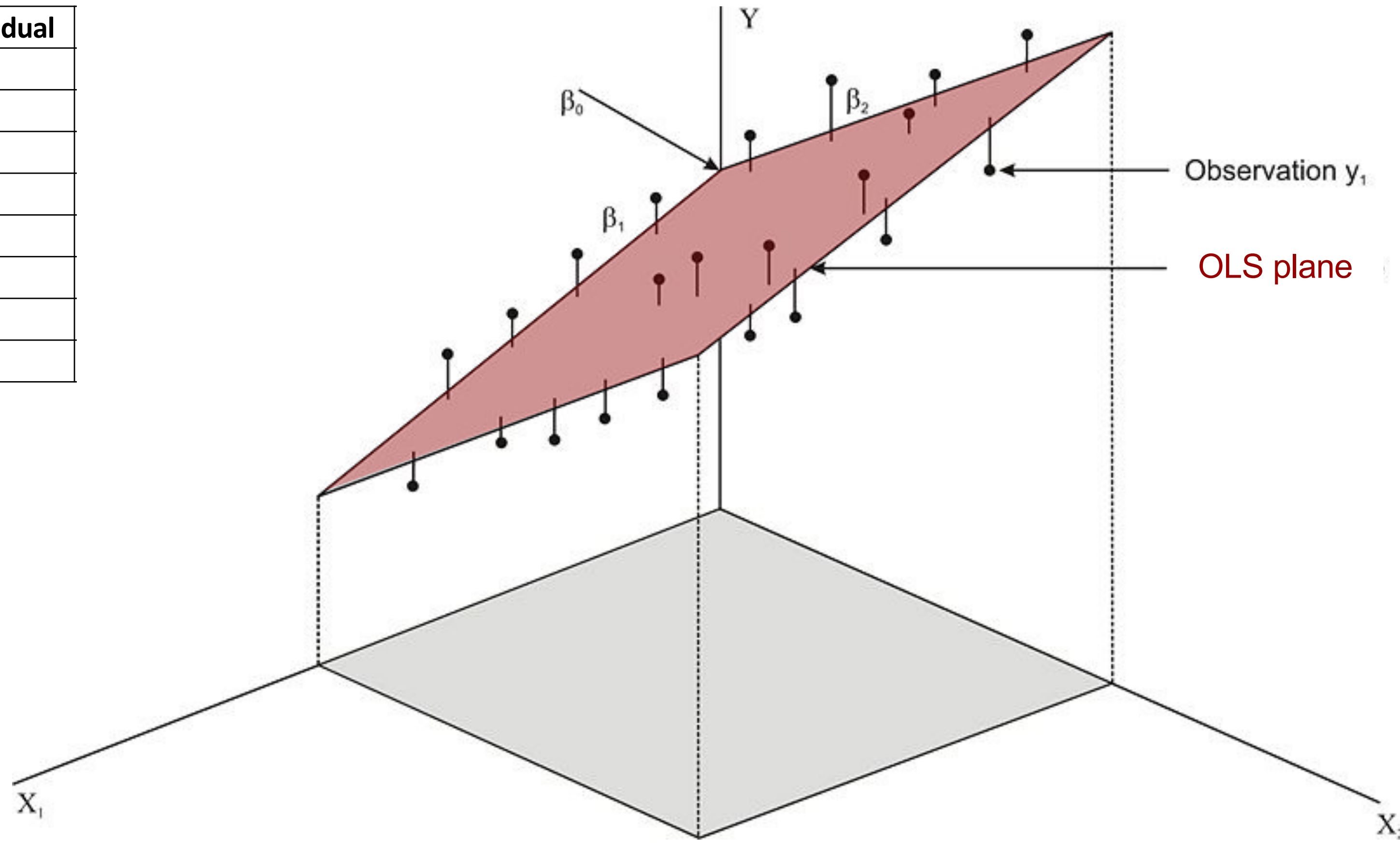
	X1	X2	Y
1			
2			
.			
.			
.			
n			



RESIDUALS

Y - Yhat

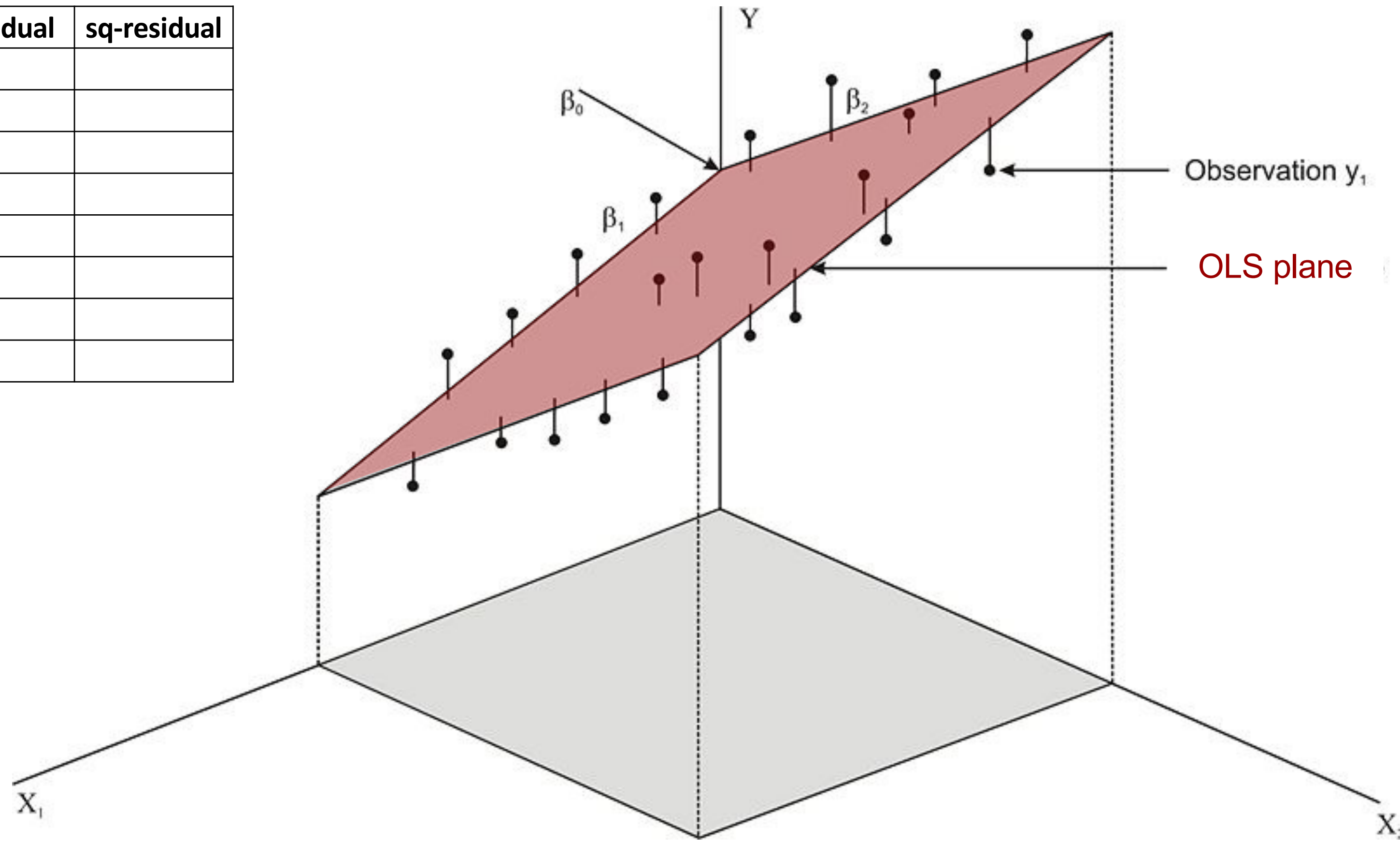
	X1	X2	Y	Yhat	residual
1					
2					
.					
.					
.					
n					



SQUARED RESIDUALS

Y - Yhat

	X1	X2	Y	Yhat	residual	sq-residual
1						
2						
.						
.						
.						
n						

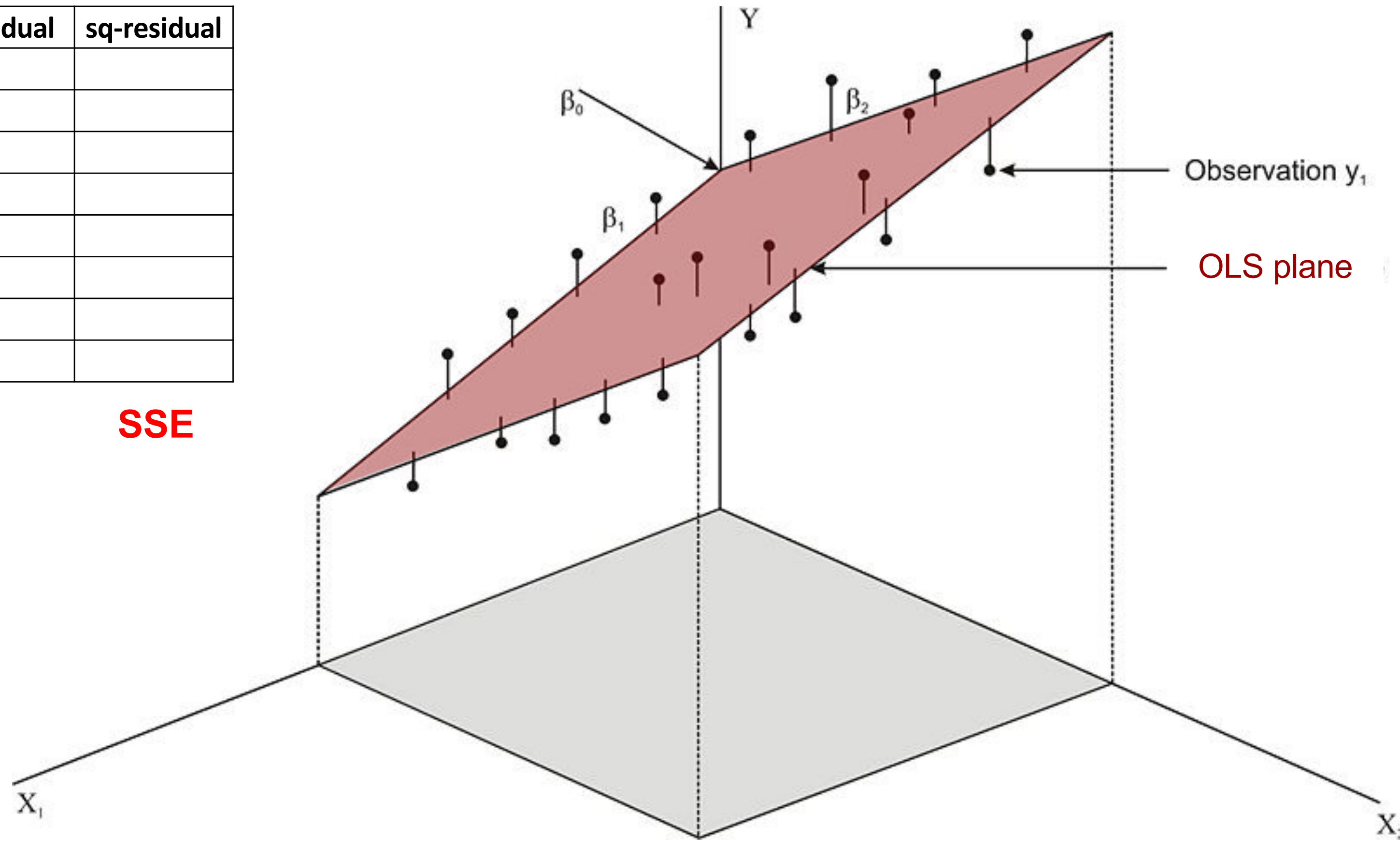


RESIDUAL SUM OF SQUARES (SSE)

Y - Yhat

	X1	X2	Y	Yhat	residual	sq-residual
1						
2						
.						
.						
.						
n						

SSE



ANOVA Decomposition

$$SST = SSE + SSR$$

How far is y_i from its mean?

$$y_i - \overline{y} \;=$$

$$y_i - \overline{y} \;=\; y_i - \hat{y}_i \;+\; \hat{y}_i - \overline{y}$$

How far is y_i from its mean?

$$y_i - \overline{y} \;=$$

$$y_i - \overline{y} \;=\; y_i - \hat{y}_i \;+\; \hat{y}_i - \overline{y}$$

$$y_i - \overline{y} \;=\; (y_i - \hat{y}_i) + (\hat{y}_i - \overline{y})$$

How far is y_i from its mean?

$$y_i - \overline{y} =$$

$$y_i - \overline{y} = y_i - \hat{y}_i + \hat{y}_i - \overline{y}$$

$$y_i - \overline{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \overline{y})$$

$$(y_i - \overline{y})^2 = [(y_i - \hat{y}_i) + (\hat{y}_i - \overline{y})]^2$$

TOTAL VARIABILITY OF Y

$$(y_i - \overline{y})^2 = [(y_i - \hat{y}_i) + (\hat{y}_i - \overline{y})]^2$$

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}[(y_i - \hat{y}_i) + (\hat{y}_i - \overline{y})]^2$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}[\underbrace{(y_i - \hat{y}_i)}_{a} + \underbrace{(\hat{y}_i - \overline{y})}_{b}]^2$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}[\underbrace{(y_i - \hat{y}_i)}_{a} + \underbrace{(\hat{y}_i - \overline{y})}_{b}]^2 = \sum_{i=1}^{n} a^2 + b^2 + 2ab$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}[\underbrace{(y_i - \hat{y}_i)}_{a} + \underbrace{(\hat{y}_i - \overline{y})}_{b}]^2$$

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}\underbrace{(y_i - \hat{y}_i)^2}_{a^2} + \sum_{i=1}^{n}\underbrace{(\hat{y}_i - \overline{y})^2}_{b^2} + 2\sum_{i=1}^{n}\underbrace{(y_i - \hat{y}_i)(\hat{y}_i - \overline{y})}_{ab}$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i-\overline{y})^2 = \sum_{i=1}^{n}[(y_i-\hat{y}_i)+(\hat{y}_i-\overline{y})]^2$$

$$\sum_{i=1}^{n}(y_i-\overline{y})^2 = \sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2+2\underbrace{\sum_{i=1}^{n}(y_i-\hat{y}_i)(\hat{y}_i-\overline{y})}_{0}$$

$$\sum_{i=1}^{n}(y_i-\overline{y})^2 = \sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n}(\underbrace{y_i - \hat{y}_i}_{e_i})^2 + \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2$$

ANOVA DECOMPOSITION

$$\sum_{i=1}^{n}(y_i - \overline{y})^2 = \sum_{i=1}^{n} e_i^2 + \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2$$

$$SST = SSE + SSR$$

Total sum of squares

Residual sum of squares

Regression sum of squares

ANOVA DECOMPOSITION, *R-squared*

$$SST \quad = \quad SSE \quad + \quad SSR$$

- SSE and SSR are in [0,1] always
- If one increases the other decreases
- SST is the same for all possible models
- SSE, SSR change with different models
- Best models give small SSE (and thus, large SSR)

R-squared definition

.

$$SST = SSE + SSR$$

small large

$$1 = \frac{SSE}{SST} + \frac{SSR}{SST}$$

R-squared definition

$$SSTotal = SSE + SSR$$

$$1 = \frac{SSE}{SST} + \frac{SSR}{SST}$$

$$1 = \frac{SSE}{SST} + R^2$$

R-squared definition

$$SSTotal = SSE + SSR$$

$$1 = \frac{SSE}{SST} + \frac{SSR}{SST}$$

$$1 = \frac{SSE}{SST} + R^2$$

R-squared formula

$$R^2 = 1 - \frac{SSE}{SST}$$

R-squared formula

$$R^2 = 1 - \frac{SSE}{SST}$$

This quantity changes with different models

This quantity is the same for all models

R-squared formula

$$R^2 = 1 - \frac{SSE}{SST}$$

- Small SSE gives large R^2
- Adding more predictors to the regression model decreases SSE, always

Goodness of Fit

The least squares method always produces a straight line or plane, even

- if there is *no* relationship between the variables, or
- if the relationship is *non*-linear

Goodness of Fit

The least squares method always produces a straight line or plane, even

- if there is *no* relationship between the variables, or
- if the relationship is *non*-linear

Hence, in addition to building the model,

we need to measure how well the model “fits” the data.

Comparing Regression Models

Comparing Regression Models

- R-squared
- Adjusted R-squared
- AIC

MULTIPLE LINEAR REGRESSION

- R^2 useful to compare models
  with the same number *p* of predictors
- R^2 not useful to compare models
  with different number *p* of predictors
  since R^2 increases with *p*, always

MULTIPLE LINEAR REGRESSION

Consider *6* predictors $X_1, X_2,\ldots,X_6$ and 2 models

$$Yhat = b_0 + b_1 X_1 + b_3 X_3 + b_6 X_6$$

$$Yhat = b_0 + b_2 X_2 + b_4 X_4 + b_5 X_5$$

These models have same number of predictors thus we can compare these models using R^2

MULTIPLE LINEAR REGRESSION

Consider *8* predictors $X_1, X_2,\ldots,X_8$ and 2 models

$$Yhat = b_0 + b_1 X_1 + b_3 X_3 + b_6 X_6$$

$$Yhat = b_0 + b_2 X_2 + b_4 X_4 + b_5 X_5 + b_7 X_7 + b_8 X_8$$

These models have different number of predictors

We cannot compare these models using R^2

MULTIPLE LINEAR REGRESSION

How to compare models with different number of predictors?

- Adjusted R^2 (larger is better)
- AIC (smaller is better)

R-squared vs adj *R-squared - interpretation*

- *100R^2 the percentage of variation in Y that is explained by the model*

- *Adjusted R^2 has no interpretation*

How is Adjusted R-squared computed?

MEAN SQUARES

- *MST Total Mean Square*
- *MSE Mean Square Error*
- *MSR Regression Mean Square*

Formulas for Mean Squares

$$MST = \frac{SST}{n-1}$$ = variance of Y

$$MSR = \frac{SSR}{p}$$

$$MSE = \frac{SSE}{n-p-1}$$

Formulas for Mean Squares

$$MST = \frac{SST}{n-1}$$ = variance of Y

$$MSR = \frac{SSR}{n-1}$$

$$MSE = \frac{SSE}{n-1}$$

$$SST = SSE + SSR$$

$$\frac{SST}{n-1} = \frac{SSE}{n-1} + \frac{SSR}{n-1}$$

$$MST \approx MSE + MSR$$ approximately

$$SST = SSE + SSR$$

$$\frac{SST}{n-1} = \frac{SSE}{n-1} + \frac{SSR}{n-1}$$

$$MST \approx MSE + MSR$$ approximately

$$1 = \frac{MSE}{MST} + \frac{MSR}{MST}$$

$$1 = \frac{MSE}{MST} + adj\ R^2$$

$$SST = SSE + SSR$$

$$\frac{SST}{n-1} = \frac{SSE}{n-1} + \frac{SSR}{n-1}$$

$$MST \approx MSE + MSR$$ approximately

$$1 = \frac{MSE}{MST} + \frac{MSR}{MST}$$

$$\boxed{1 = \frac{MSE}{MST}} + adj\ R^2$$

FORMULAS

$$R^2 = 1 - \frac{SSE}{SST}$$

$$adj\ R^2 \approx 1 - \frac{MSE}{MST}$$

ANOVA Table

ANOVA Table for **SLR**

SST = SSE + SSR

$$MST = \frac{SST}{n-1}$$

variance of Y

$$MSR = \frac{SSR}{p}$$

$$MSE = \frac{SSE}{n-p-1}$$

ANOVA Table for **SLR**

SST = SSE + SSR

$$MST = \frac{SST}{n-1} \qquad MSR = \frac{SSR}{1} \qquad MSE = \frac{SSE}{n-2}$$

variance of Y

	degrees of freedom	Sum of squares	Mean squares	F stat	p-value
Predictor	*1*	SSR	MSR	MSR/MSE	
Residual	*n-2*	SSE	MSE		
Total	*n-1*	SST	MST		

Useful measures from the ANOVA TABLE

```
import statsmodels.api as sm

table1 = sm.stats.anova_lm(m2)
table1
```

	df	sum_sq	mean_sq	F	PR(>F)
Odometer	1.0	19.255607	19.255607	180.642989	5.750781e-24
Residual	98.0	10.446293	0.106595		

SSE MSE

Useful measures from the ANOVA TABLE

```
import statsmodels.api as sm
table1 = sm.stats.anova_lm(m2)
table1
```

	df	sum_sq	mean_sq	F	PR(>F)
Odometer	1.0	19.255607	19.255607	180.642989	5.750781e-24
Residual	98.0	10.446293	0.106595		

$S = \sqrt{MSE} = 0.3265$ average distance to regression line

Example – ANOVA TABLE

```
import statsmodels.api as sm
table1 = sm.stats.anova_lm(m2)
table1
```

		sum_sq	mean_sq	F	PR(>F)
Odometer	SSR =	19.255607	19.255607	180.642989	5.750781e-24
Residual	SSE =	10.446293	0.106595		
Total	SST =	29.7019			

$R^2 = 1 – (SSE/SST) = 1 – (10.4463/29.7019) = 0.6483$

Example – ANOVA TABLE

```
import statsmodels.api as sm
table1 = sm.stats.anova_lm(m2)
table1
```

	df	sum_sq	mean_sq	F	PR(>F)
Odometer	1.0	19.255607	19.255607	180.642989	5.750781e-24
Residual	98.0	10.446293	0.106595		
Total	99	29.7019	MST = 29.7019 / 99 = 0.30		

Adj R^2 = 1 – (MSE/MST) = 1 – (0.106595/0.30) = 0.6447

ANOVA Table for **MLR**

SST = SSE + SSR

$$MST = \frac{SST}{n-1}$$

$$MSR = \frac{SSR}{p}$$

$$MSE = \frac{SSE}{n-p-1}$$

variance of Y

	degrees of freedom	Sum of squares	Mean squares	F stat	p-value
Predictor	*p*	SSR	MSR	MSR/MSE	
Residual	*n-p-1*	SSE	MSE		
Total	*n-1*	SST	MST		

What is AIC?

Akaike Information Criteria (AIC)

- Measures the loss of information by fitting a model from a sample (and not from the population)

- For MLR the AIC is

$$AIC = n \log\left(\frac{SSE}{n}\right) + 2p$$

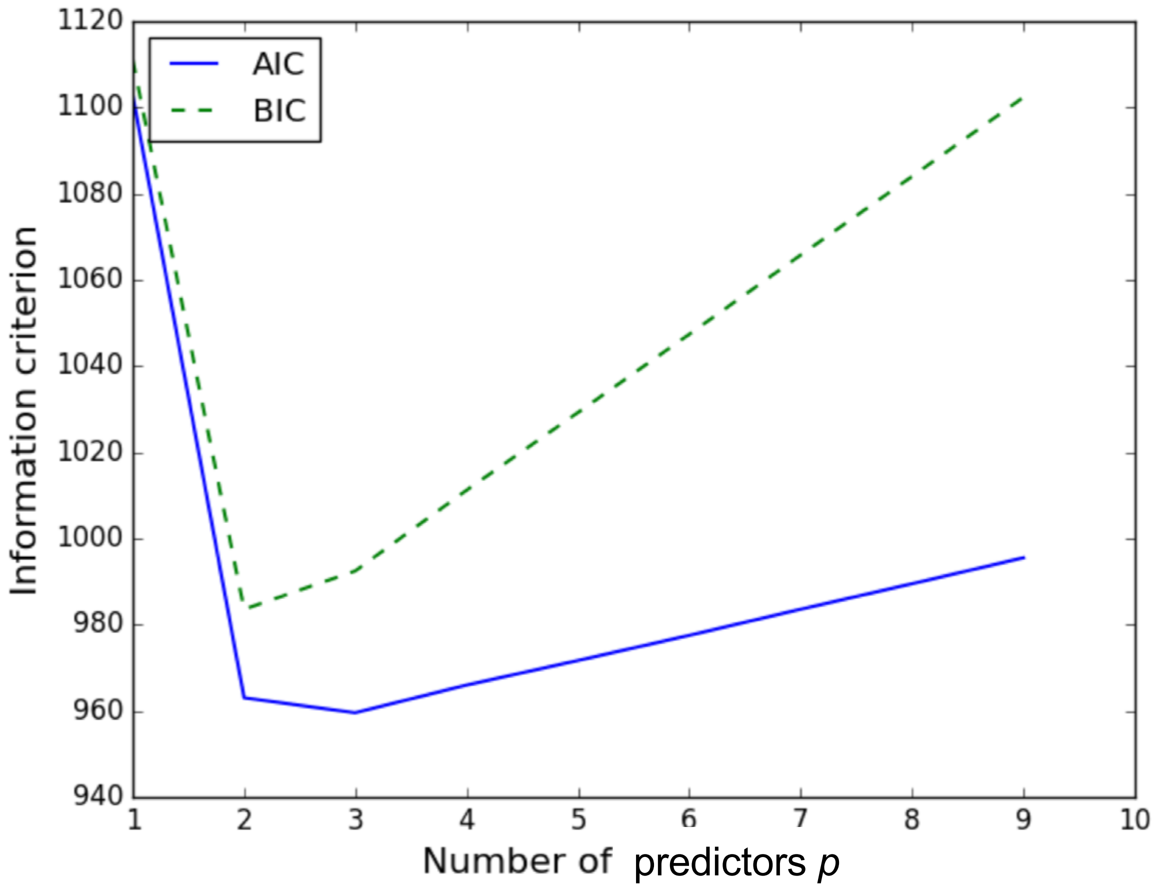
Akaike Information Criteria (AIC)

- Since it measures a loss, we prefer models with small AIC
- It makes a balance between SSE and *p*

$$AIC = n \log\left(\frac{SSE}{n}\right) + 2p$$

Akaike Information Criteria (AIC)

Choose the model with the smallest *AIC*



MULTIPLE LINEAR REGRESSION

- R^2 depends on SSE only,
  therefore it is useful to compare models with the same number *p* of predictors

- Adj-R^2 and AIC depend on SSE and *p*,
  therefore they are useful to compare models with different number *p* of predictors

EXAMPLE
Cars93 dataset

EXAMPLE 1

- Fit a MLR Model to predict the city mileage, MPG.city, of a car using as predictors the car's weight, horsepower, RPM, engine size, number of cylinders, and number of passengers
- Predict the city mileage of a 6-passenger car with 2800 pounds, 6 cylinders, 150 HP, 6600 RPM, and 1.9 engine size

EXAMPLE 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('Cars93.csv')
```

```
list1 = ['MPG.city','Cylinders','EngineSize','Horsepower
         'RPM','Passengers','Weight']
df1 = df[list1]
df1[:5]
```

Response

	MPG.city	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	25	4	1.8	140	6300	5	2705
1	18	6	3.2	200	5500	5	3560
2	20	6	2.8	172	5500	5	3375
3	19	6	2.8	172	5500	6	3405
4	22	4	3.5	208	5700	4	3640

EXAMPLE 1

```
df1.dtypes
```

```
MPG.city        int64
Cylinders      object
EngineSize    float64
Horsepower      int64
RPM             int64
Passengers      int64
Weight          int64
```

```
list1 = ['MPG.city','Cylinders','EngineSize','Horsepower
         'RPM','Passengers','Weight']
df1 = df[list1]
df1[:5]
```

	MPG.city	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	25	4	1.8	140	6300	5	2705
1	18	6	3.2	200	5500	5	3560
2	20	6	2.8	172	5500	5	3375
3	19	6	2.8	172	5500	6	3405
4	22	4	3.5	208	5700	4	3640

Why is Cylinders not numeric?

EXAMPLE 1

```
# Only one car with rotary cylinder
```

```
df1[df1.Cylinders == 'rotary']
```

	MPG.city	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
56	17	rotary	1.3	255	6500	2	2895

```
# remove it
```

```
df2 = df1[df1.Cylinders != 'rotary'].copy()
```

change the data type to int64

```
df2.Cylinders = df2.Cylinders.astype('int64')
df2.dtypes
```

```
MPG.city          int64
Cylinders         int64
EngineSize      float64
Horsepower        int64
RPM               int64
Passengers        int64
Weight            int64
```

EXAMPLE 1

```
y0 = df2['MPG.city']
X0 = df2.drop(columns = 'MPG.city',axis = 1)
```

Response in a Series

Predictors in a DataFrame

library sklearn

EXAMPLE 1

.

```
y0 = df2['MPG.city']
X0 = df2.drop(columns = 'MPG.city',axis = 1)
```

```
X0[:5]
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705
1	6	3.2	200	5500	5	3560
2	6	2.8	172	5500	5	3375
3	6	2.8	172	5500	6	3405
4	4	3.5	208	5700	4	3640

← X0.columns

EXAMPLE 1

.

```
y0 = df2['MPG.city']
X0 = df2.drop(columns = 'MPG.city',axis = 1)
```

```
from sklearn.linear_model import LinearRegression
```

```
model1 = LinearRegression().fit(X0,y0)
```

```
model1.intercept_
```

```
36.91997468801334
```

```
model1.coef_
```

```
array([ 0.1014525 ,  0.87431541, -0.03032167,  0.00161991, -0.2384795
       -0.00661264])
```

not clear what coefficient belongs to each predictor

EXAMPLE 1

```
model1.coef_
```

```
array([ 0.1014525 ,  0.87431541, -0.03032167,  0.00161991, -0.2384795 ,
       -0.00661264])
```

```
pd.DataFrame(model1.coef_,columns = ['coef'],
             index = X0.columns)
```

← create a one-column dataframe with the regression coefficients

	coef
Cylinders	0.101453
EngineSize	0.874315
Horsepower	-0.030322
RPM	0.001620
Passengers	-0.238479
Weight	-0.006613

with this DataFrame it is clear what coefficient belongs to each predictor

EXAMPLE 1

```
model1.coef_
```

```
array([ 0.1014525 ,  0.87431541, -0.03032167,  0.00161991, -0.2384795 
       -0.00661264])
```

```
pd.DataFrame(model1.coef_,columns = ['coef'],
             index = X0.columns)
```

the row names identify the predictor

	coef
Cylinders	0.101453
EngineSize	0.874315
Horsepower	-0.030322
RPM	0.001620
Passengers	-0.238479
Weight	-0.006613

```
X0[:5]
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705
1	6	3.2	200	5500	5	3560
2	6	2.8	172	5500	5	3375
3	6	2.8	172	5500	6	3405
4	4	3.5	208	5700	4	3640

EXAMPLE 1 - PREDICTION

Predict the city mileage of a car with

- 6 cylinders
- 1.9 Engine Size
- 150 HP
- 6600 RPM
- 6 Passengers
- Weight 2800 pounds

EXAMPLE 1 - PREDICTION

Predict the city mileage of a car with

- 6 cylinders
- 1.9 Engine Size
- 150 HP
- 6600 RPM
- 6 Passengers
- Weight 2800 pounds

```
X0[:5]
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705
1	6	3.2	200	5500	5	3560
2	6	2.8	172	5500	5	3375
3	6	2.8	172	5500	6	3405
4	4	3.5	208	5700	4	3640

EXAMPLE 1 - PREDICTION

```
newvalue = X0[:1].copy()
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705

- creating a one-row dataframe with the 1st row of X0

EXAMPLE 1 - PREDICTION

```
newvalue = X0[:1].copy()
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705

```
newvalue.Cylinders = 6
newvalue.EngineSize = 1.9
newvalue.Horsepower = 150
newvalue.RPM = 6600
newvalue.Passengers = 6
newvalue.Weight = 2800
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	6	1.9	150	6600	6	2800

- creating a one-row dataframe with the 1st row of X0
- .copy( ) is needed since this new dataframe is to be modified

EXAMPLE 1 - PREDICTION

```
newvalue = X0[:1].copy()
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705

```
newvalue.Cylinders = 6
newvalue.EngineSize = 1.9
newvalue.Horsepower = 150
newvalue.RPM = 6600
newvalue.Passengers = 6
newvalue.Weight = 2800
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	6	1.9	150	6600	6	2800

- creating a one-row dataframe with the 1st row of X0
- .copy( ) is needed since this new dataframe is to be modified
- Column of ones not needed with sklearn

predict city mileage

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

```
array([25.38678406])
```

library statsmodels.api

EXAMPLE 1

```
X0[:5]
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705
1	6	3.2	200	5500	5	3560
2	6	2.8	172	5500	5	3375
3	6	2.8	172	5500	6	3405
4	4	3.5	208	5700	4	3640

↑ need to insert a **column of ones**

EXAMPLE 1 – statsmodels.api

```
X0[:5]
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	4	1.8	140	6300	5	2705
1	6	3.2	200	5500	5	3560
2	6	2.8	172	5500	5	3375
3	6	2.8	172	5500	6	3405
4	4	3.5	208	5700	4	3640

```
import statsmodels.api as sm
```

```
X1 = X0.copy()
X1.insert(0,'const',1)
X1[:5]
```

← .copy( ) since X1 to be modified

	const	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	1	4	1.8	140	6300	5	2705
1	1	6	3.2	200	5500	5	3560
2	1	6	2.8	172	5500	5	3375
3	1	6	2.8	172	5500	6	3405
4	1	4	3.5	208	5700	4	3640

EXAMPLE 1 – statsmodels.api – build model

```
m1 = sm.OLS(y0,X1).fit()
m1.summary()
```

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

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

compare to
sklearn model1

	coef
Cylinders	0.101453
EngineSize	0.874315
Horsepower	-0.030322
RPM	0.001620
Passengers	-0.238479
Weight	-0.006613

EXAMPLE 1 – statsmodels.api

```
m1.params
```

```
const          36.919975
Cylinders       0.101453
EngineSize      0.874315
Horsepower     -0.030322
RPM             0.001620
Passengers     -0.238479
Weight         -0.006613
dtype: float64
```

```
# yhat = 36.92 + 0.1014 Cylinders + 0.874 EngineSize
#              -0.03 Horsepower + 0.0016 RPM
#              -0.2385 Passengers -0.0066 Weight
```

regression equation

```
# interpret equation
```

Average

```
# City mileage increases by 0.1014 for each additional cylinder
```

if all other variables do not change

Average

```
# City mileage decreases by 0.0066 for each additional pound
```

if all other variables do not change

EXAMPLE 1 – statsmodels.api Prediction

```
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	6	1.9	150	6600	6	2800

```
newvalue.insert(0, 'constant',1)
newvalue
```

	constant	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	1	6	1.9	150	6600	6	2800

```
m1.predict(newvalue)
```

```
0     25.386784
```

EXAMPLE 1 – statsmodels.api Prediction

```
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	6	1.9	150	6600	6	2800

```
newvalue.insert(0, 'constant',1)
newvalue
```

	constant	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	1	6	1.9	150	6600	6	2800

```
m1.predict(newvalue)
```

```
0     25.386784
```

predicted with sklearn

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

```
array([25.38678406])
```

EXAMPLE 1 – statsmodels.api Prediction, and 90% CI and PI

```
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	6	1.9	150	6600	6	2800

```
# get CI and PI
```

```
d2 = m1.get_prediction(newvalue)
d2.summary_frame(alpha = 0.10)
```

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	25.386784	1.439436	22.993032	27.780536	19.832545	30.941024

EXAMPLE 1 – statsmodels.api Prediction, 90% CI, and PI

```
newvalue
```

	Cylinders	EngineSize	Horsepower	RPM	Passengers	Weight
0	6	1.9	150	6600	6	2800

```
# get CI and PI
```

```
d2 = m1.get_prediction(newvalue)
d2.summary_frame(alpha = 0.10)
```

			Confidence Interval		Prediction Interval	
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	25.386784	1.439436	22.993032	27.780536	19.832545	30.941024

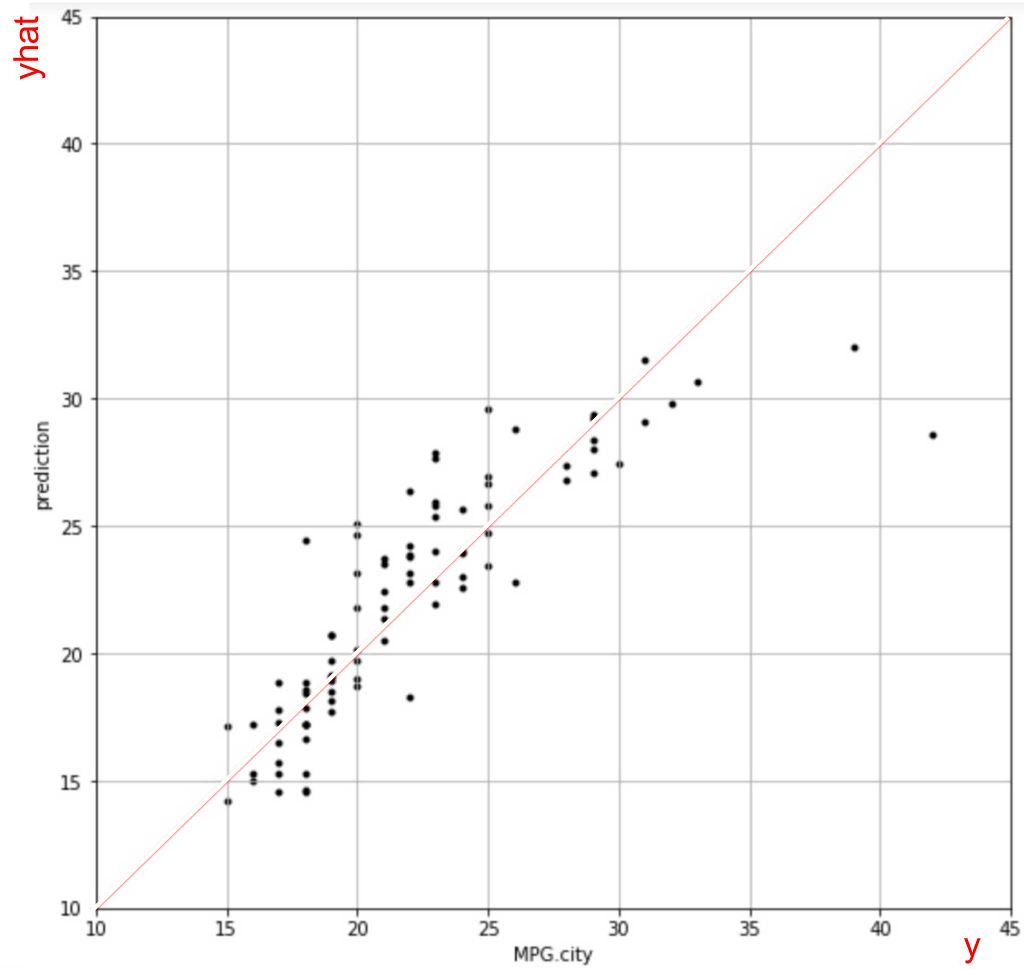
EXAMPLE 1 – y vs yhat

- predict city mileage for each car in the dataset

```
yhat = m1.fittedvalues
```

display the scatterplot

```
plt.figure(figsize = (9,9))
plt.scatter(y0,yhat,s=9,color='k')
```



EXAMPLE 1 – y vs yhat

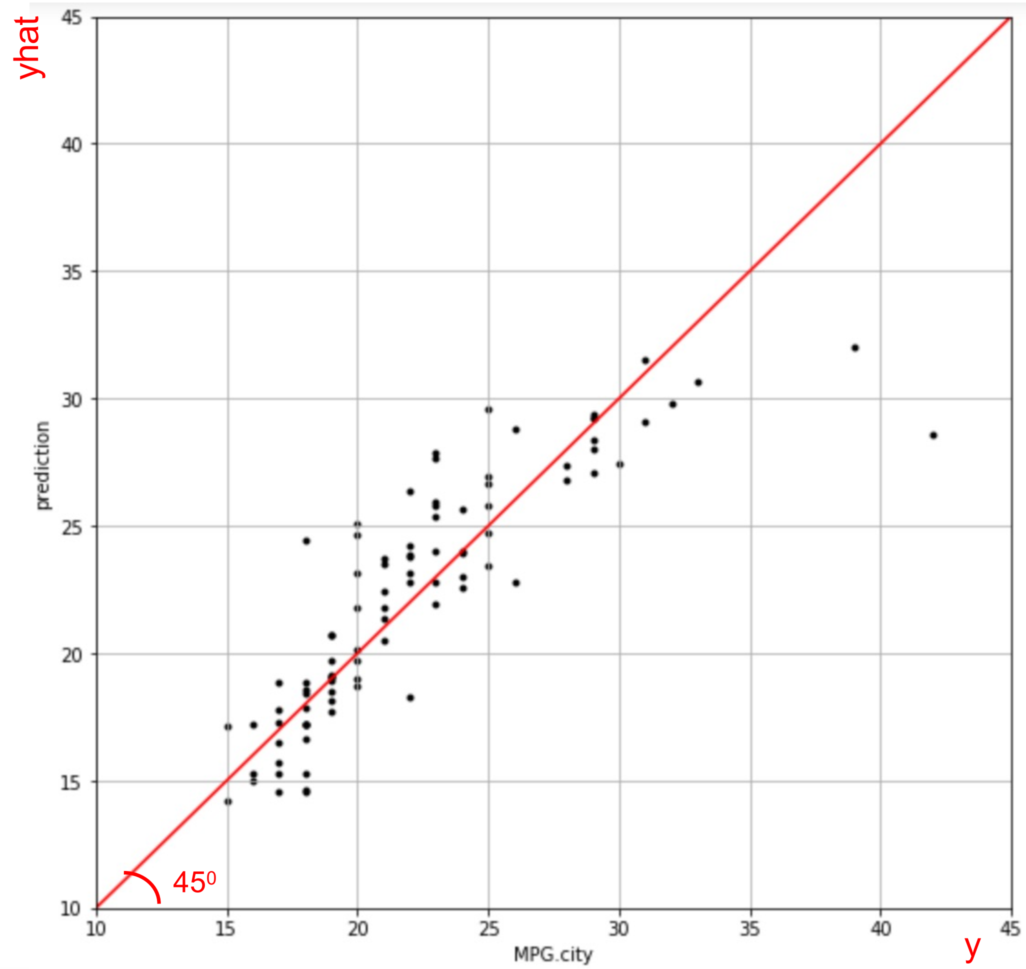
- predict city mileage for each car in the dataset

```
yhat = m1.fittedvalues
```

```
# Divide (10,45) into 100 segments.
# Store values in xvals
```

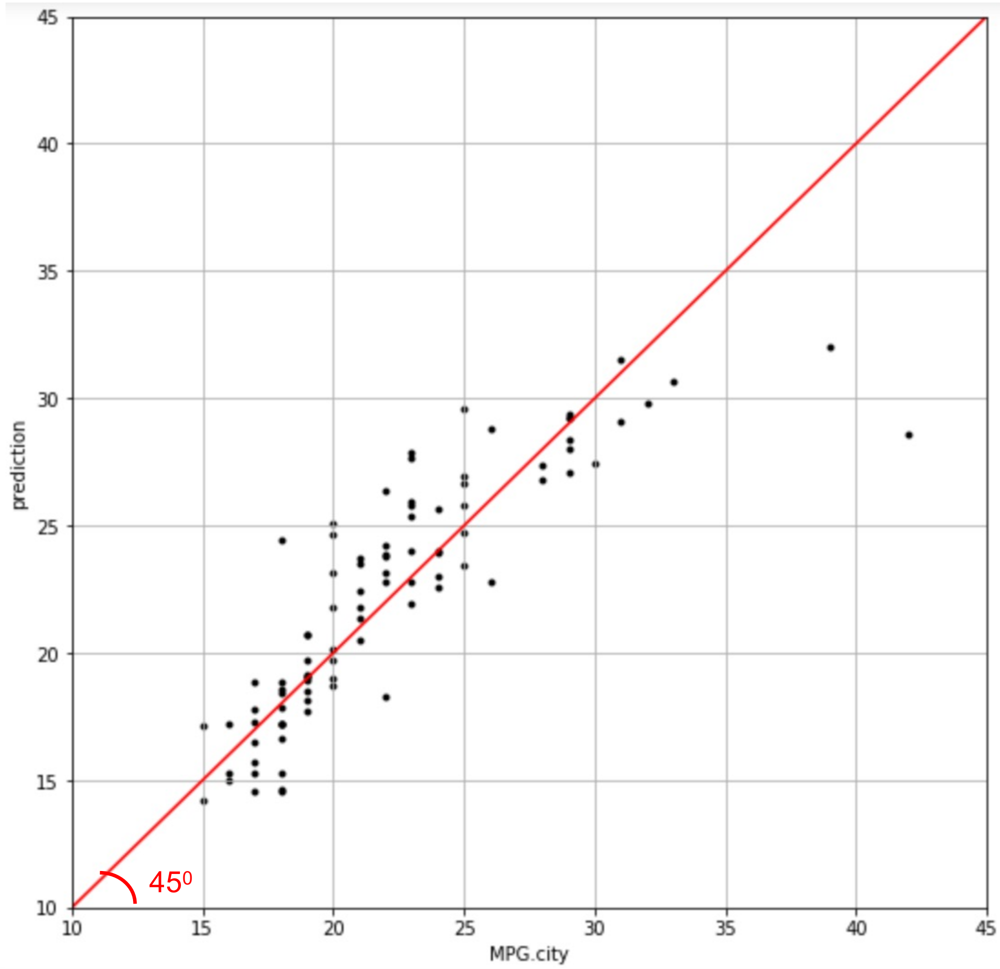
```
xvals = np.linspace(10,45,100)
yvals = xvals
```

```
plt.figure(figsize = (9,9))
plt.scatter(y0,yhat,s=9,color='k')
# plot diagonal line
plt.plot(xvals,yvals,color='r')
plt.xlim(10,45)
plt.ylim(10,45)
```



EXAMPLE 1 – y vs yhat

We would have a perfect fit if all points lie on the 45^0 line



EXAMPLE 1 – y vs yhat

We would have a perfect fit if all points lie on the 45^0 line

For this model, R^2 is 0.732

Thus, this set of predictors explain 73.2% of MPG variability

```
m1 = sm.OLS(y0,X1).fit()
m1.summary()
```

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

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

EXAMPLE 1

Questions to be answered

- What are the best predictors?
- How good is the model for prediction?