Data Analysis with Python

Question

what does each line in the **csv** file represent

- (●) row
- ( ) column
- ( ) header

Correct
correct, check the video

What is the name of the attribute what we want to predict

- ( ) dataframe
- ( ) feature
- (●) target (label)

Correct
correct

1. Each column contains a:

- (●) attribute or feature
- ( ) different used car

✓ **Correct**
correct

What is the primary instrument used in Pandas?

- ( ) Arrays
- (●) Dataframes
- ( ) Matrices

✓ **Correct**
correct

Scientifics Computing Libraries in Python

1. **Scientifics Computing** Libraries

Pandas
(Data structures & tools)

NumPy
(Arrays & matrices)

SciPy
(Integrals, solving differential equations, optimization)

What is the use of pandas?
Pandas is a popular and powerful open-source library in Python for data manipulation and analysis. It offers data structures and functions that simplify working with structured data, such as tables and time series data. Here are some key uses of pandas:

1. **Data Manipulation:** Pandas provides a wide range of tools for data manipulation. You can perform tasks such as filtering, sorting, grouping, merging, and reshaping data, making it easy to clean and transform datasets.

2. **Data Cleaning:** Pandas offers functions to handle missing data, remove duplicates, and perform data imputation. It allows you to identify and handle inconsistent or erroneous data, improving data quality.

3. **Data Analysis:** Pandas provides statistical and analytical functions that facilitate data analysis. You can compute descriptive statistics, aggregate data, apply mathematical operations, and perform calculations on data columns.

4. **Time Series Analysis:** Pandas has robust support for working with time series data. It allows you to handle time-based indexing, resampling, shifting, and rolling window calculations. This makes it convenient for analyzing and modeling time-dependent data.

5. **Data Visualization:** Pandas integrates well with visualization libraries like Matplotlib and Seaborn. It provides functions to generate plots, charts, and graphs to visualize data and gain insights.

6. **Data Input/Output:** Pandas supports reading and writing data in various formats, such as CSV, Excel, SQL databases, JSON, and more. It simplifies data import and export tasks, enabling seamless integration with external data sources.

7. **Integration with Other Libraries:** Pandas seamlessly integrates with other libraries in the scientific Python ecosystem, such as NumPy, Scikit-learn, and TensorFlow. It enhances data analysis workflows by leveraging the capabilities of these libraries.

8. **Data Preprocessing for Machine Learning:** Pandas is commonly used for data preprocessing tasks in machine learning. It helps in preparing data for model training by handling missing values, encoding categorical variables, and scaling numerical features.

Overall, pandas is widely used in data science, research, finance, business analytics, and many other fields where structured data analysis is required. It simplifies data manipulation, enables efficient data exploration, and provides the foundation for data-driven decision making.

What is the use of numpy?

1. **NumPy (Numerical Python) is a fundamental library in Python for scientific computing and data analysis. It provides powerful tools for working with large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.**
2. **Efficient Array Operations: NumPy provides an ndarray object, which is an efficient container for large arrays of homogeneous data. It allows you to perform element-wise operations, array slicing, reshaping, and indexing, making it easier and faster to manipulate and process arrays compared to regular Python lists.**

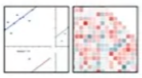
3. **Numerical Computations: NumPy offers a comprehensive set of mathematical functions for array computations. These include basic mathematical operations (e.g., addition, subtraction, multiplication, division), advanced mathematical functions (e.g., trigonometric, exponential, logarithmic functions), linear algebra operations (e.g., matrix multiplication, eigenvalues, solving linear equations), and statistical functions (e.g., mean, median, standard deviation).**

4. **Data Representation: NumPy arrays are commonly used to represent and manipulate data in various scientific and numerical applications. They can store numerical data, such as sensor readings, experimental results, or image pixels, in a memory-efficient and computationally efficient manner. NumPy arrays are the foundation for many other libraries in the scientific Python ecosystem.**

Visualization Libraries in Python

2. **Visualization** Libraries

Matplotlib
(plots & graphs, most popular)



Seaborn
(plots : heat maps, time series, violin plots)

Algorithmic Libraries in Python

3. **Algorithmic** libraries

Scikit-learn
(Machine Learning : regression, classification,…)

Statsmodels
(Explore data, estimate statistical models, and perform statistical tests.)

what libraries do you use for data visualization.:

- (●) matplotlib
- ( ) numpy
- ( ) scikit-learn

✓ **Correct**
correct

1. What description best describes the library Pandas? **1 / 1 point**

- ( ) Includes functions for some advanced math problems as listed in the slide as well as data visualization.
- (●) Offers data structure and tools for effective data manipulation and analysis. It provides fast access to structured data. The primary instrument of Pandas is a two-dimensional table consisting of columns and rows labels which are called a DataFrame. It is designed to provide an easy indexing function.
- ( ) Uses arrays as their inputs and outputs. It can be extended to objects for matrices, and with a little change of coding, developers perform fast array processing.

✓ **Correct**
correct

what method was applied on the dataframe **df** to get the following output :

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

- (●) df.head()

Exporting to different formats in Python

Data Format	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
Excel	pd.read_excel()	df.to_excel()
sql	pd.read_sql()	df.to_sql()

1. What task does the following lines of code perform? **1 / 1 point**

```
path='C:\Windows\…\ automobile.csv'
df.to_csv(path)
```

- (●) Exports your Pandas dataframe to a new csv file, in the location specified by the variable path.
- ( ) Loads a csv file.

Correct
correct

Basic Insights of Dataset - Data Types

Pandas Type	Native Python Type	Description
object	string	numbers and strings
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	time data.

dataframe.describe()

- Returns a statistical summary

```
df.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

what data type do you expect the column that contains price to be

- ○ object
- ◉ float64
- ○ datetime64

✓ **Correct**

correct

dataframe.describe(include="all")

- Provides full summary statistics

```
df.describe(include="all")
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000	205	205	205
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	NaN	NaN	NaN
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN

how would you generate descriptive statistics for **all** the columns for the dataframe **df**

○

```
1  df.describe()
```

◉

```
1  df.describe(include = "all")
```

1. What is the correct output of? 1 / 1 point

```
1   df.describe(include="all")
```

○ a)

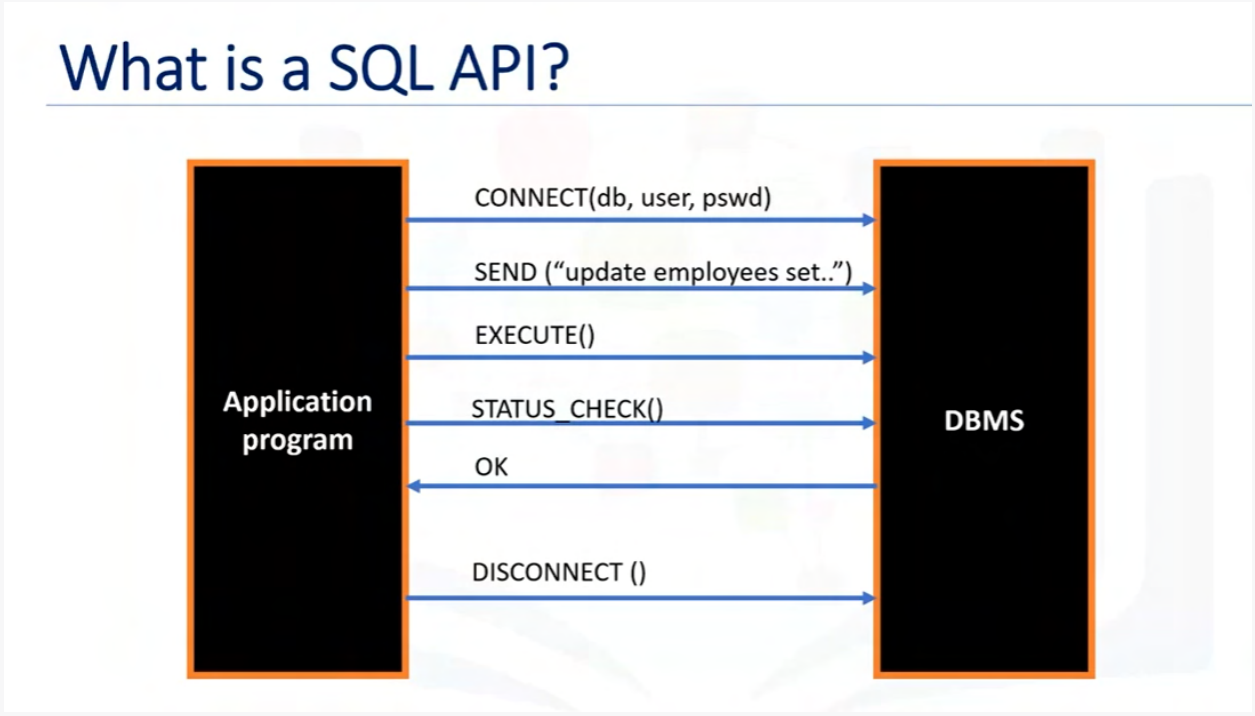
	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

◉ b)

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000	205	205	205
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	NaN	NaN	NaN
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN

✓ Correct
correct

Accessing databases using Python



Define the Business Problem: Look at the data and make some high-level decision on what kind of analysis should be done

Import and Export Data in Python: How to import data from multiple data sources using the Pandas library and how to export files into different formats.

Analyze Data in Python: How to do some introductory analysis in Python using functions like dataframe.head() to view the first few lines of the dataset, dataframe.info() to view the column names and data types.

Graded Quiz: Importing Datasets

week-1

1. What does csv stand for? **1 point**

- (•) Comma Separated Values
- ( ) Car Sold values
- ( ) none of the above

2. Select the libraries you will use for this course? **1 point**

- [x] matplotlib
- [x] pandas
- [x] scikit-learn
- [ ] TensorFlow

3. We have the list **headers_list:** **1 point**

```
1  headers_list=['A','B','C']
2
```

We also have the dataframe **df** that contains three columns, what is the correct syntax to replace the headers of the dataframe df with values in the list **headers_list?**

- (•) **df.columns = headers_list**
- ( ) **df.head()**
- ( ) **df.tail()**

4. Consider the segment of the following dataframe: 1 point

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi

What is the type of the column **make**?

- (●) object
- ( ) int64
- ( ) float64

5. How would you generate descriptive statistics for all the columns for the dataframe **df?** 1 point

( )
```
1  df.describe()
```

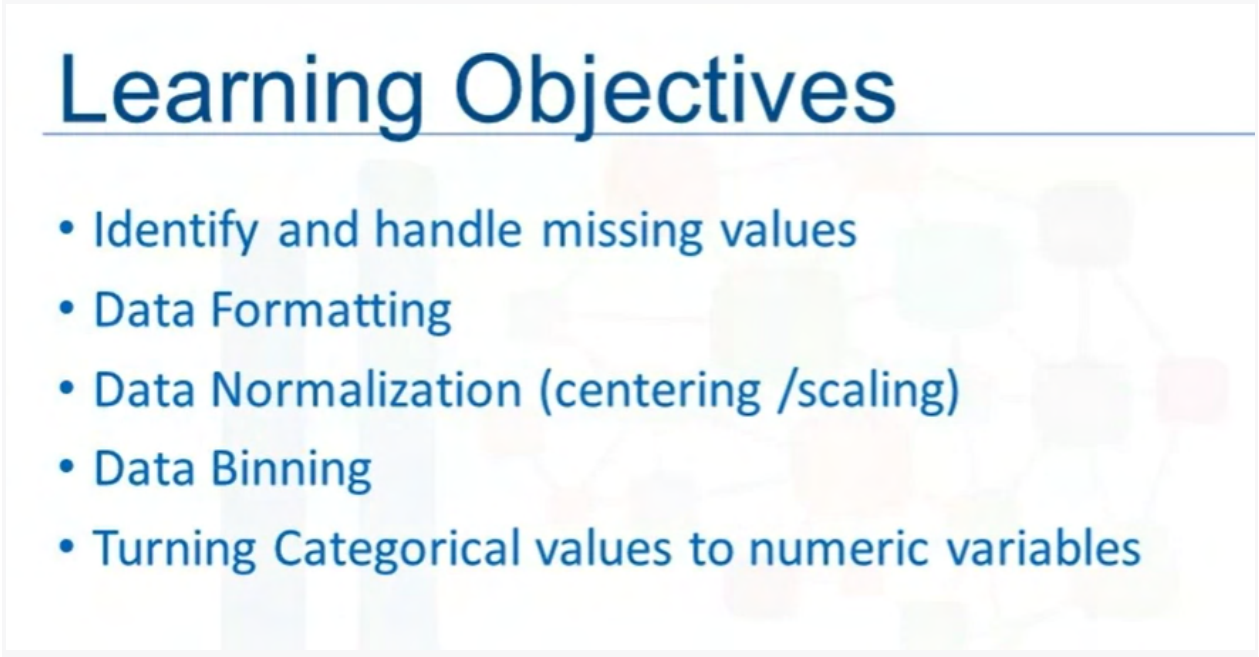
(●)
```
1  df.describe(include = "all")
```

Data Pre-processing

Also known as:

Data Cleaning, Data Wrangling

The process of converting or mapping data from the initial “raw” form into another format, in order to prepare the data for further analysis.



What Is Data Wrangling? Data Wrangling is the process of collecting, and modifying the raw data into another form for analyzing and decision-making easily. It is also known as Data Munging. For example, when we want only some part of the data that is useful based on the application, then we can do data wrangling.

How to deal with missing data?

Check with the data collection source

Drop the missing values

- drop the variable
- drop the data entry

Replace the missing values

- replace it with an average (of similar datapoints)
- replace it by frequency
- replace it based on other functions

Leave it as missing data

how would you deal with missing values for categorical data

- [x] replace the missing value with the mode of the particular column

Correct
correct, the mode is the value that appears most often

- [ ] replace the missing value with the mean of the particular column
- [x] replace the missing value with the value that appears most often of the particular column

Correct
correct, this is called the mode

what does the following line of code do to the dataframe **df**:

```
1   df.dropna(axis=0)
```

- ○ replaces all values **nan** with the mean
- ◉ drops all rows that contain a **nan**
- ○ drops all columns that contain a **nan**

✓ **Correct**
correct

1. How would you access the column "body-style" from the dataframe **df?** 1 / 1 point

```
df[ "body-style"]
```

```
df==”bodystyle"
```

Correct
correct

2. What is the correct symbol for missing data? 1 / 1 point

- nan
- no-data

Correct
correct

Data Formatting

- Data are usually collected from different places and stored in different formats.
- Bringing data into a common standard of expression allows users to make meaningful comparison.

Non-formatted:
- confusing
- hard to aggregate
- hard to compare

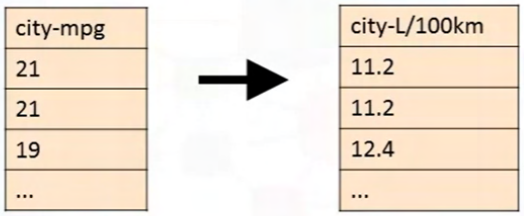
City
NY
New York
N.Y
N.Y

City
New York
New York
New York
New York

Formatted:
- more clear
- easy to aggregate
- easy to compare

Applying calculations to an entire column

- Convert ”mpg” to “L/100km” in Car dataset.




```
df[”city-mpg”]= 235/df[“city-mpg”]
```

Incorrect data types

- Sometimes the wrong data type is assigned to a feature.

```
df[“price”].tail(5)

200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, dtype: object
```

Correcting data types

To *identify* data types:

- Use **`dataframe.dtypes()`** to identify data type.

To *convert* data types:

- Use **`dataframe.astype()`** to convert data type.

Example: convert data type to integer in column “price”

```
df["price"] = df["price"].astype("int")
```

what task does the following line of code perform:

```
1  df[["price"]] = df[["price"]].astype("float")
```

- (●) cast the column price to a **float**
- ( ) cast the column price to an **int**
- ( ) cast the column float to a price

✓ **Correct**

correct

1. How would you rename the column "city_mpg" to "city-L/100km"? **1 / 1 point**

```
df.rename(columns={"city_mpg": "city-L/100km"}, inplace=True)
```

```
df.rename(columns={"city_mpg": "city-L/100km"})
```

Correct
correct

Data Normalization

- Uniform the features value with different range.

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
171.2	65.5	52.4
176.6	66.2	54.3
176.6	66.4	54.3
177.3	66.3	53.1
192.7	71.4	55.7
192.7	71.4	55.7
192.7	71.4	55.9

scale	[150,250]	[50,100]	[50,100]
impact	large	small	small

Data Normalization

age	income
20	100000
30	20000
40	500000

age	income
0.2	0.2
0.3	0.04
0.4	1

Not-normalized

- "age" and "income" are in different range.
- hard to compare
- "income" will influence the result more

Normalized

- similar value range.
- similar intrinsic influence on analytical model.

Methods of normalizing data

Several approaches for normalization:

① $x_{new} = \frac{x_{old}}{x_{max}}$ **Simple Feature scaling**

② $x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$ **Min-Max**

③ $x_{new} = \frac{x_{old} - \mu}{\sigma}$ **Z-score**

z-score values typically range between 0 to 1

- (●) False
- ( ) True

✓ **Correct**
correct

Simple Feature Scaling in Python

With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...	...	...

→

length	width	height
0.81	64.1	48.8
0.81	64.1	48.8
0.87	65.5	52.4
...	...	...

```
df["length"] = df["length"]/df["length"].max()
```

Min-max in Python

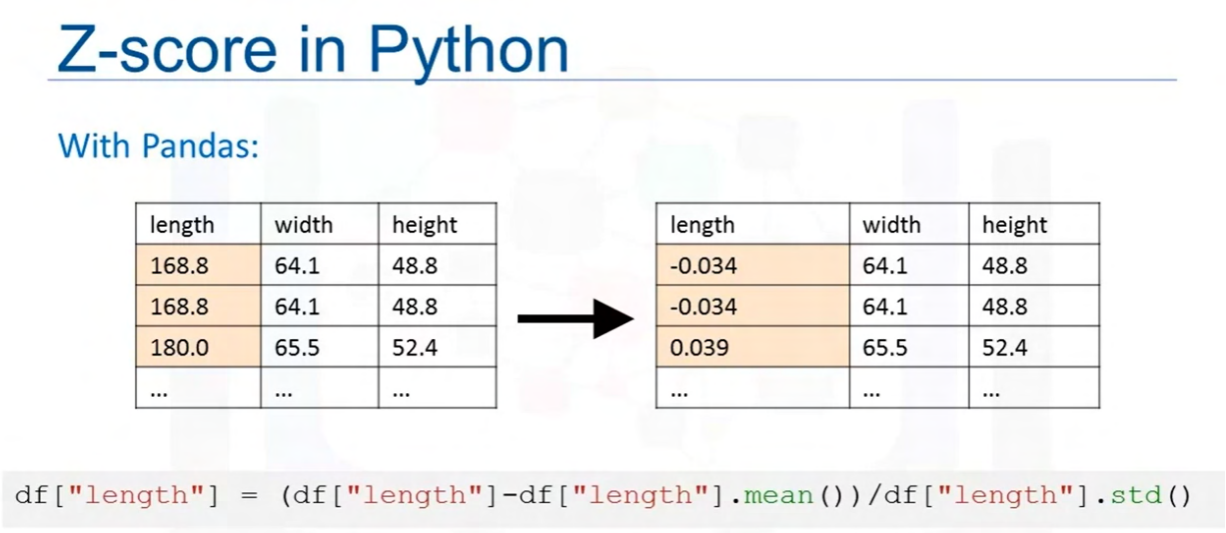
With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...	...	...

→

length	width	height
0.41	64.1	48.8
0.41	64.1	48.8
0.58	65.5	52.4
...	...	...

```
df["length"] = (df["length"]-df["length"].min())/
               (df["length"].max()-df["length"].min())
```



z-score values range between?

ChatGPT

Z-score values can range from negative infinity to positive infinity. The purpose of calculating the z-score is to standardize a value by measuring how many standard deviations it is away from the mean. A positive z-score indicates that the value is above the mean, while a negative z-score indicates that it is below the mean.

The z-score formula is as follows:

makefile

Copy code

Here, `x` represents the original value, `mean` represents the mean of the data, and `standard deviation` represents the standard deviation of the data.

The resulting z-score can be any real number, depending on the original value's distance from the mean. However, it is more common for the majority of z-score values to fall within the range of -3 to +3. Values outside this range indicate that the original value is relatively far from the mean, either in the positive or negative direction.

It's worth noting that z-scores beyond -3 to +3 are not inherently incorrect or invalid. They simply indicate that the value is relatively extreme in comparison to the rest of the data.

What is the maximum value of the feature scaling?

In feature scaling, the maximum value of the scaled feature depends on the specific scaling method being used. Two commonly used scaling methods are Min-Max scaling and Standardization (Z-score scaling).

> Min-Max Scaling: In Min-Max scaling, the feature values are transformed to a fixed range, typically between 0 and 1. The maximum value of the scaled feature will be 1.
> Standardization (Z-score Scaling): In Standardization, the feature values are transformed to have a mean of 0 and a standard deviation of 1. Unlike Min-Max scaling, Standardization does not have a predefined maximum value since it adjusts the values based on their distribution. The maximum value in the original data may not necessarily correspond to a maximum value in the standardized data.

It's important to note that feature scaling is performed to normalize the range of feature values and does not inherently define a maximum value. The choice of scaling method depends on the specific requirements of the data and the machine learning algorithm being used.

1. Which of the following is the correct formula for z -score or data standardization? 1 / 1 point

$x_{new} = \frac{x_{old}}{x_{max}}$ a

$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$ b

$x_{new} = \frac{x_{old} - \mu}{\sigma}$ c

c

Correct
You simply subtract the mean and divide by the standard deviation.

Binning

- Binning: Grouping of values into ”bins”
- Converts numeric into categorical variables
- Group a set of numerical values into a set of “bins”
- “price” is a feature range from 5,000 to 45,500
  (in order to have a **better representation** of price)

price: 5000, 10000,12000,12000, 30000, 31000, 39000, 44000,44500

bins: low Mid High

Turning categorical variables into quantitative variables in Python

Categorical → Numeric

Solution:

- Add dummy variables for each unique category
- Assign 0 or 1 in each category

Car	Fuel	...	gas	diesel
A	gas	...	1	0
B	diesel	...	0	1
C	gas	...	1	0
D	gas	...	1	0

"One-hot encoding"

Dummy variables in Python pandas

- Use pandas.get_dummies() method.
- Convert categorical variables to dummy variables (0 or 1)

fuel
gas
diesel
gas
gas

```
pd.get_dummies(df['fuel'])
```

Dummy variables in Python pandas

- Use pandas.get_dummies() method.
- Convert categorical variables to dummy variables (0 or 1)

fuel
gas
diesel
gas
gas

gas	diesel
1	0
0	1
1	0
1	0

```
pd.get_dummies(df['fuel'])
```

1. Consider the column 'diesel'; what should the value for Car B be? 1 / 1 point

Car	Fuel	...	gas	diesel
A	gas	...	1	0
B	diesel	...	0	
C	gas	...	1	0
D	gas	...	1	0

- ○ 0
- ◉ 1

Correct
correct

Identify and Handle Missing Values: Drop rows with incomplete information and impute missing data using the mean values.
Understand Data Formatting: Wrangle features in a dataset and make them meaningful for data analysis.

Apply normalization to a data set: By understanding the relevance of using feature scaling on your data and how normalization and standardization have varying effects on your data analysis.

Graded Quiz Week-2

1. What task do the following lines of code perform? **1 point**

```
avg=df['horsepower'].mean(axis=0)
df['horsepower'].replace(np.nan, avg)
```

- ○ replace all the NaN values with the mean
- ◉ calculate the mean value for the **'horsepower'** column and replace all the NaN values of that column by the mean value
- ○ nothing; because the parameter **inplace** is not set to true

2. Consider the dataframe df; convert the column df["city-mpg"] to df["city-L/100km'] by dividing 235 by each element in the column 'city-mpg'. **1 point**

◉
```
df['city-L/100km'] = 235/df["city-mpg"]
```

○
```
df['city-L/100km'] = df["city-mpg"].diV(235)
```

3. How would you cast the column **"losses"** to an integer? 1 point

```
df[["losses"]]=df[["losses"]].astype("int")
```

```
df[["losses"]].astype("int")
```

4. The following code is an example of: 1 point

```
(df["length"]-df["length"].mean())/df["length"].std()
```

- simple feature scaling
- min-max scaling
- z-score

5. Consider the two columns 'horsepower', and 'horsepower-binned'; from the dataframe **df**; how many categories are there in the 'horsepower-binned' column? 1 point

	horsepower	horsepower-binned
0	111.0	Medium
1	111.0	Medium
2	154.0	Medium
3	102.0	Medium
4	115.0	Medium
5	110.0	Medium
6	110.0	Medium
7	110.0	Medium
8	140.0	Medium
9	101.0	Low
10	101.0	Low
11	121.0	Medium
12	121.0	Medium
13	121.0	Medium
14	182.0	High
15	182.0	High
16	182.0	High
17	48.0	Low
18	70.0	Low
19	70.0	Low

3

What happens if the method describe is applied to a dataframe with NaN values

- an error will occur
- all the statistics calculated using NaN values will also be NaN
- NaN values will be excluded

Correct
correct

WEEK-3

Grouping data

- Use Panda **dataframe. Groupby()** method:
  - Can be applied on categorical variables
  - Group data into categories
  - Single or multiple variables

How would you use the **groupby** function to find the average "price" of each car based on "body-style" ?

```
1   df[['price','body-style']].groupby(['body-style'],as_index= False).mean()
```

Groupby()- Example

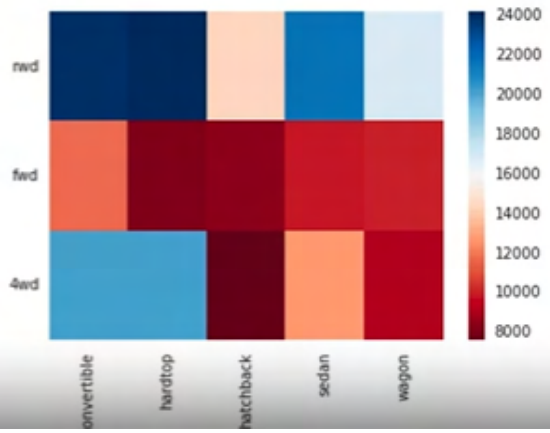
```
df_test = df[['drive-wheels', 'body-style', 'price']]
df_grp = df_test.groupby(['drive-wheels', 'body-style'], as_index=False).mean()
df_grp
```

	drive-wheels	body-style	price
0	4wd	convertible	20239.229524
1	4wd	sedan	12647.333333
2	4wd	wagon	9095.750000
3	fwd	convertible	11595.000000
4	fwd	hardtop	8249.000000
5	fwd	hatchback	8396.387755
6	fwd	sedan	9811.800000
7	fwd	wagon	9997.333333
8	rwd	convertible	23949.600000
9	rwd	hardtop	24202.714286
10	rwd	hatchback	14337.777778
11	rwd	sedan	21711.833333
12	rwd	wagon	16994.222222

Heatmap

- Plot target variable over multiple variables

```
plt.pcolor(df_pivot, cmap='RdBu')
plt.colorbar()
plt.show()
```



1. Select the appropriate description of a pivot table:

- (●) A pivot table has one variable displayed along the columns and the other variable displayed along the rows.
- ( ) A pivot table contains statistical information for each column

✓ **Correct**
correct

Correlation

What is Correlation?

- Measures to what extent different variables are interdependent.

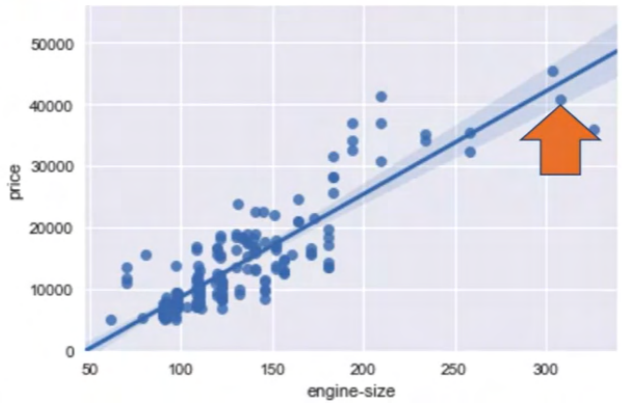
- For example:
  - Lung cancer → Smoking
  - Rain → Umbrella

- Correlation doesn't imply causation.

Correlation - Positive Linear Relationship

- Correlation between two features (engine-size and price).

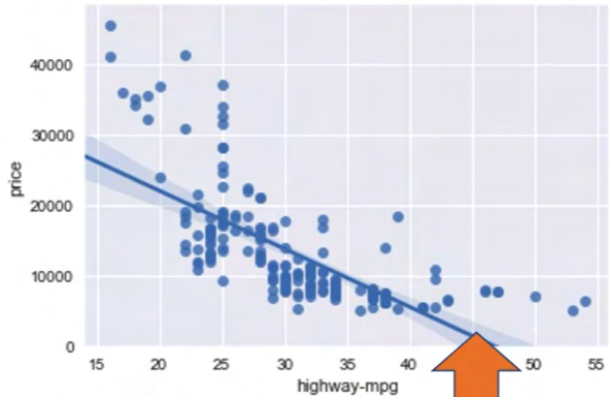
```
sns.regplot(x="engine-size", y= "price", data=df)
plt.ylim(0,)
```



Correlation - Negative Linear Relationship

- Correlation between two features (highway-mpg and price).

```
sns.regplot(x="highway-mpg", y= "price", data=df)
plt.ylim(0,)
```



Pearson Correlation

- Measure the strength of the correlation between two features.
  - Correlation coefficient
  - P-value

- Correlation coefficient
  - Close to +1: Large Positive relationship
  - Close to -1: Large Negative relationship
  - Close to 0: No relationship

- P-value
  - P-value < 0.001 **Strong** certainty in the result
  - P-value < 0.05 **Moderate** certainty in the result
  - P-value < 0.1 **Weak** certainty in the result
  - P-value > 0.1 **No** certainty in the result

Pearson Correlation

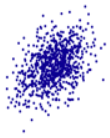
- Measure the strength of the correlation between two features.
  - Correlation coefficient
  - P-value

- Correlation coefficient
  - Close to +1: Large Positive relationship
  - Close to -1: Large Negative relationship
  - Close to 0: No relationship

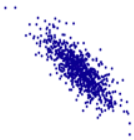
- P-value
  - P-value < 0.001 **Strong** certainty in the result
  - P-value < 0.05 **Moderate** certainty in the result
  - P-value < 0.1 **Weak** certainty in the result
  - P-value > 0.1 **No** certainty in the result

- Strong Correlation:
  - Correlation coefficient close to 1 or -1
  - P value less than 0.001

1. Select the plot with a negative correlation: 1 / 1 point



a



b

Correct
correct

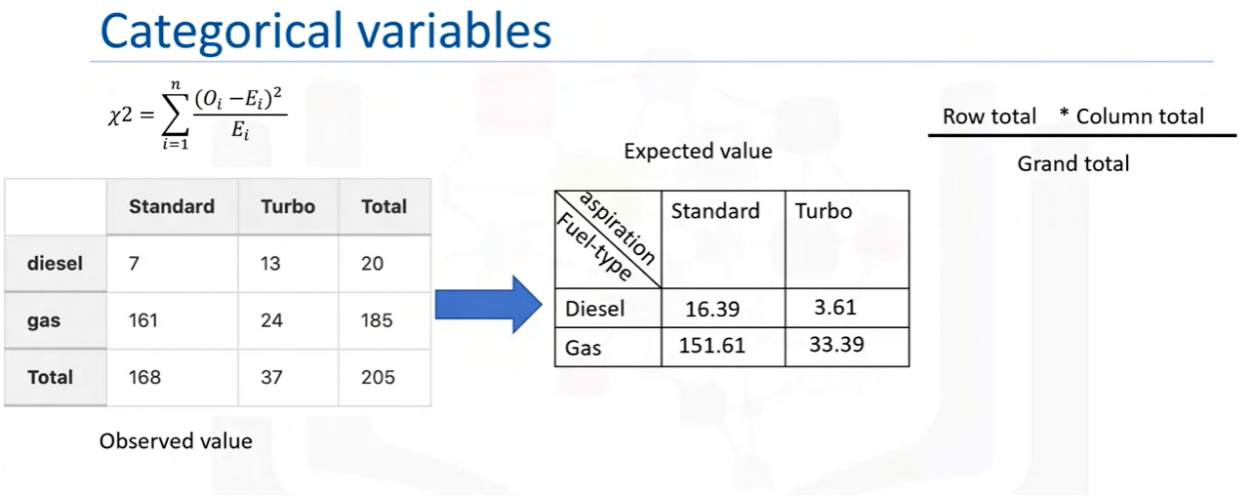
Categorical variables

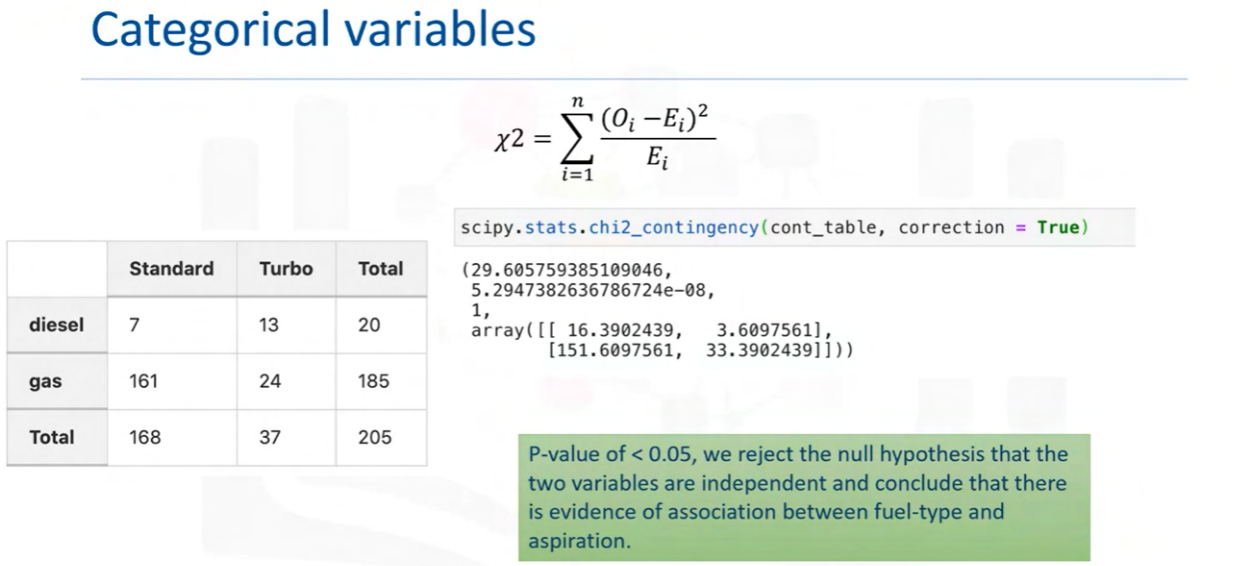
- Is there an association between fuel-type and aspiration?

$$\chi 2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Observed value

	Standard	Turbo	Total
diesel	7	13	20
gas	161	24	185
Total	168	37	205





Describe Exploratory Data Analysis: By summarizing the main characteristics of the data and extracting valuable insights.
Compute basic descriptive statistics: Calculate the mean, median, and mode using python and use it as a basis in understanding the distribution of the data.
Create data groups: How and why you put continuous data in groups and how to visualize them.
Define correlation as the linear association between two numerical variables: Use Pearson correlation as a measure of the correlation between two continuous variables

Define the association between two categorical variables:
Understand how to find the association of two variables using the Chi-square test for association and how to interpret them.

Week-3 Graded Quiz: Exploratory Data Analysis

1. What task does the method **value_counts** perform? 1 point

- ○ Returns summary statistics
- ◉ Returns counts of unique values
- ○ Returns the first five columns of a dataframe

2. If we have 10 columns and 100 samples, how large is the output of **df.corr()**? 1 point

- ○ 10 x 100
- ◉ 10x10
- ○ 100x100

3. If the Pearson Correlation between two variables is zero, then ... 1 point

- ○ The two variables have zero mean
- ◉ The two variables are not correlated

4. Consider the dataframe **df**;what method displays the first five rows of a dataframe? 1 point

- ○ df.describe()
- ◉ df.head()
- ○ df.tail()

4. Consider the dataframe **df**;what method displays the first five rows of a dataframe? 1 point

- ○ df.describe()
- ◉ df.head()
- ○ df.tail()

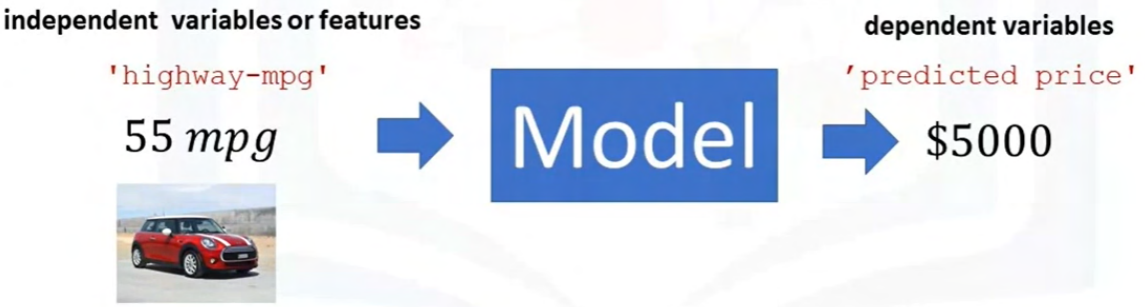
5. What is the Pearson Correlation between variables X and Y, if X = -Y? 1 point

- ◉ -1
- ○ 1
- ○ 0

WEEK - 4

Model Development

- A model can be thought of as a mathematical equation used to predict a value given one or more other values
- Relating one or more independent variables to dependent variables.



Model Development

- Usually the more **relevant data** you have the more accurate your model is

Model Development

To understand why more data is important consider the following situation:

1. you have two almost identical cars
2. Pink cars sell for significantly less

'highway-mpg'
'curb-weight'
'engine-size'
'highway-mpg'

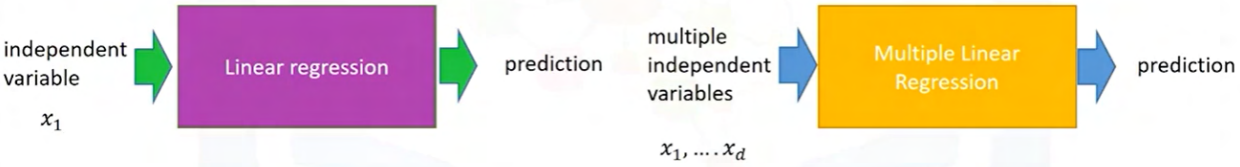
→ Model → $Y = \$5400$

'highway-mpg'
'curb-weight'
'engine-size'
'highway-mpg'

→ Model → $Y = \$5400$

Introduction

- Linear regression will refer to one independent variable to make a prediction
- Multiple Linear Regression will refer to multiple independent variables to make a prediction



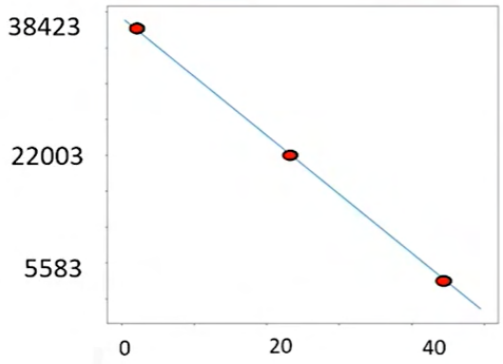
Simple Linear Regression

1. The predictor (independent) variable - x
2. The target (dependent) variable - y

$$y = b_0 + b_1 x$$

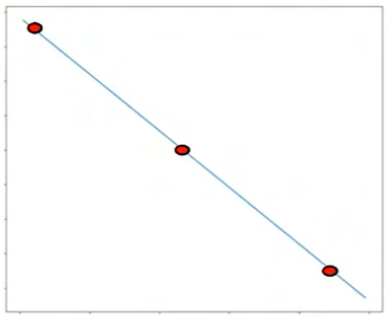
- b_0: **the intercept**
- b_1 : **the slope**

Simple Linear Regression: Fit



$$X = \begin{bmatrix} \quad \end{bmatrix} \quad Y = \begin{bmatrix} \quad \end{bmatrix}$$

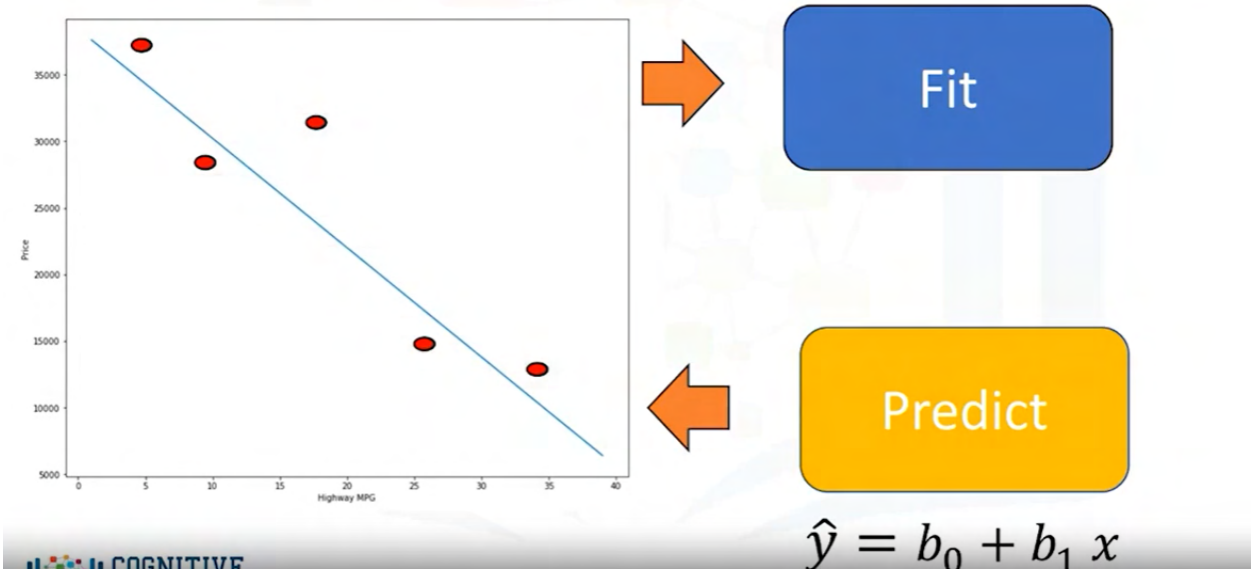
Simple Linear Regression: Fit



$$X = \begin{bmatrix} 0 \\ 20 \\ 40 \end{bmatrix} \quad Y = \begin{bmatrix} 38423 \\ 22003 \\ 5583 \end{bmatrix}$$

- Trained points/data is used to build the model
- Simple linear regression maps independent variable x to output variable y using the model
- Model is nothing but the mapping function or an mathematical equation used to generate the output labels from the given input variables

Simple Linear Regression



Fitting a Simple Linear Model Estimator

- X :Predictor variable
- Y: Target variable

1. Import linear_model from scikit-learn

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

2. Create a Linear Regression Object using the constructor :

```
lm=LinearRegression()
```

Fitting a Simple Linear Model

- We define the predictor variable and target variable

```
X = df[['highway-mpg']]
Y = df['price']
```

- Then use lm.fit (X, Y) to fit the model , i.e fine the parameters b_0 and b_1

- We can obtain a prediction

Consider the following lines
of code

```
from sklearn.linear_model import LinearRegression

lm=LinearRegression()
X = df[['highway-mpg']]
Y = df['price']

```

Type the next line of code to fit the model

```
lm.fit(X,Y)
```

Correct
Correct you just apply the method fit with the variables X
and Y as arguments

Fitting a Simple Linear Model

- We define the predictor variable and target variable

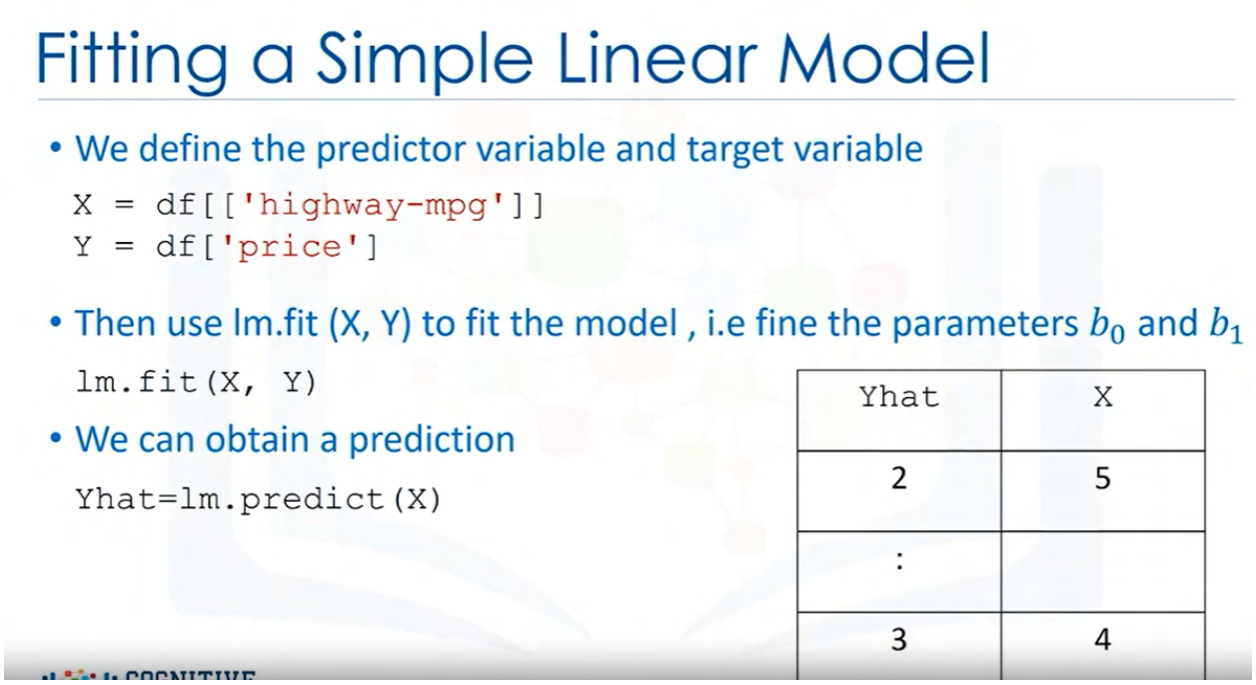
```
X = df[['highway-mpg']]
Y = df['price']
```

- Then use lm.fit (X, Y) to fit the model , i.e fine the parameters b_0 and b_1

```
lm.fit(X, Y)
```

- We can obtain a prediction

```
Yhat=lm.predict(X)
```



How is the model going to build and work is?

Linear regression is imported from the scikit-learn library
Created an object from the linear regression model
We will first keep out training data set from the dataset what we have
Then we will have x,y
x=independent variable (input) - feature
y=output variable (label)
Fit these both x,y columns into the linear regression model and refine the slope and intercept which are b0,b1 parameters of the model-mathematical equation
Now, we have our model.
Use this mathematical (linear equation) model for the testing dataset or use it for prediction

SLR – Estimated Linear Model

- We can view the intercept (b_0): `lm.intercept_` → `38423.305858`
- We can also view the slope (b_1): `lm.coef_` → `-821.73337832`
- The Relationship between Price and Highway MPG is given by:
- **Price = 38423.31 - 821.73 * highway-mpg**

$$\hat{Y} = b_0 + b_1 x$$

Multiple Linear Regression (MLR)

This method is used to explain the relationship between:

- One continuous target (Y) variable
- Two or more predictor (X) variables

Multiple Linear Regression (MLR)

$$\hat{Y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

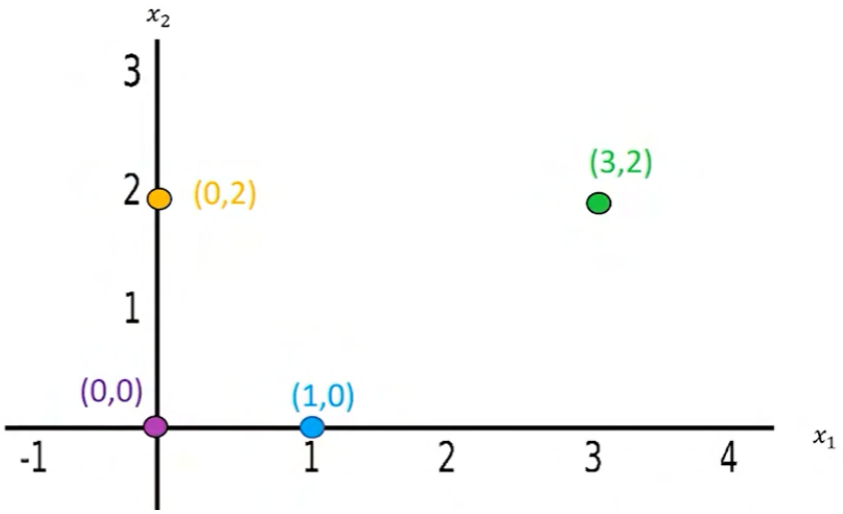
- b_0 : **intercept (X=0)**
- b_1: the **coefficient** or **parameter** of x_1
- b_2: the **coefficient** of **parameter** x_2 and so on..

Multiple Linear Regression (MLR)

$$\hat{Y} = 1 + 2x_1 + 3x_2$$

- The variables x_1 and x_2 can be visualized on a 2D plane, lets do an example on the next slide

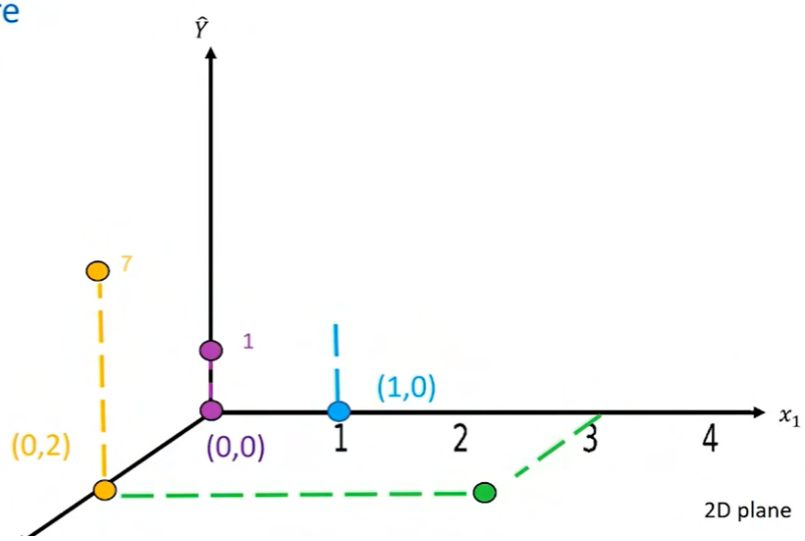
n	x_1	x_2
1	0	0
2	0	2
3	1	0
4	3	2



• This is shown below where

$$\hat{Y} = 1 + 2x_1 + 3x_2$$

n	x_1	x_2	$\hat{Y}$
1	0	0	1
2	0	2	7
3	1	0	3
4	3	2	13



Multiple linear regression

Two or more independent variables of the training dataset as multiple independent variables and then model is built by fitting the group of independent variables and output label.

Take the group of independent variables into a dataframe and fit the dataframe with the y label into the linear regression and generate a mathematical equation. Use this equation to predict the output for the testing dataset.

Fitting a Multiple Linear Model Estimator

1. We can extract the for 4 predictor variables and store them in the variable Z

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

2. Then train the model as before:

```
lm.fit(Z, df['price'])
```

3. We can also obtain a prediction

```
Yhat=lm.predict(X)
```

x_1	x_2	x_3	x_4
3	5	-4	3
:	:	:	:
2	4	2	-4

Yhat
2
:
3

MLR – Estimated Linear Model

1. Find the intercept (b_0)

```
lm.intercept_
-15678.742628061467
```

2. Find the coefficients (b_1, b_2, b_3, b_4)

```
lm.coef_
array([52.65851272 ,4.69878948,81.95906216 , 33.58258185])
```

The Estimated Linear Model:

- **Price** = -15678.74 + (52.66) * **horsepower** + (4.70) * **curb-weight** + (81.96) * **engine-size** + (33.58) * **highway-mpg**

$$\hat{Y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

X= predictor or independent variable or feature

Y= target or dependent variable or label

Week-4 practice quiz

1. consider the following lines of code, what variable contains the predicted values : 1 / 1

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
X = df[['highway-mpg']]
Y = df['price']
lm.fit(X, Y)
Yhat=lm.predict(X)
```

- ○ X
- ◉ Yhat
- ○ Y

✓ **Correct**
correct

2. consider the following equation: 1 / 1

$$y = b_0 + b_1 x$$

the variable y is?

Regression Plot

Why use regression plot?

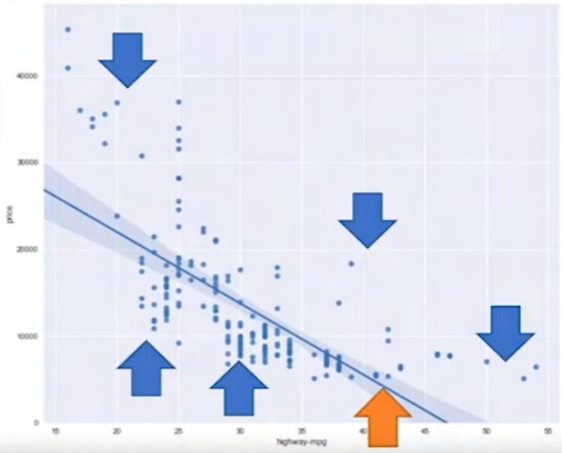
It gives us a good estimate of:

1. The relationship between two variables
2. The strength of the correlation
3. The direction of the relationship (positive or negative)

Regression Plot

Regression Plot shows us a combination of:

- The scatterplot: where each point represents a different y
- The fitted linear regression line (ŷ).

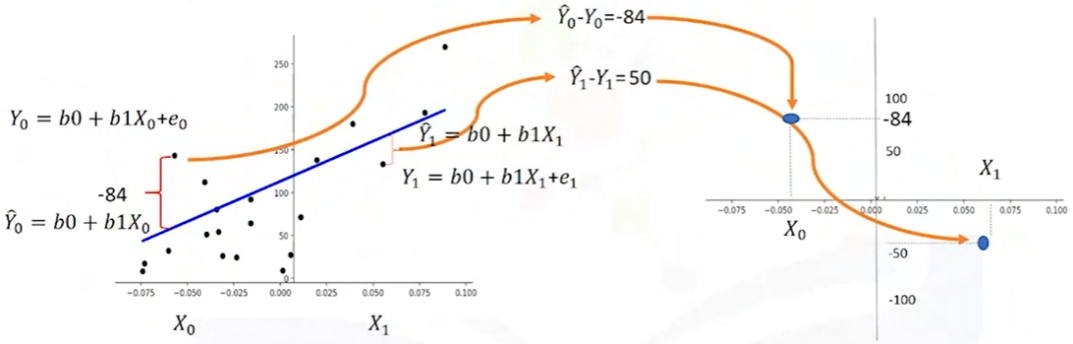


Regression Plot

```
import seaborn as sns

sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

Residual Plot

Y-axis: residuals

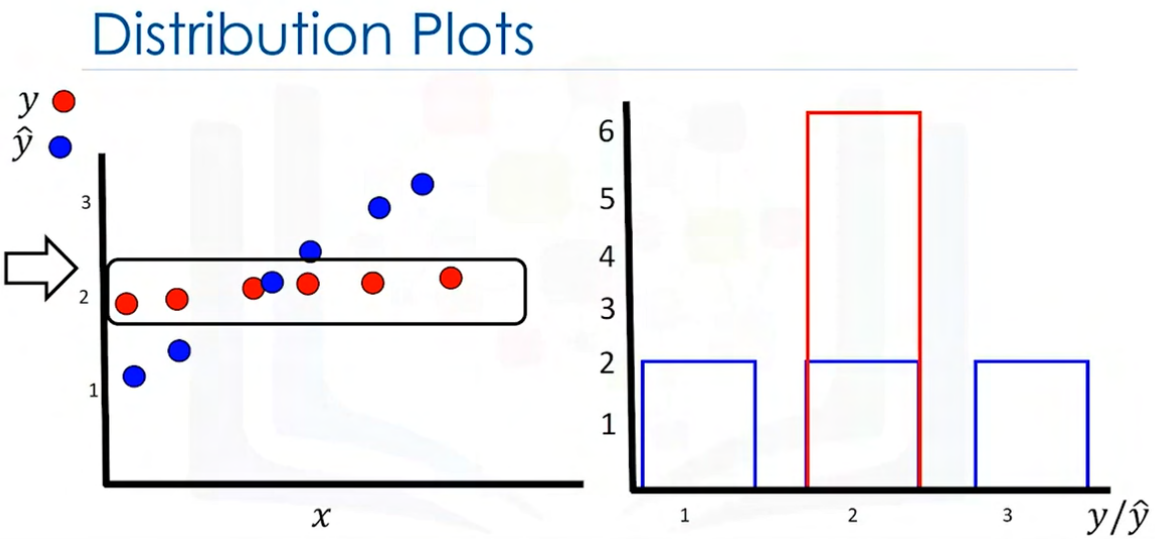


X-axis: the predictor variable or fitted values.

PLOT BETWEEN THE PREDICTED VALUES AND ACTUAL VALUES

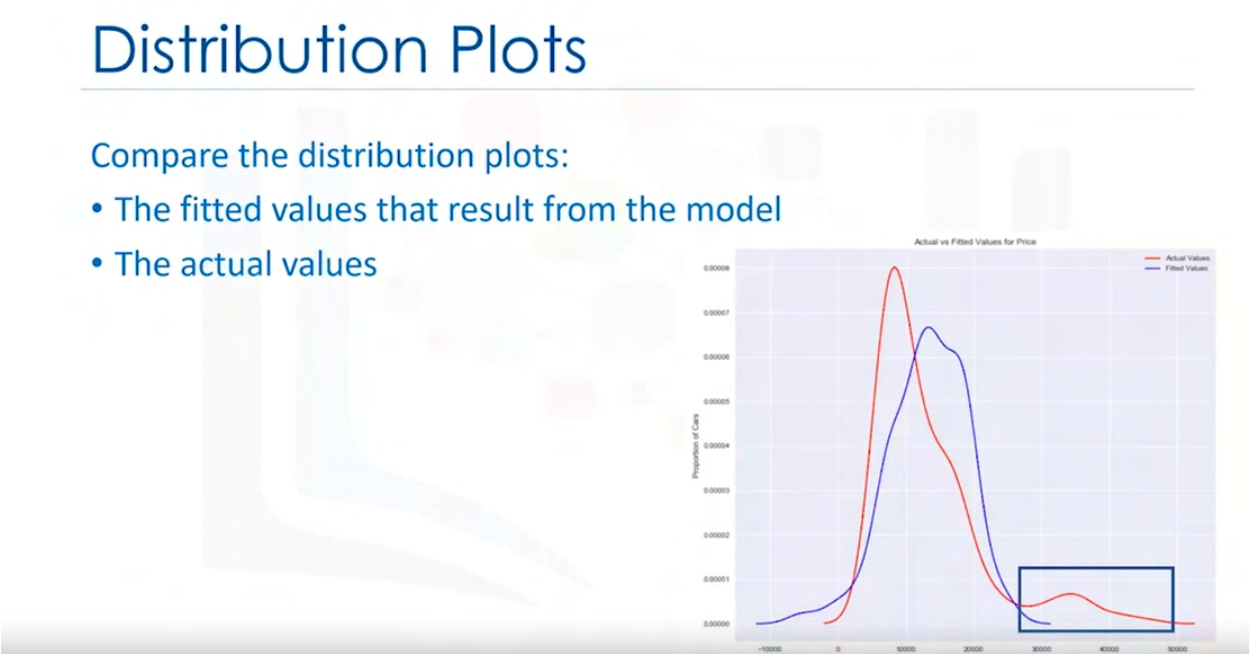
Distribution plots

Plot between predicted values and actual values



y=target values -actual values

y^=predicted values - fitted values that result from the model

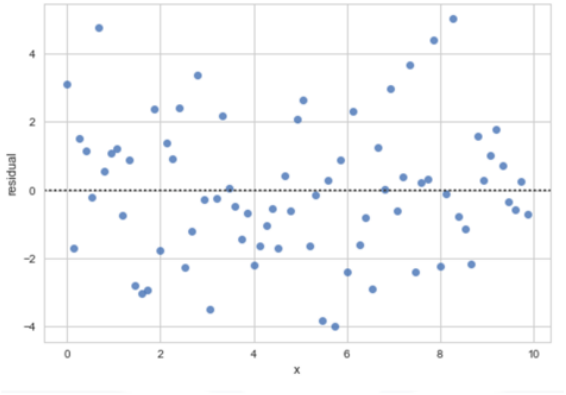


Seaborn is used to build such distribution plots

Distribution Plots

```
import seaborn as sns

ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")

sns.distplot(Yhat, hist=False, color="b", label="Fitted Values" , ax=ax1)
```

1. Consider the following **Residual Plot**, is our linear model correct : 1 / 1 point

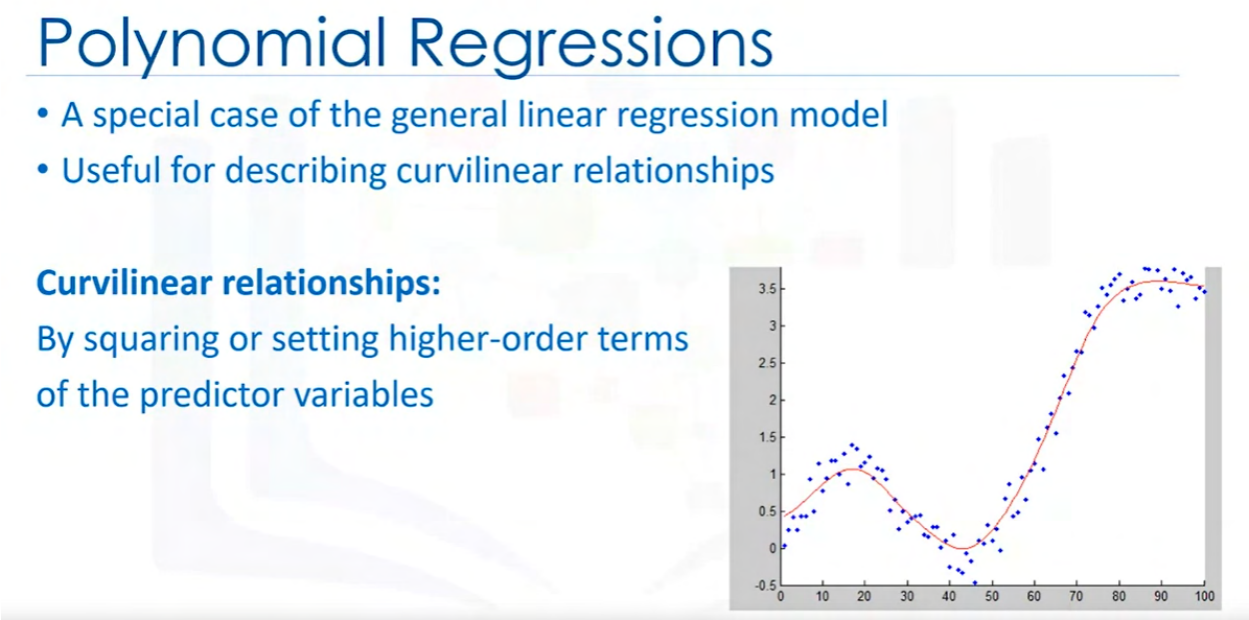


- (●) yes
- ( ) incorrect

Correct
correct

What do we do if a linear model is not the best fit for the data?
Let's go with …
Polynomial regression and pipelines

Polynomial regression and pipelines



As we have seen the plot, between the actual values and predicted values.

The predicted values are too far from the linear equation (model) we have built, which says that it is not an appropriate model for predictions.
Let's move to polynomial regression..

Polynomial Regression

- Quadratic – 2nd order

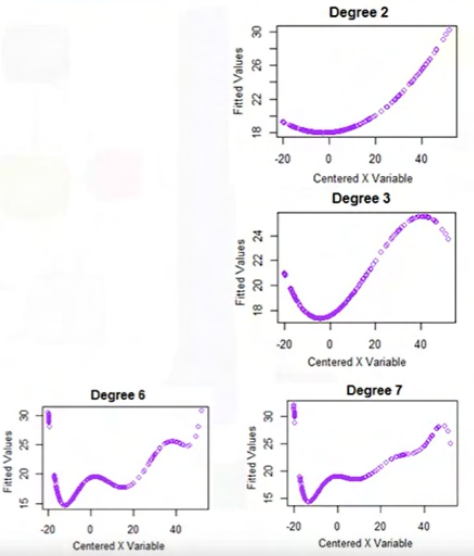
$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2$$

- Cubic – 3rd order

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3$$

- Higher order

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3 + ..$$



Polynomial Regression

1. Calculate Polynomial of 3rd order

```
f=np.polyfit(x,y,3)

p=np.poly1d(f)
```

2. We can print out the model

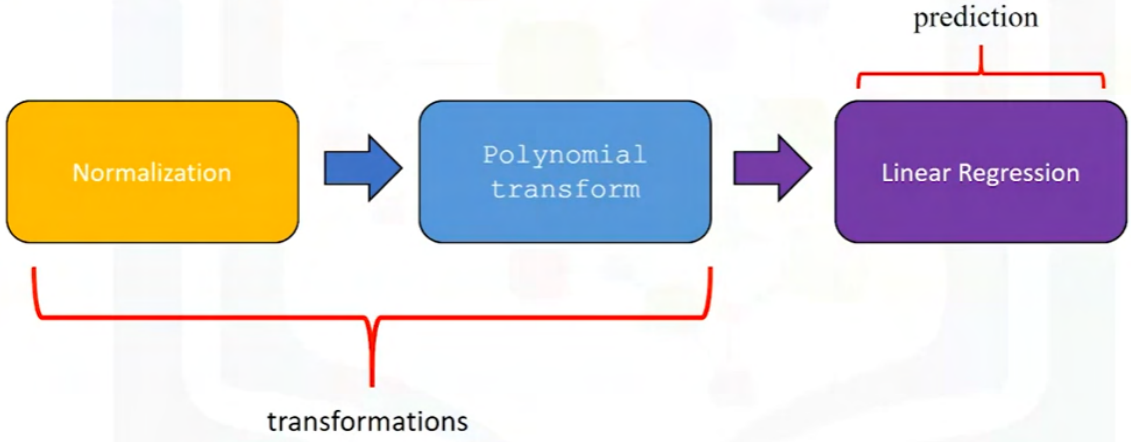
```
print (p)
```

Polynomial Regression

- We can also have multi dimensional polynomial linear regression

Pipelines

- There are many steps to getting a prediction



Pipeline Constructor

```
Input=[('scale',StandardScaler()),('polynomial',PolynomialFeatures(degree=2),…
('mode',LinearRegression())]
```

- Pipeline constructor

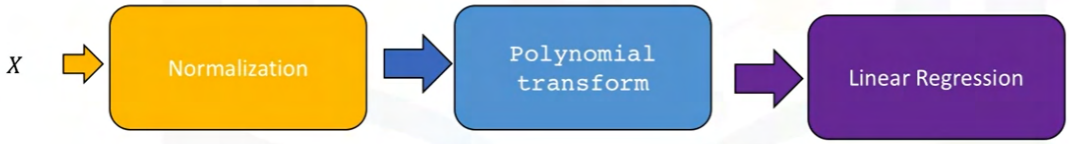
```
pipe=Pipeline(Input)
```

pipeline object

Pipeline Constructor

- We can train the pipeline object

```
Pipe.fit(df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y)

yhat=Pipe.predict(X[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
```



1. What functions are used to generate Polynomial Regression with more than one dimension

1 / 1 point

```
f=np.polyfit(x,y,3)
p=np.poly1d(f)
```

```
pr=PolynomialFeatures(degree=2)
pr.fit_transform([1,2], include_bias=False)
```

Measures for sample evaluation

Measures for In-Sample Evaluation

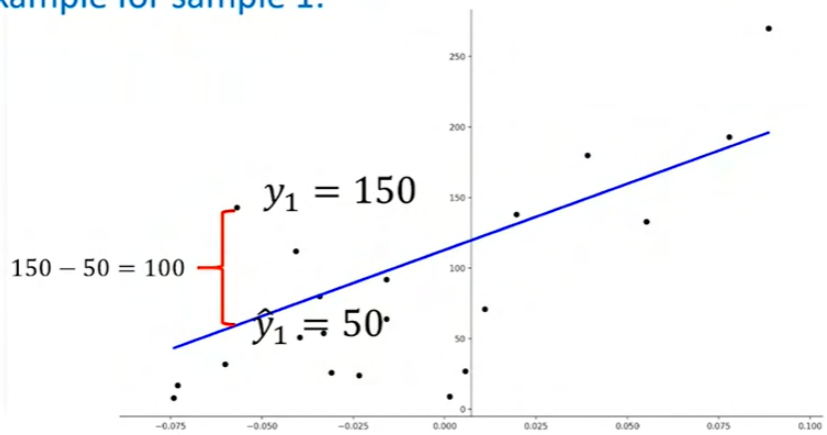
- A way to numerically determine how good the model fits on dataset.

- Two important measures to determine the fit of a model:
  - **Mean Squared Error (MSE)**
  - **R-squared (R^2)**

MSE

Mean Squared Error (MSE)

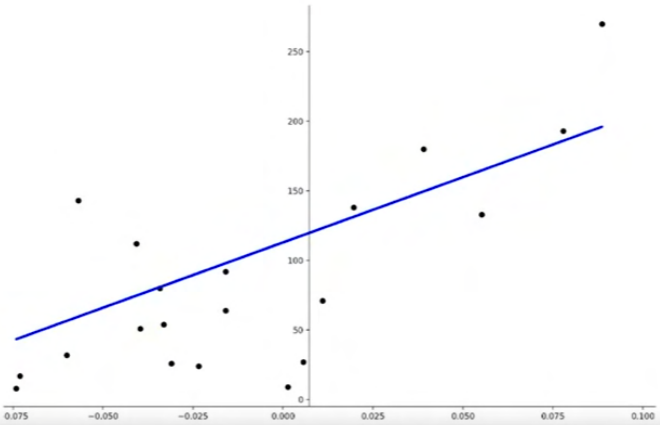
- For Example for sample 1:

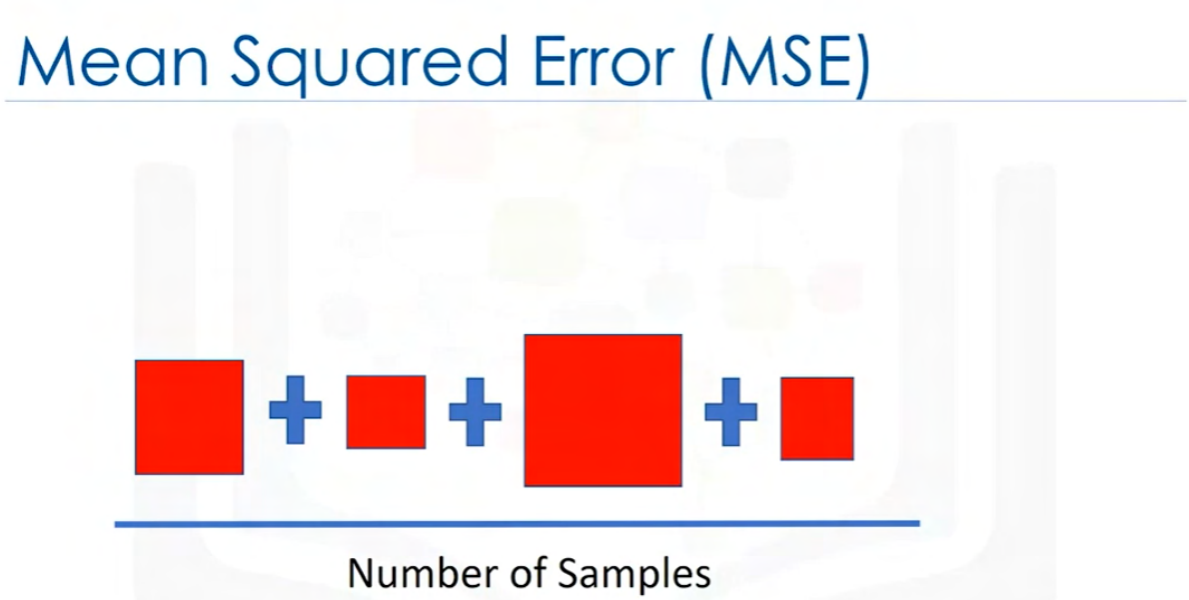


Mean Squared Error (MSE)

- To make all the values positive we square it

$(100)^2$





How would you calculate the mean squared error between your predicted values **Yhat** and actual values **Y**?

```
from  sklearn.metrixs import mean_squared_error

mean_squared_error(Y,Yhat)
```

```
X = df[['highway-mpg']]
Y = df['price']
lm.fit(X, Y)
lm.score(Yhat,Y)

```

Correct
correct

R-Squared value- R^2 value - coefficient of determination
Determines how close the data is to the fitted regression line

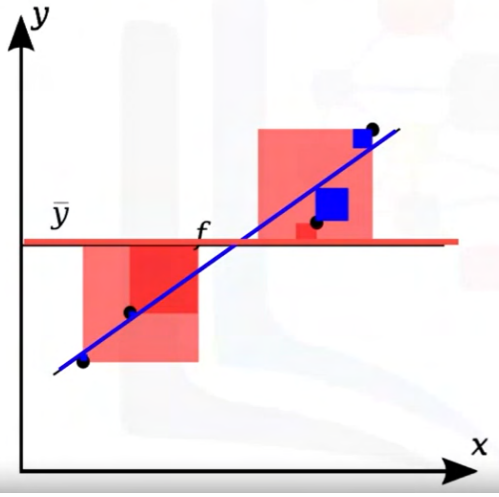
R-squared/ R^2

- The Coefficient of Determination or R squared (R^2)
- Is a measure to determine how close the data is to the fitted regression line.
- R^2: the percentage of variation of the target variable (Y) that is explained by the linear model.

Coefficient of Determination (R^2)

$$R^2 = \left(1 - \frac{\text{MSE of regression line}}{\text{MSE of the average of the data}}\right)$$

Coefficient of Determination (R^2)



- The blue line represents the regression line
- The blue squares represents the MSE of the regression line
- The red line represents the average value of the data points
- The red squares represent the MSE of the red line
- We see the area of the blue squares is much smaller than the area of the red squares

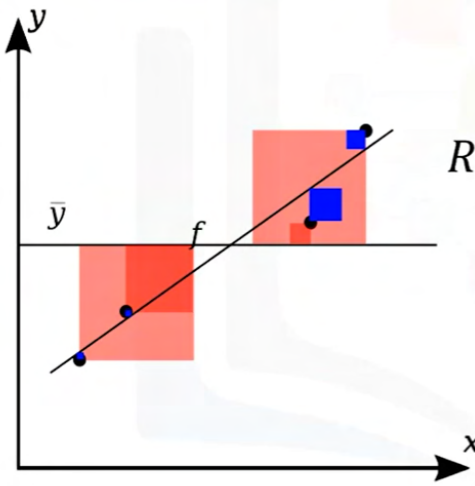
Coefficient of Determination (R^2)

- In this case ratio of the areas of MSE is close to zero

$$\frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}} = \frac{\square + \square}{\square + \square}$$

$$= 0$$

Coefficient of Determination (R^2)



$$R^2 = \left(1 - \frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}}\right)$$

$$= (1 - 0)$$

$$= (1 - 0)$$

$$= 1$$

R-squared/ R^2

- Generally the values of the MSE are between 0 and 1.
- We can calculate the R^2 as follows

```
X = df[['highway-mpg']]
Y = df['price']

lm.fit(X, Y)
```

```
lm.score(X,y)
0.496591188
```

1. Of the following answer values, which one is the minimum value of **R^2**?

- ○ 10
- ◉ 0
- ○ 1

✓ **Correct**
correct

Decision Making: Determining a Good Model Fit

To determine final best fit, we look at a combination of:

- Do the predicted values make sense
- Visualization
- Numerical measures for evaluation
- Comparing Models

Do the predicted values make sense

- First we train the model

```
lm.fit(df['highway-mpg'],df['prices'])
```

- Let's predict the price of a car with 30 highway-mpg.

```
lm.predict(np.array(30.0).reshape(-1,1))
```

- Result: $ 13771.30

Do the predicted values make sense

- First we train the model

```
lm.fit(df['highway-mpg'],df['prices'])
```

- Let's predict the price of a car with 30 highway-mpg.

```
lm.predict(np.array(30.0).reshape(-1,1))
```

- Result: $ 13771.30

```
lm.coef_
-821.73337832
```

- **Price** = 38423.31 - 821.73 * **highway-mpg**

Do the predicted values make sense

- First we import numpy

```
import numpy as np
```

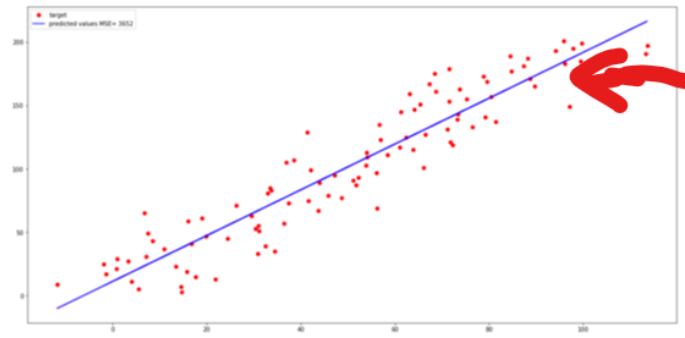
- We use the numpy function arrange to generate a sequence from 1 to 100

```
new_input=np.arange(1,101,1).reshape(-1,1)
```

1	2	...	99	100

Consider the plots a and b below; red is the sample data and the blue line is the predicted value. Which plot has a higher means square error?

a



not close

b

close

Comparing MLR and SLR

Does a lower Mean Square Error imply better fit?

- Not necessarily

1. Mean Square Error for a Multiple Linear Regression Model will be smaller than the Mean Square Error for a Simple Linear Regression model, since the errors of the data will decrease when more variables are included in the model
2. Polynomial regression will also have a smaller Mean Square Error than the linear regular regression

Define the explanatory variable and the response variable: Define the response variable (y) as the focus of the experiment and the explanatory variable (x) as a variable used to explain the change of the response variable. Understand the differences between Simple Linear Regression because it concerns the study of only one explanatory variable and Multiple Linear Regression because it concerns the study of two or more explanatory variables.
Evaluate the model using Visualization: By visually representing the errors of a variable using scatterplots and interpreting the results of the model.
Identify alternative regression approaches: Use a Polynomial Regression when the Linear regression does not capture the curvilinear relationship between variables and how to pick the optimal order to use in a model.
Interpret the R-square and the Mean Square Error: Interpret R-square (x 100) as the percentage of the variation in the response variable *y* that is explained by the variation in explanatory variable(s) *x.* The Mean Squared Error tells you how close a regression line is to a set of points. It does this by taking the average distances from the actual points to the predicted points and squaring them.

Graded Quiz -4: Model Development

1. What does the following line of code do? **1 point**

```
1  lm = LinearRegression()
```

- ○ Fit a regression object lm
- ◉ Create a linear regression object
- ○ Predict a value

2. What is the maximum value of **R^2** that can be obtained? **1 point**

- ○ 10
- ○ 0
- ◉ 1

3. We create a polynomial feature as follows "**PolynomialFeatures(degree=2)**"; what is the order of the polynomial? **1 point**

- ○ 0
- ○ 1
- ◉ 2

4. What value of **R^2** (coefficient of determination) indicates your model performs best? **1 point**

- ○ -1
- ◉ 1
- ○ 0

5. Consider the following equation: 1 point

$$y = b_0 + b_1 x$$

The variable **y** is what?

- [x] The target or dependent variable
- [ ] The predictor or independent variable
- [ ] The intercept

Week -5

Model Evaluation and Refinement

Model Evaluation

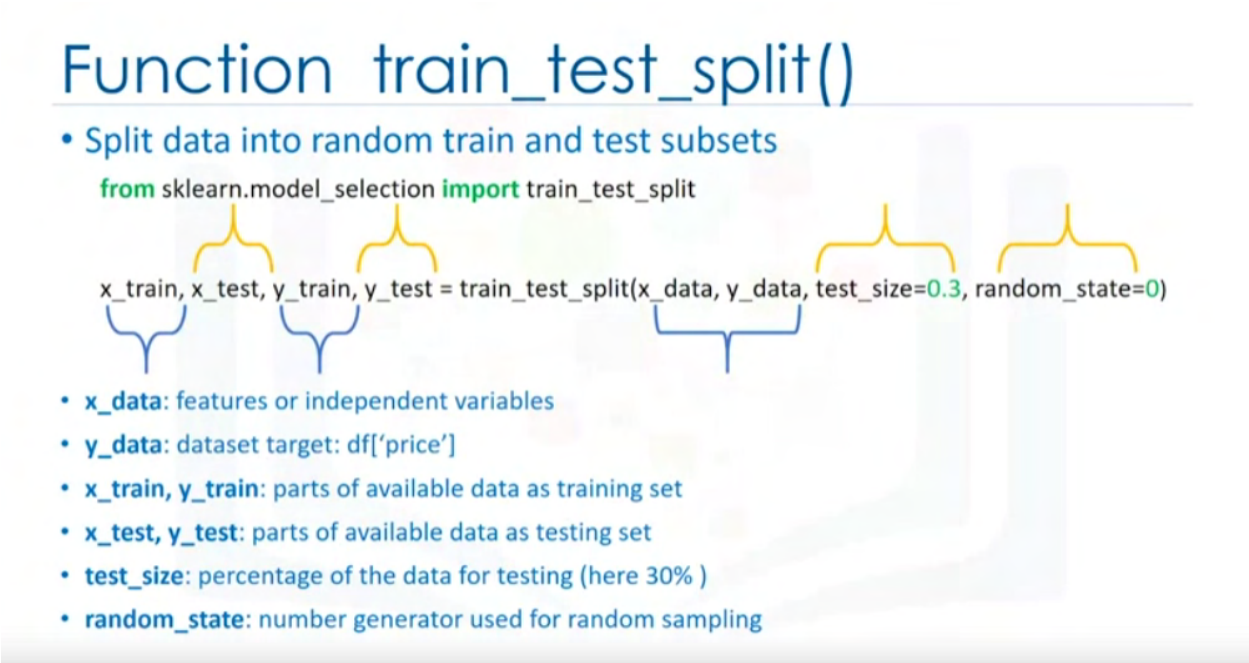
- In-sample evaluation tells us how well our model will fit the data used to train it
- Problem?
  - It does not tell us how well the trained model can be used to predict new data
- Solution?
  - In- sample data or training data
  - Out-of-sample evaluation or test set

Training/Testing Sets

Data:

- Split dataset into:
  - Training set (70%),
  - Testing set (30%)

- Build and train the model with a training set
- Use testing set to assess the performance of a predictive model
- When we have completed testing our model we should use all the data to train the model to get the best performance



Question

consider the following lines of code:

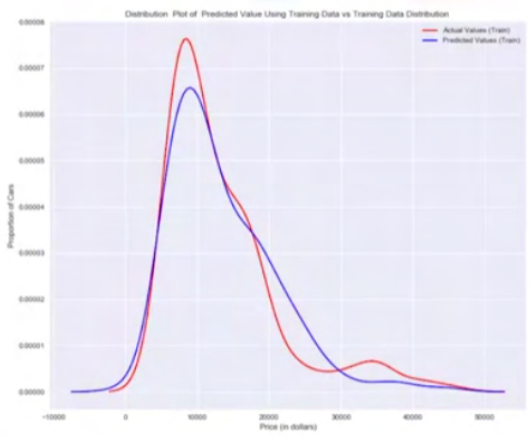
```
from sklearn.model_selection import train_test_split


x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=0)


```

what does the variable **y_data** contain

- ○ percentage of the data for testing
- ◉ dataset target
- ○ features or independent variables

✓ **Correct**
correct

Generalization Error

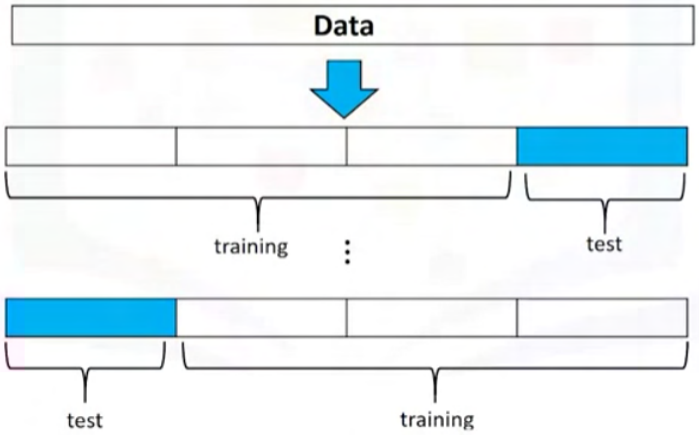


Lots of Training Data

Model

Cross Validation

- Most common out-of-sample evaluation metrics
- More effective use of data (each observation is used for both training and testing)



Function cross_val_score()

```
from sklearn.model_selection import cross_val_score

scores= cross_val_score(lr, x_data, y_data, cv=3)

np.mean(scores)
```

Question

consider the following lines of code, how may partitions or folds are used in the function **cross_val_score**:

```
from sklearn.model_selection import cross_val_sc
scores= cross_val_score(lr, x_data, y_data, cv=10)

```

- 4
- 10
- 5

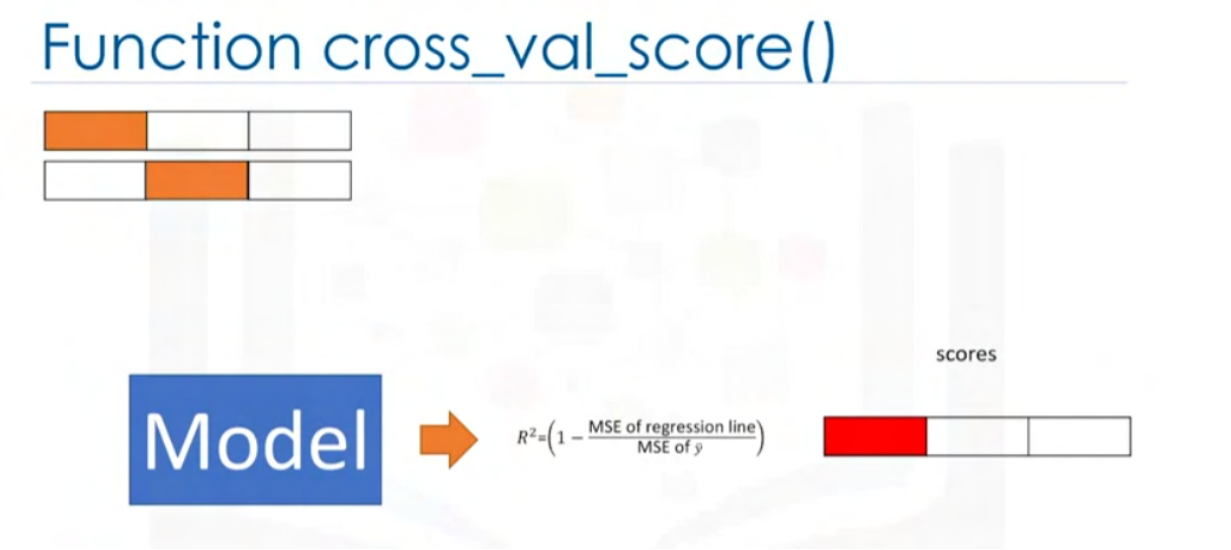
Correct
correct

Skip Continue

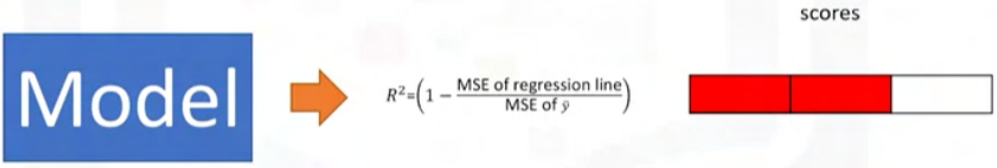
- 4
- 10
- 5

Correct
correct

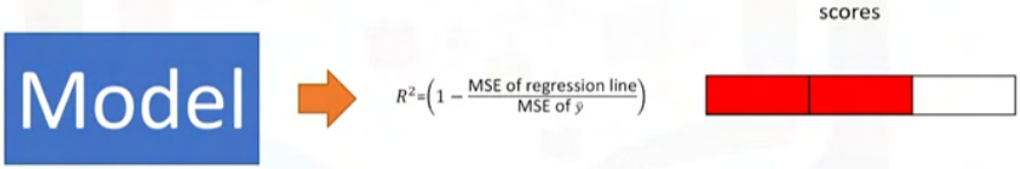
Skip Continue



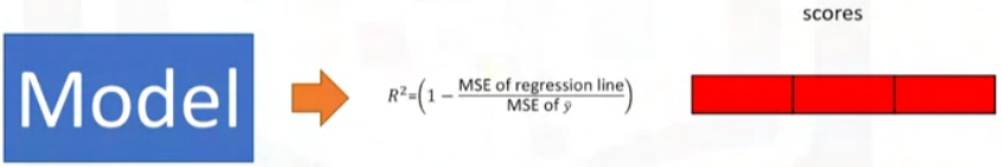
Function cross_val_score()



Function cross_val_score()



Function cross_val_score()

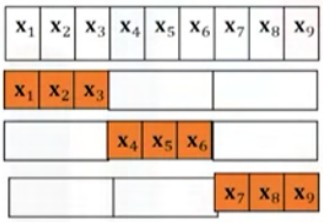


Function cross_val_predict()

- It returns the prediction that was obtained for each element when it was in the test set
- Has a similar interface to cross_val_score()

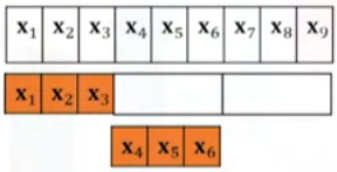
```
from sklearn.model_selection import cross_val_predict

yhat= cross_val_predict (lr2e, x_data, y_data, cv=3)
```

Function cross_val_predict()

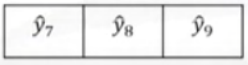


Model

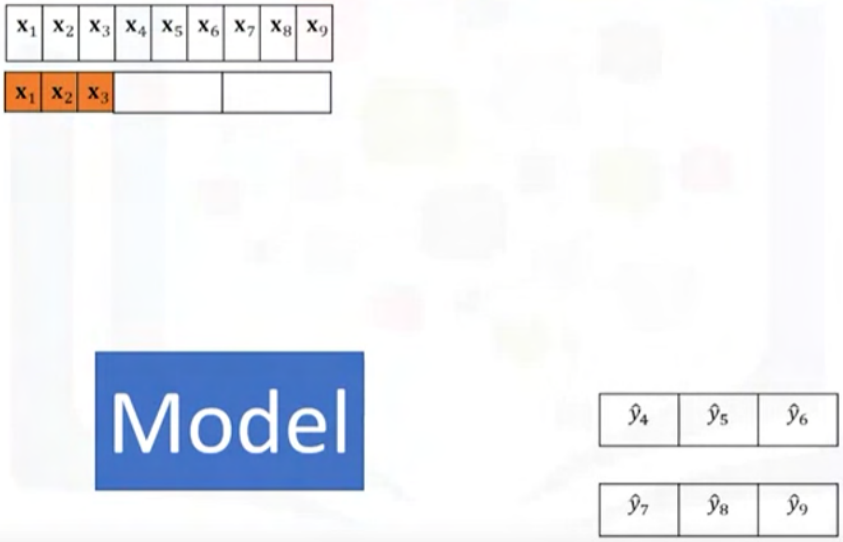
Function cross_val_predict()



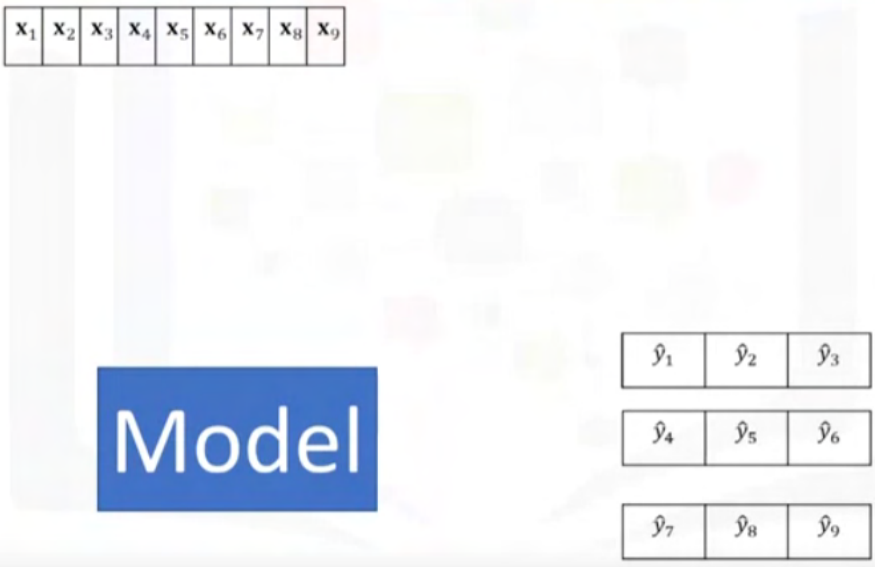
Model



Function cross_val_predict()



Function cross_val_predict()



1. What is the correct use of the "train_test_split" function such that 90% of the data samples will be utilized for training, the parameter "random_state" is set to zero, and the input variables for the features and targets are x_data, y_data respectively. 1 / 1 point

```
train_test_split(x_data, y_data, test_size=0.9, random_state=0)
```

```
train_test_split(x_data, y_data, test_size=0.1, random_state=0)
```

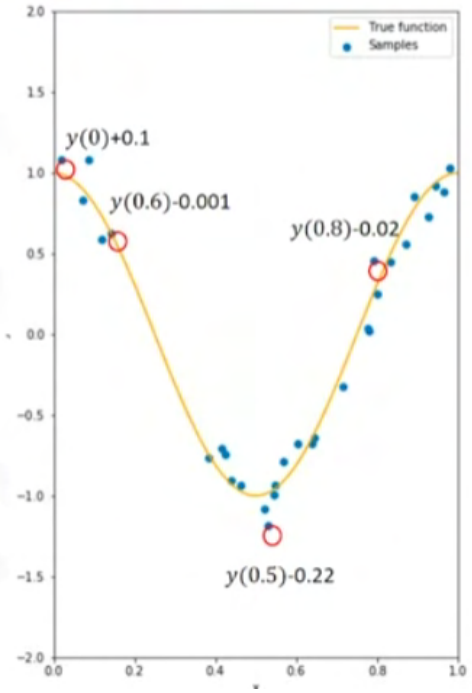
Correct

Overfitting, Underfitting and Model Selection

The goal of Model Selection is to determine the order of the polynomial to provide the best estimate of the function y(x). If we try and fit the function with a linear function, the line is not complex enough to fit the data. As a result, there are many errors. This is called underfitting, where the model is too simple to fit the data. If we increase the order of the polynomial, the model fits better, but the model is still not flexible enough and exhibits underfitting. This is an example of the 8th order polynomial used to fit the data.

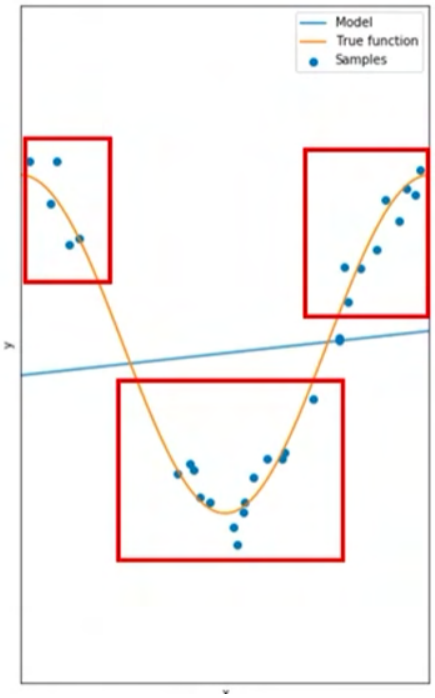
Model Selection

$y(x)$+noise



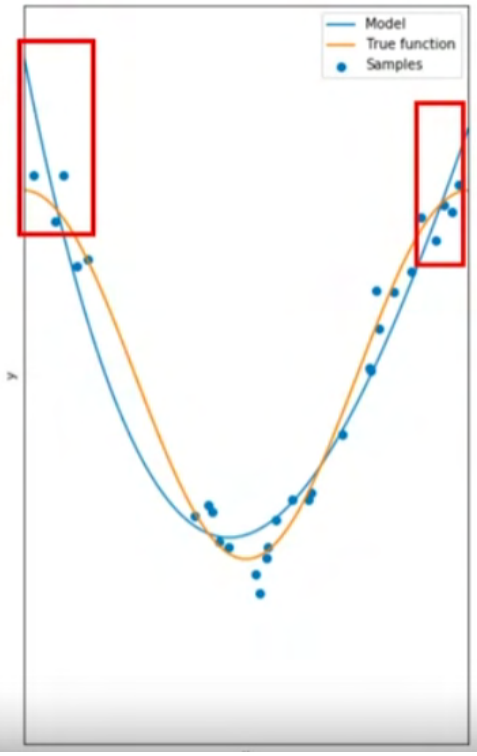
If we increase the order of the polynomial, the model fits better, but the model is still not flexible enough and exhibits underfitting.

$$y = b_0 + b_1 x$$



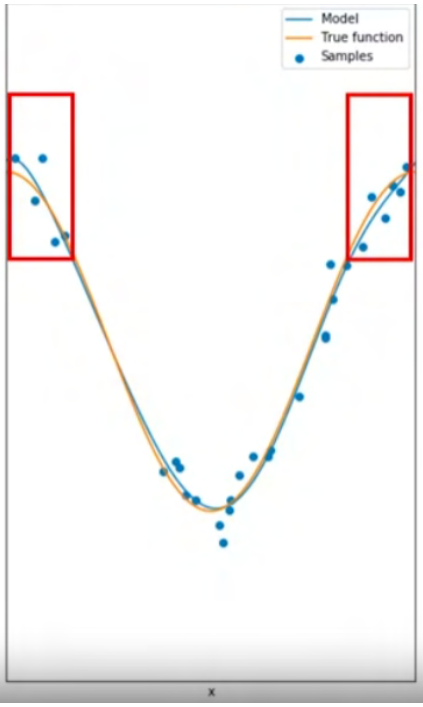
The model too simple to fit the data - this is called under fitting (simple linear equation is not always the solution)

$$y = b_0 + b_1 x + b_2 x^2$$



This is an example of the 8th order polynomial used to fit the data.

$$\hat{y} = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5 + b_6 x^6 + b_7 x^7 + b_8 x^8$$

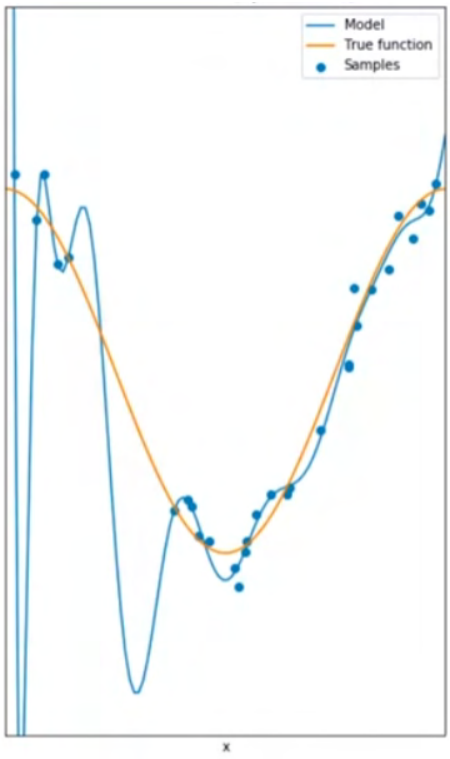


We see the model does well at fitting the data and estimating the function even at the inflection points.

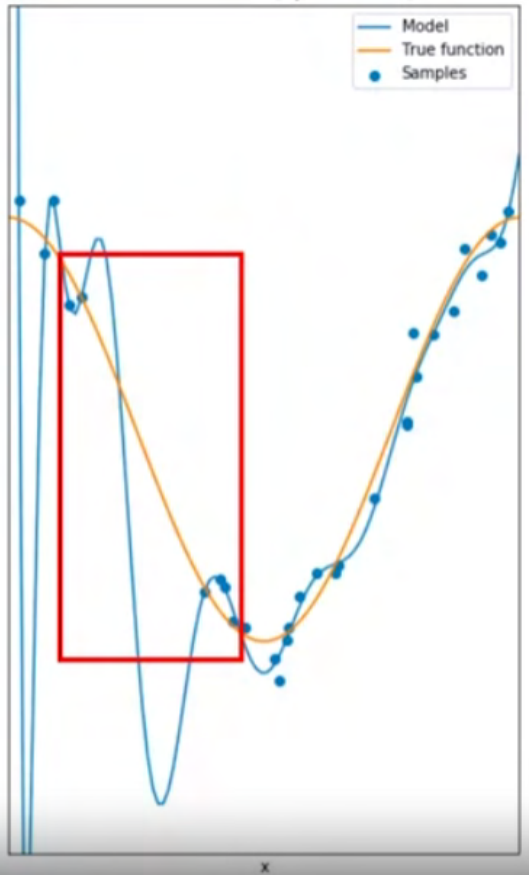
Increasing it to a 16th order polynomial, the model does extremely well at tracking the training point but performs poorly at estimating the function. This is especially apparent where there is little training data. The estimated function oscillates without tracking the function. This is called **overfitting**, where the model is too flexible and fits the noise rather than the function.

where the model is too flexible and fits the noise rather than the function.

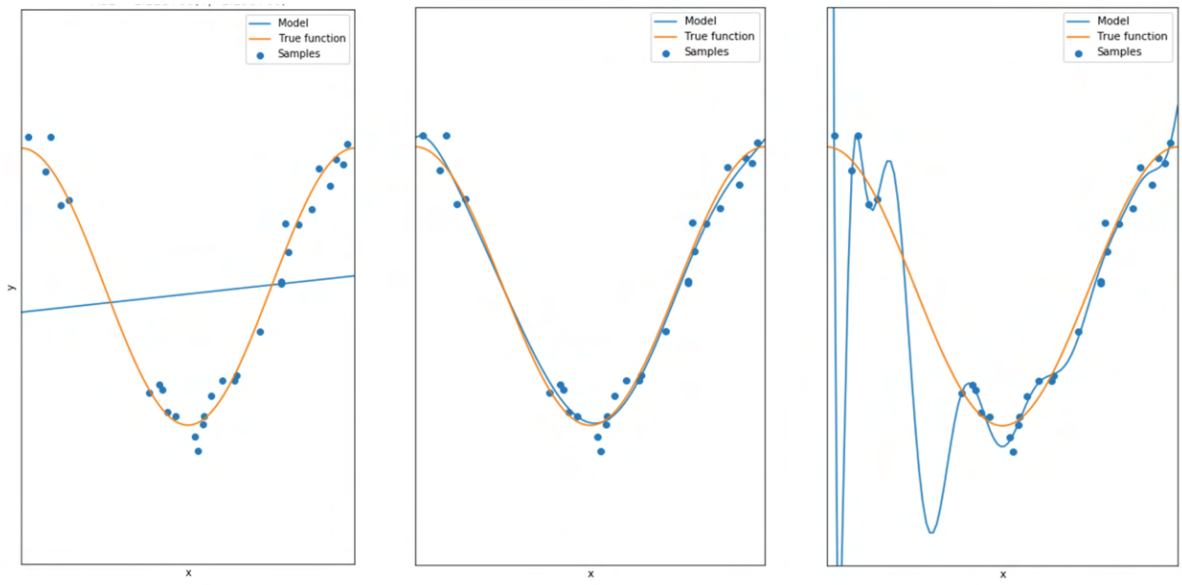
$$\hat{y} = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5 + b_6 x^6 + b_7 x^7 + b_8 x^8 + ..$$
$$+ b_9 x^9 + b_{10} x^{10} + b_{11} x^{11} + b_{12} x^{12} + b_{13} x^{13} + b_{14} x^{14} + b_{15} x^{15} + b_{16} x^{16}$$



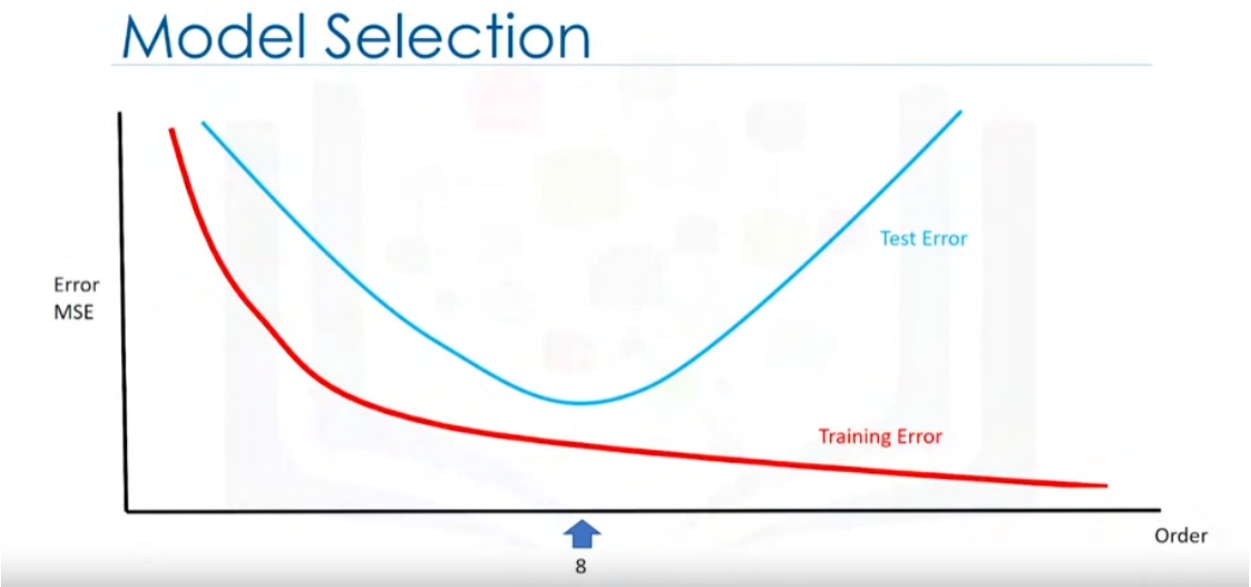
$$\hat{y} = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5 + b_6 x^6 + b_7 x^7 + b_8 x^8 + ..$$
$$+ b_9 x^9 + b_{10} x^{10} + b_{11} x^{11} + b_{12} x^{12} + b_{13} x^{13} + b_{14} x^{14} + b_{15} x^{15} + b_{16} x^{16}$$

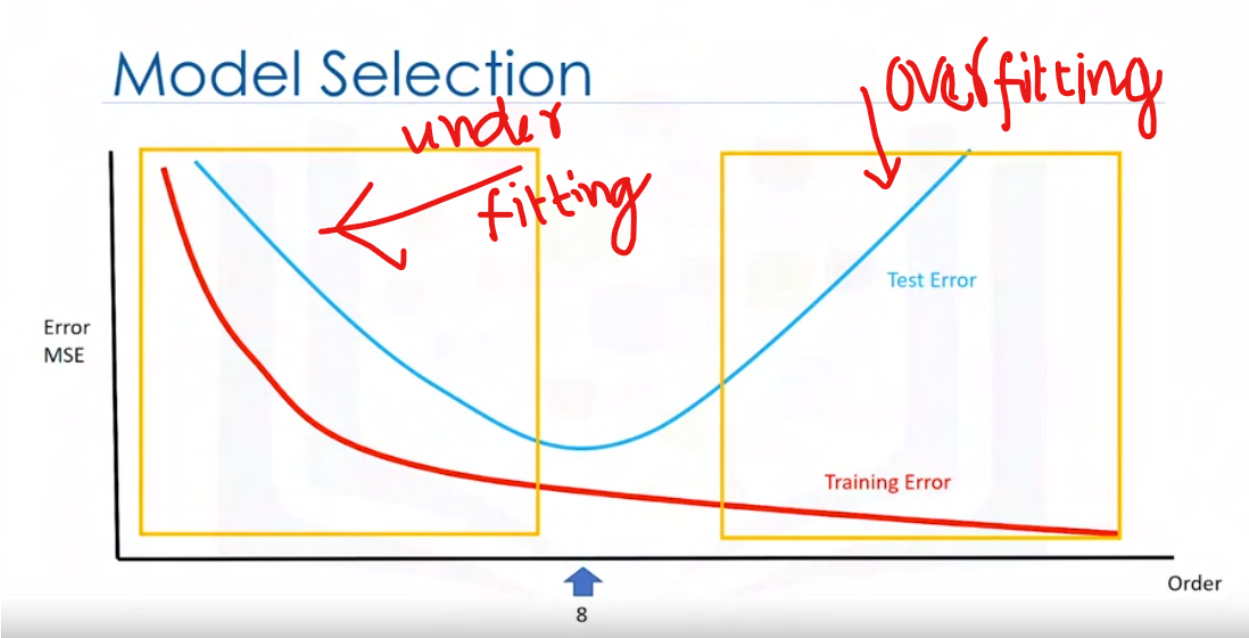


select the plot a, b or c that best demonstrates **overfitting**:

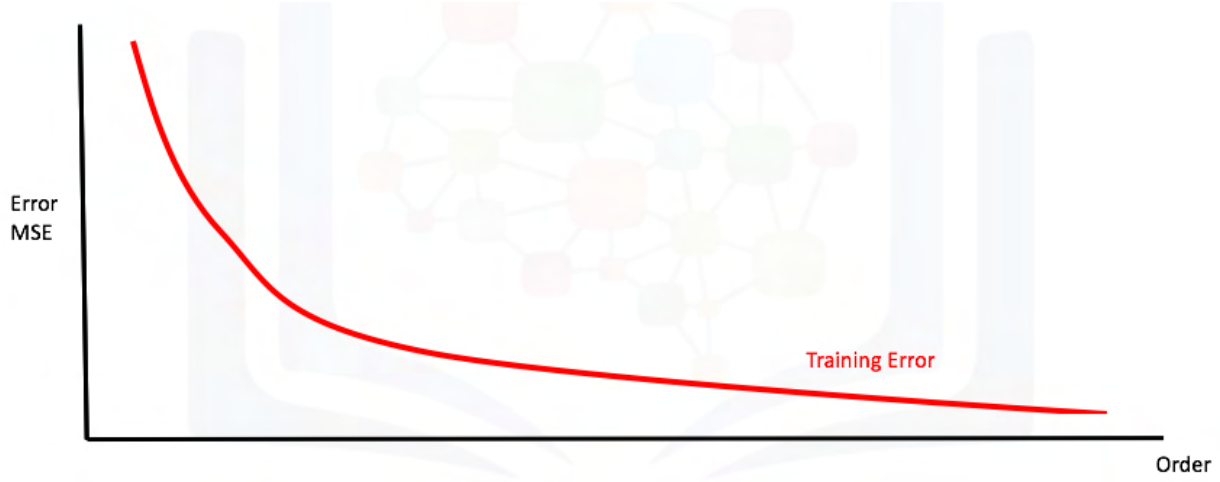


C

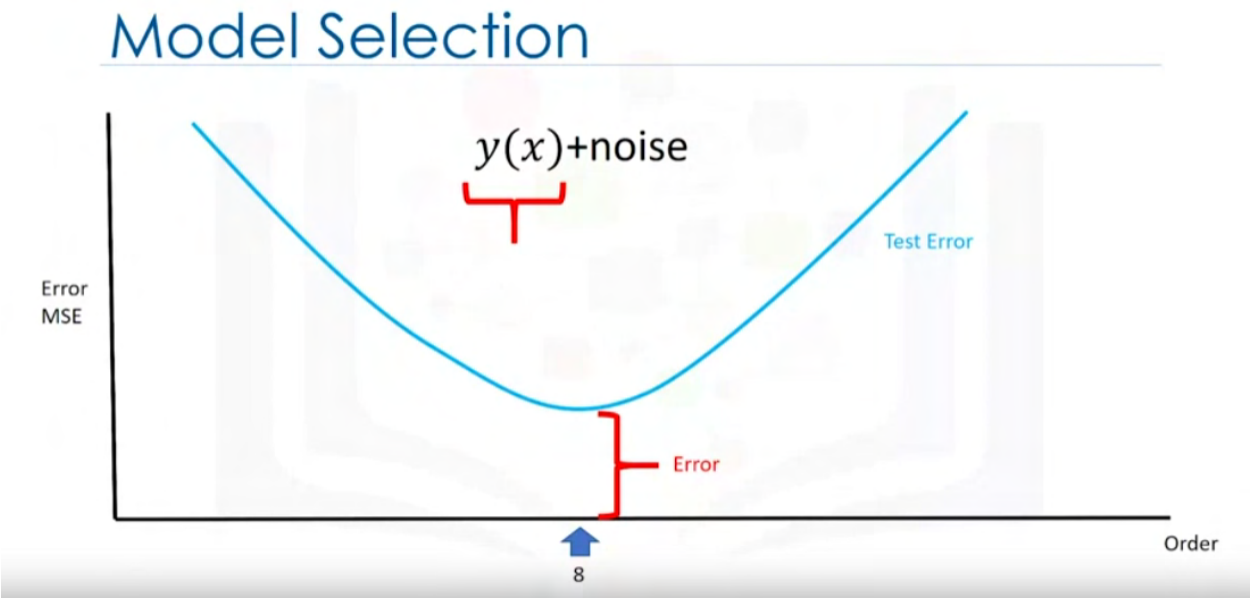


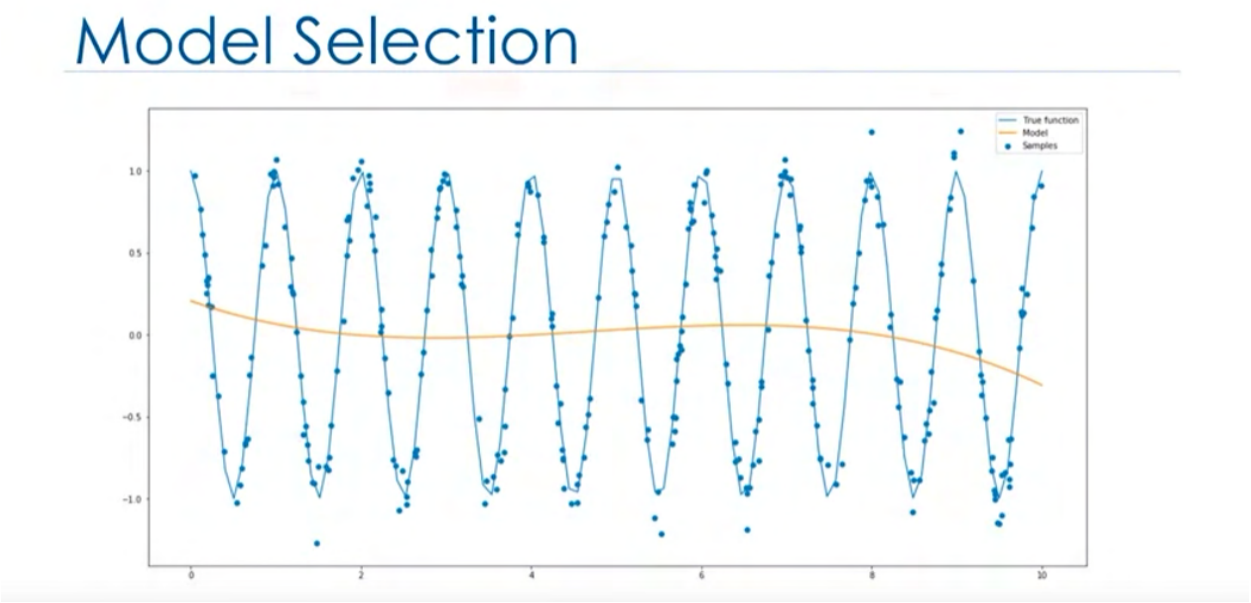


True or False, the following plot shows that as the order of the polynomial increases the mean square error of our model decreases on the test data:

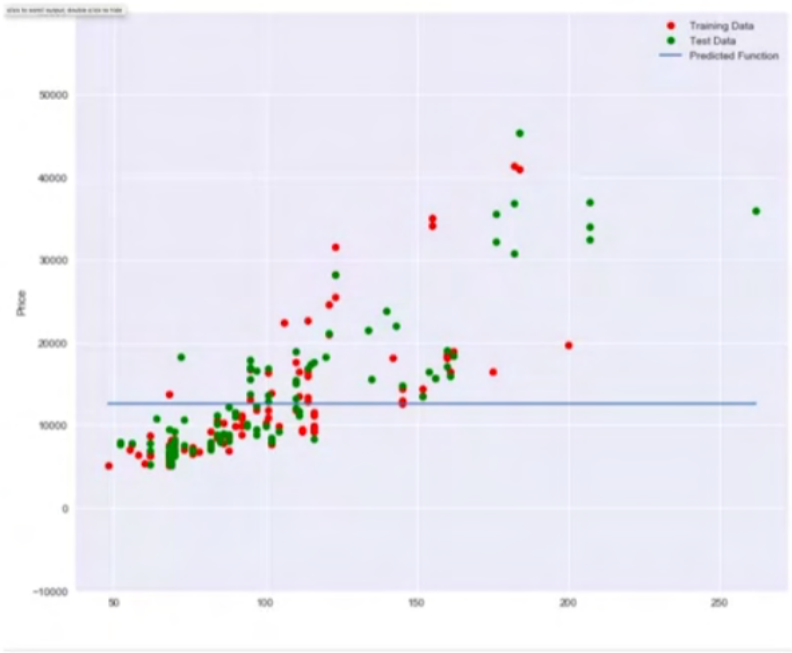


False





Our sample points may have come from a different function. For example, in this plot, the data is generated from a sine wave. The polynomial function does not do a good job at fitting the sine wave. For real data, the model may be too difficult to fit or we may not have the correct type of data to estimate the function.

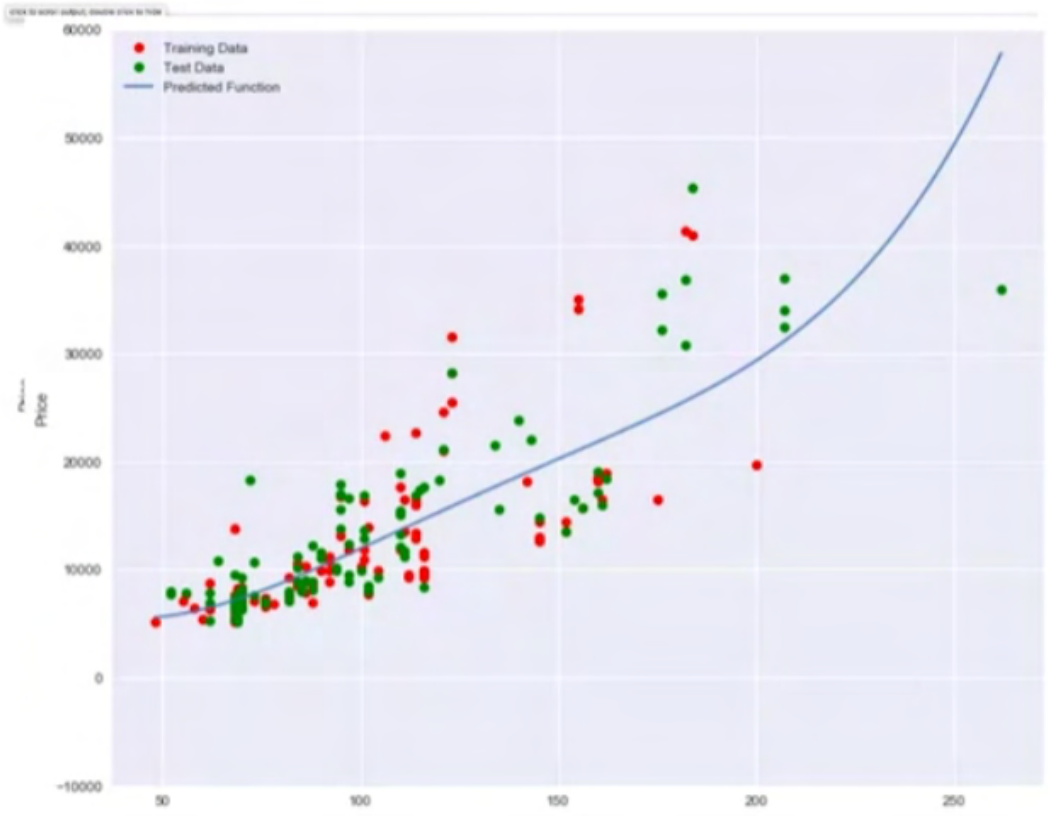


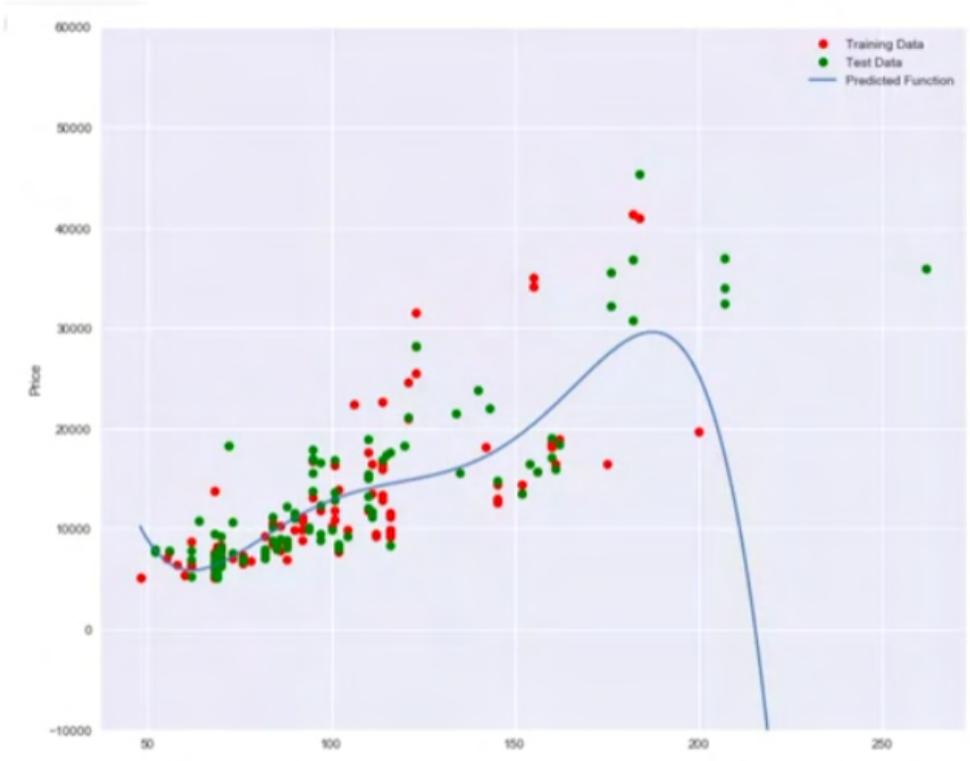
Let's try different order polynomials on the real data using horsepower. The red points represent the training data. The green points represent the test data. If we just use the mean of the data, our model does not perform well. A linear function does fit the data better.

Let’s see different order polynomials to fit the data

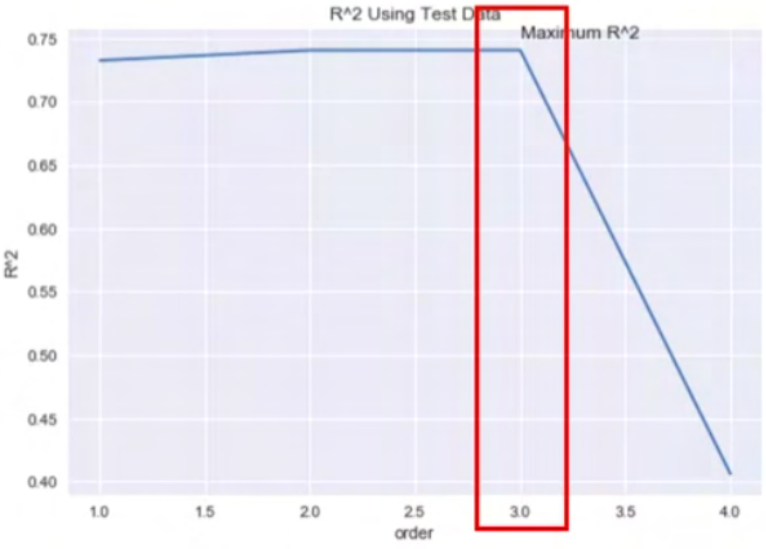

Training Data
Test Data
Predicted Function
Price
50000
40000
30000
20000
10000
0
−10000
50
100
150
200
250


Training Data
Test Data
Predicted Function
Price
60000
50000
40000
30000
20000
10000
0
−10000
50
100
150
200
250


Training Data
Test Data
Predicted Function
Price
60000
50000
40000
30000
20000
10000
0
−10000
50
100
150
200
250


Training Data
Test Data
Predicted Function
Price
60000
50000
40000
30000
20000
10000
0
−10000
50
100
150
200
250

Model Selection

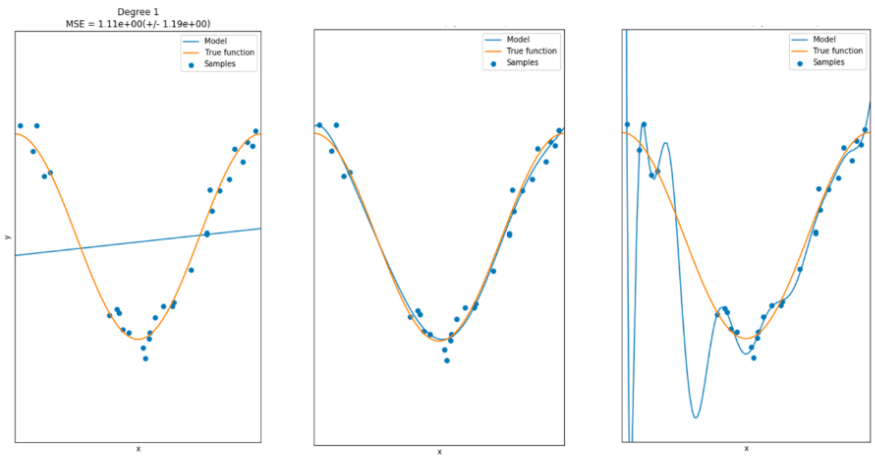


Let's find the r-squared value for different order polynomials

```
Rsqu_test=[]

order=[1,2,3,4]

for n in order:

    pr=PolynomialFeatures(degree=n)

    x_train_pr=pr.fit_transform(x_train[['horsepower']])

    x_test_pr=pr.fit_transform(x_test[['horsepower']])

    lr.fit(x_train_pr,y_train)

    Rsqu_test.append(lr.score(x_test_pr,y_test))
```

1. what model should you select

1 / 1 point



a b c

- c
- a
- b (selected)

Correct
correct

Explanation:- A has underfitting
C has over fitting
So, B - best model

Ridge Regression Introduction

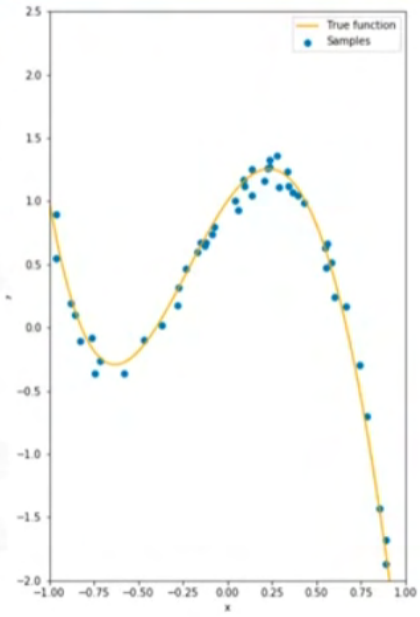
Ridge regression is a regression that is employed in a Multiple regression model when Multicollinearity occurs. Multicollinearity is when there is a strong relationship among the independent variables. Ridge regression is very common with polynomial regression. The next video shows how Ridge regression is used to regularize and reduce the standard errors to avoid overfitting a regression model

Ridge regression prevents overfitting. In this video, we will focus on polynomial regression for visualization, but overfitting is also a big problem when you have multiple independent variables, or features. Consider the following fourth order polynomial in orange.

The blue points are generated from this function. We can use a tenth order polynomial to fit the data.

Ridge Regression

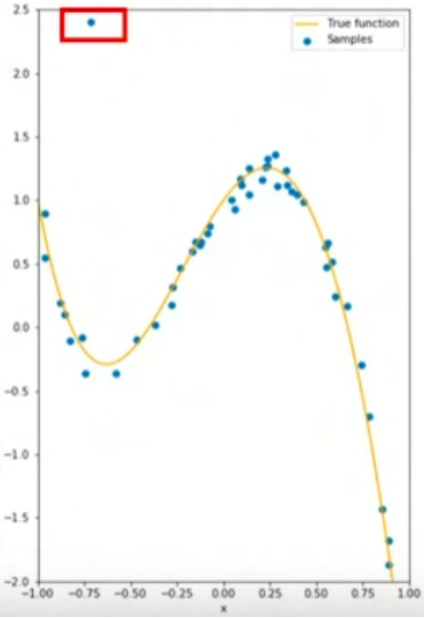
$$y = 1 + 2x - 3\,x^2 - 4x^3 + x^4$$



The estimated function in blue does a good job at approximating the true function. In many cases real data has outliers. For example, this point shown here does not appear to come from the function in orange.

Ridge Regression

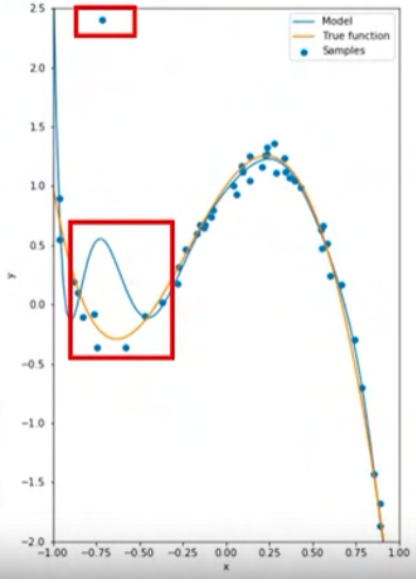
$1 + 2x - 3\,x^2 - 4x^3 + x^4$



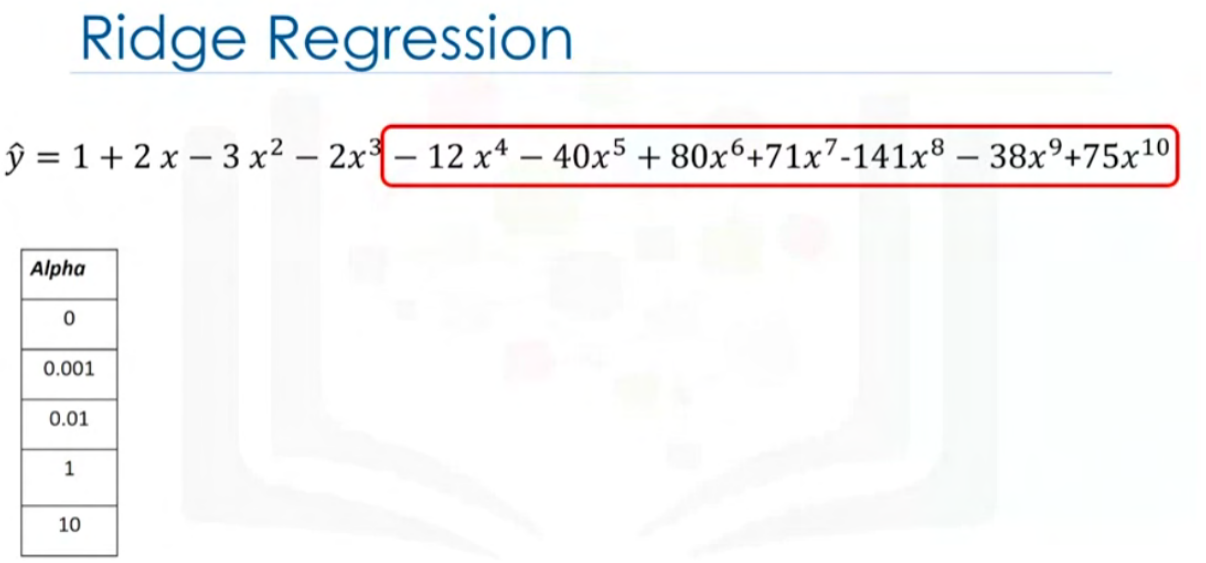
If we use a tenth order polynomial function to fit the data, the estimated function in blue is incorrect, and is not a good estimate of the actual function in orange.

Ridge Regression

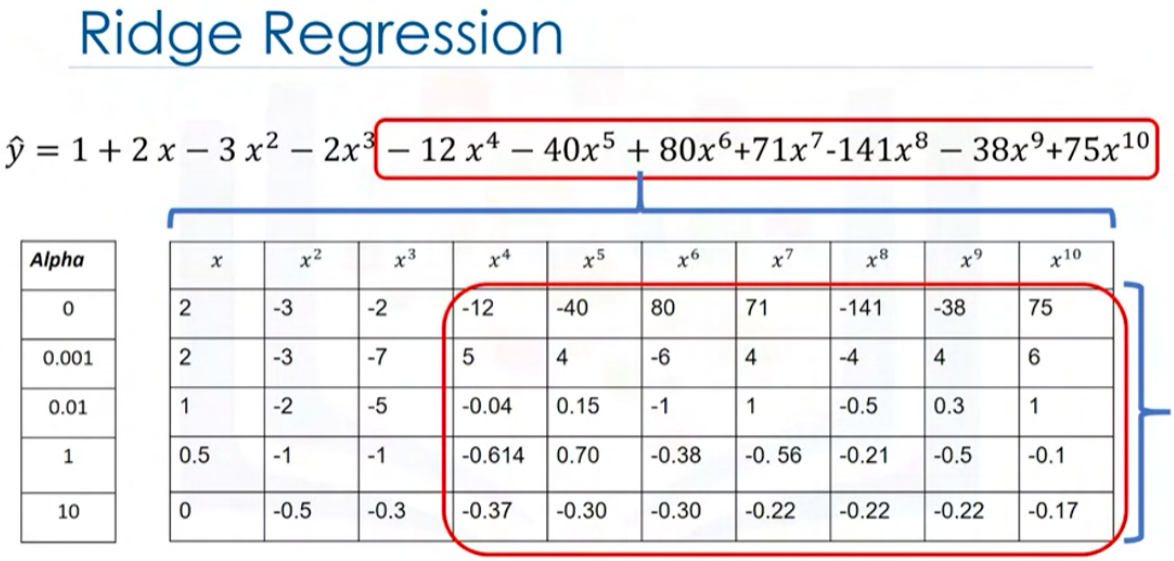
$1 + 2x - 3\,x^2 - 4x^3 + x^4$



If we examine the expression for the estimated function, we see the estimated polynomial coefficients have a very large magnitude. This is especially evident for the higher order polynomials.



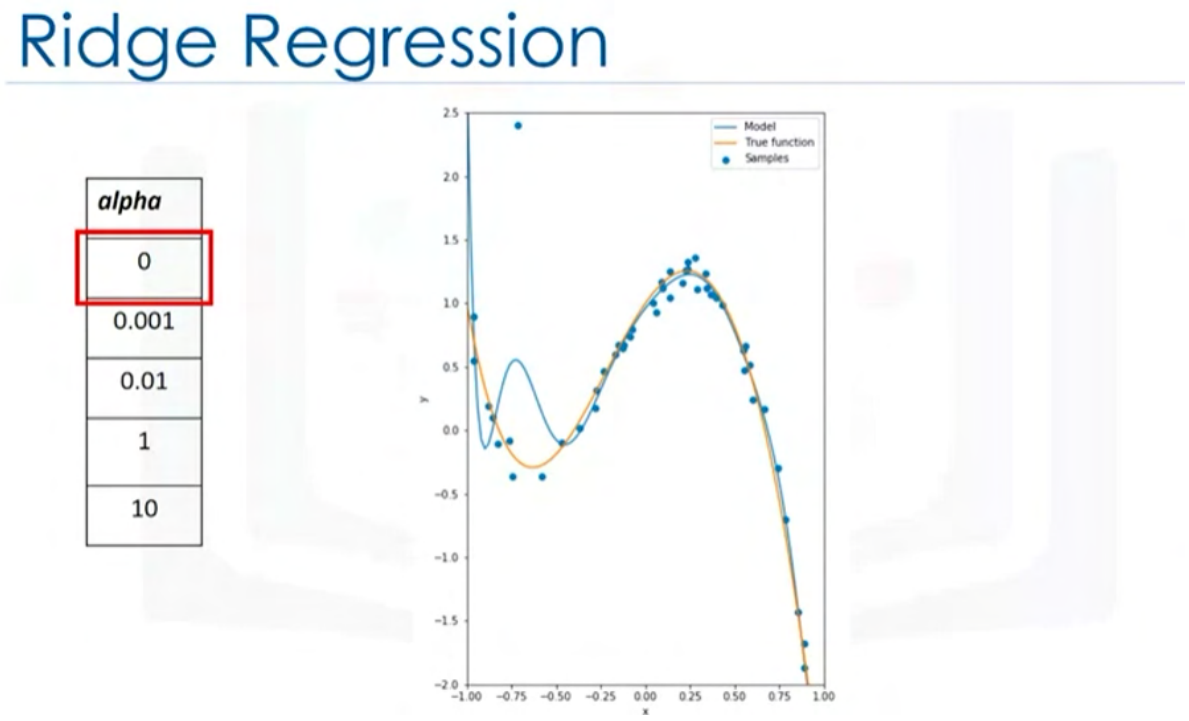
Ridge regression controls the magnitude of these polynomial coefficients by introducing the parameter alpha.



Alpha is a parameter we select before fitting or training the model. Each row in the following table represents an increasing value of alpha. Let's see how different values of alpha change the model. This table represents the polynomial coefficients for different values

of alpha. The column corresponds to the different polynomial coefficients, and the rows correspond to the different values of alpha. As alpha increases, the parameters get smaller.

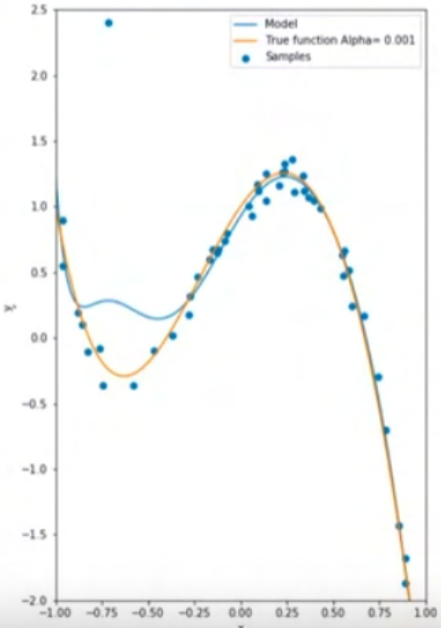
This is most evident for the higher order polynomial features. But Alpha must be selected carefully. If alpha is too large, the coefficients will approach zero and underfit the data.
If alpha is zero, the overfitting is evident.



For alpha equal to 0.001, the overfitting begins to subside.

Ridge Regression

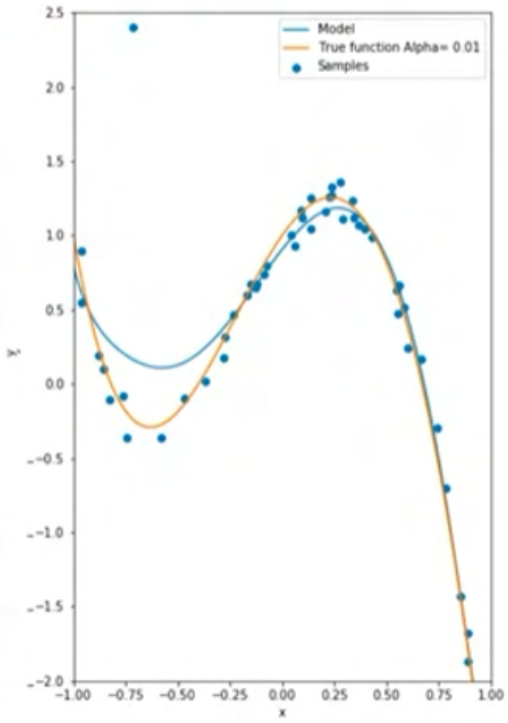
alpha
0
0.001
0.01
1
10



For Alpha equal to 0.01, the estimated function tracks the actual function.

Ridge Regression

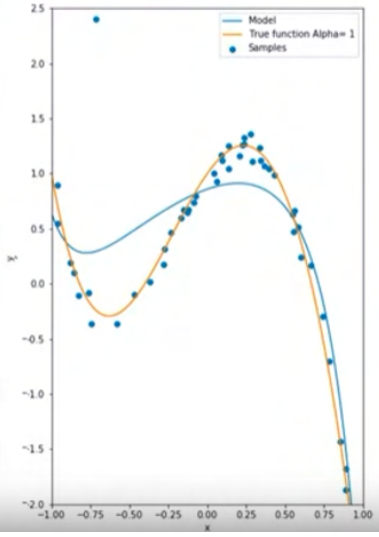
alpha
0
0.001
0.01
1
10



When alpha equals 1, we see the first signs of underfitting.

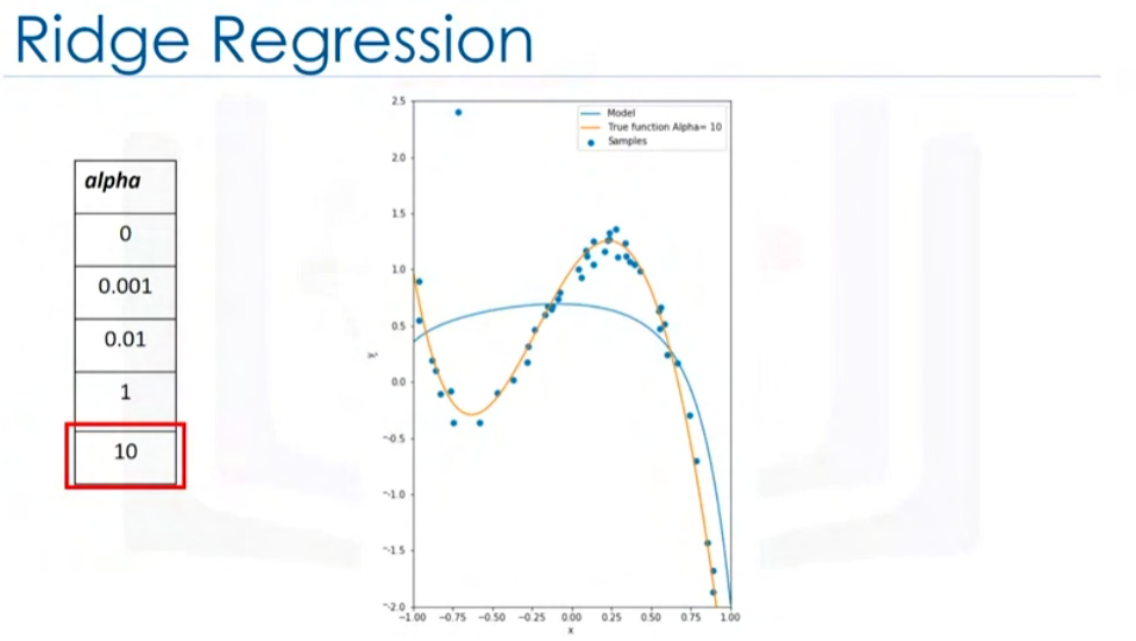
Ridge Regression

alpha
0
0.001
0.01
1
10

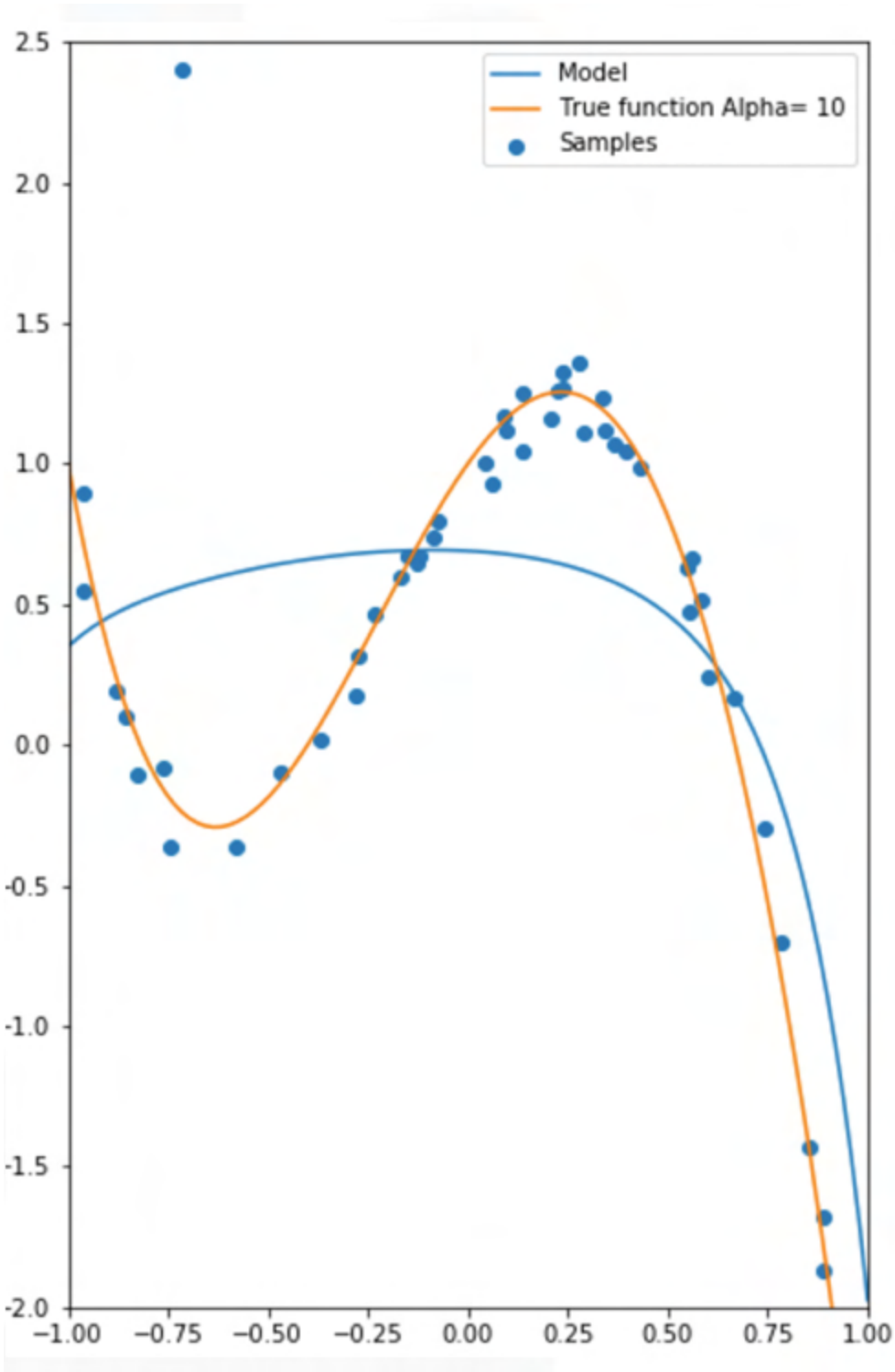


The estimated function does not have enough flexibility.

At alpha equals to 10, we see extreme underfitting. It does not even track the two points.



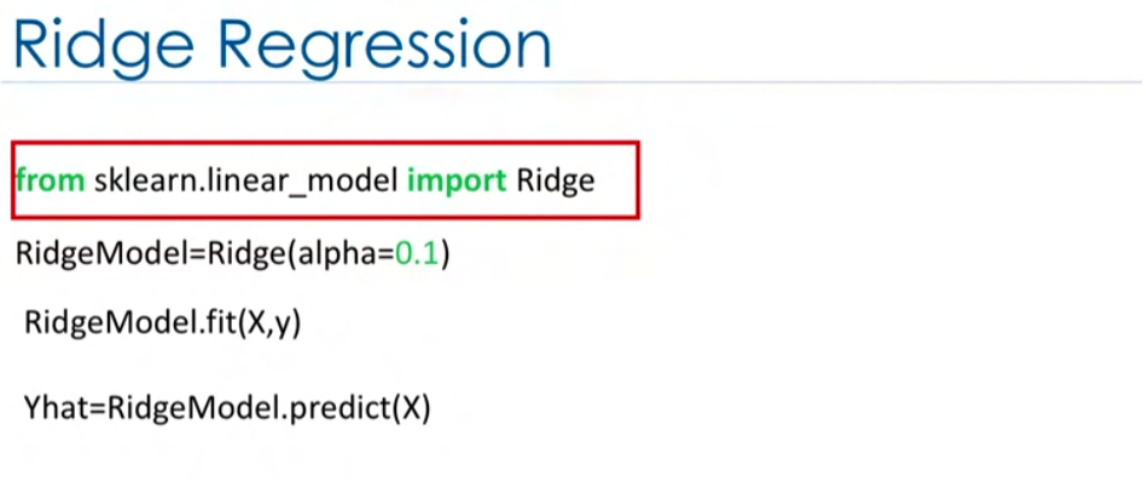
Consider the following fourth order polynomial, fitted with Ridge Regression; should we increase or decrease the parameter alpha?



Decrease is the answer

In order to select alpha, we use cross validation.

To make a prediction using ridge regression, import ridge from sklearn.linear_models. Create a ridge object using the constructor. The parameter alpha is one of the arguments of the constructor.



We train the model using the fit method.
To make a prediction, we use the predict method. In order to determine the parameter alpha, we use some data for training. We use a second set called validation data.

Ridge Regression

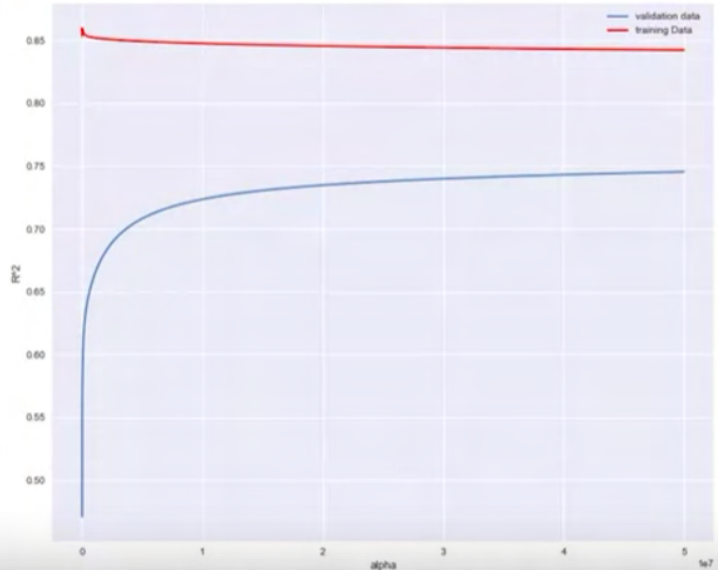
Ridge Regression

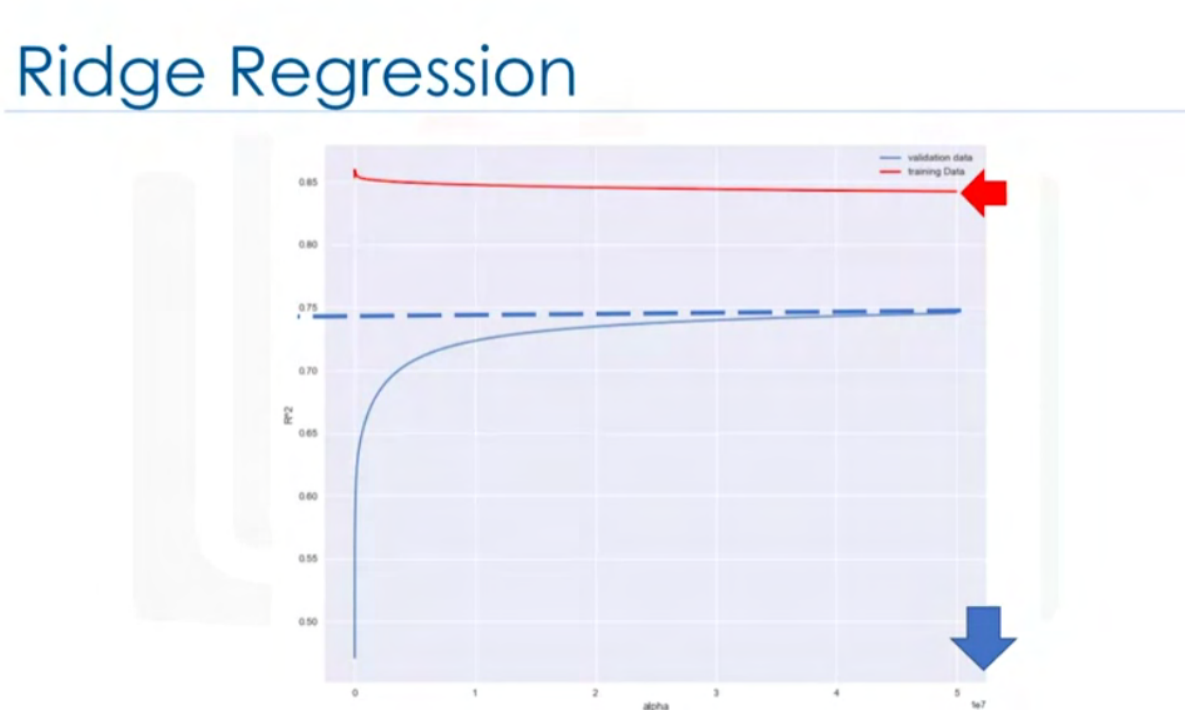
Ridge Regression

alpha

R^2
0.5
0.75
0.55

Ridge Regression



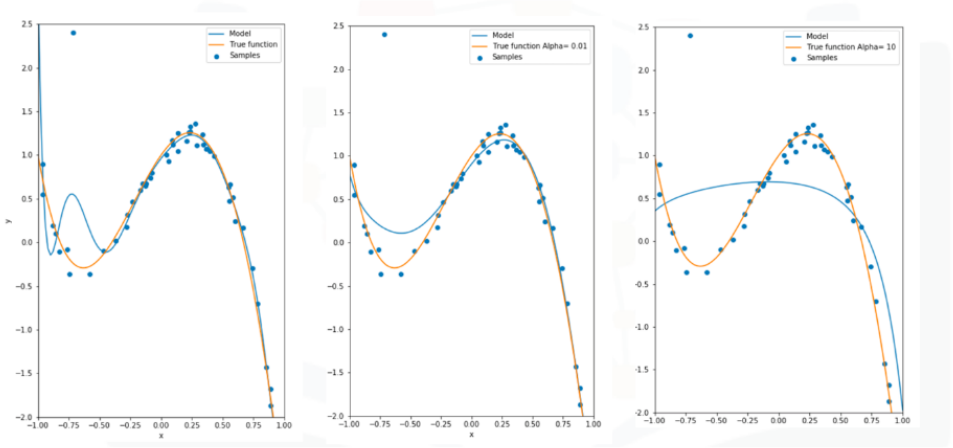


This is similar to test data, but it is used to select parameters like alpha. We start with a small value of alpha. We train the model, make a prediction using the validation data, then calculate the R-squared and store the values. Repeat the value for a larger value of alpha. We train the model again, make a prediction using the validation data, then calculate the R-squared and store the values of R-squared. We repeat the process for a different alpha value, training the model, and making a prediction. We select the value of alpha that maximizes the R-squared. Note that we can use other metrics to select the value of alpha, like mean squared error. The overfitting problem is even worse if we have lots of features.

The following plot shows the different values of R-squared on the vertical axis. The horizontal axis represents different values for alpha. We use several features from our used car data set and a second order polynomial function.

The training data is in red and validation data is in blue. We see as the value for alpha increases, the value of R-squared increases and converges at approximately 0.75. In this case, we select the maximum value of alpha because running the experiment for higher values of alpha has little impact. Conversely, as alpha increases, the R-squared on the test data decreases. This is because the term alpha prevents overfitting. This may improve the results in the unseen data, but the model has worse performance on the test data.

1. the following models were all trained on the same data, select the model with the highest value for alpha: **1 / 1 point**



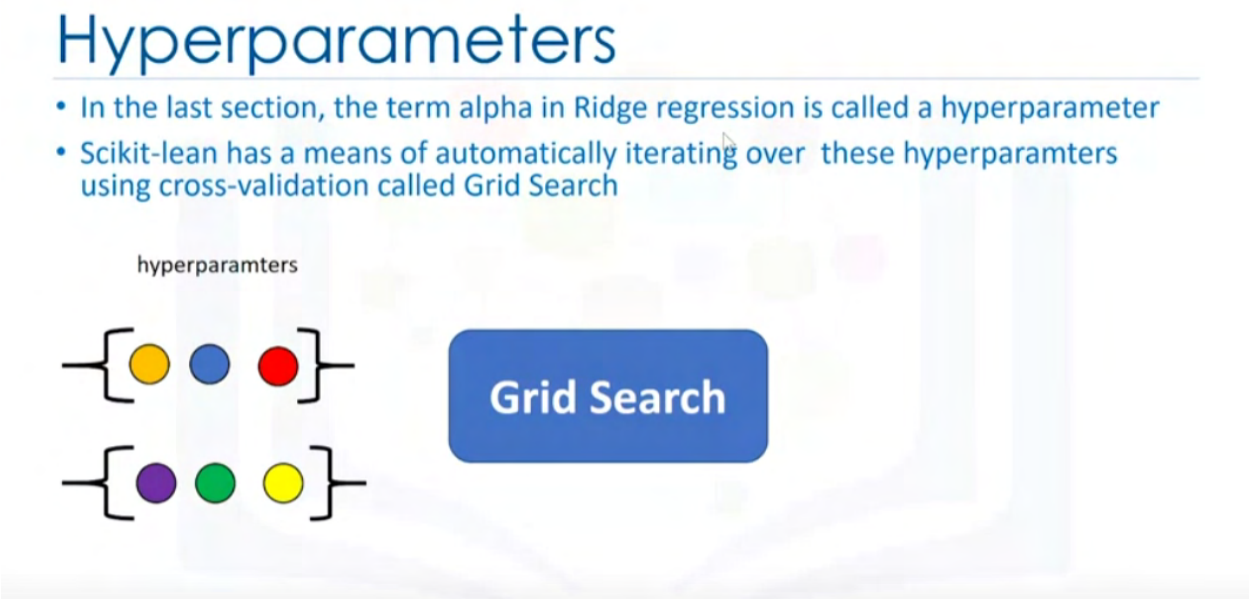
a **b** **c**

- ○ a
- ○ b
- ◉ c

✓ **Correct**

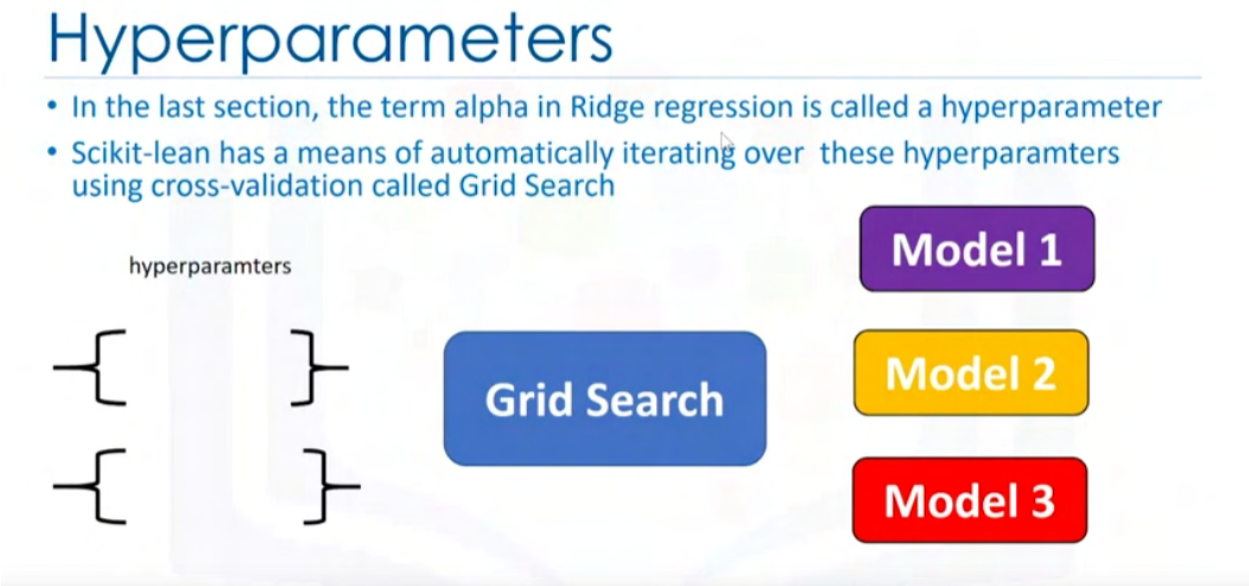
Grid Search

Grid Search allows us to scan through multiple free parameters with few lines of code. Parameters like the alpha term discussed in the previous video are not part of the fitting or training process. These values are called hyperparameters. Scikit-learn has a means of automatically iterating over these hyperparameters using cross-validation. This method is called Grid Search.

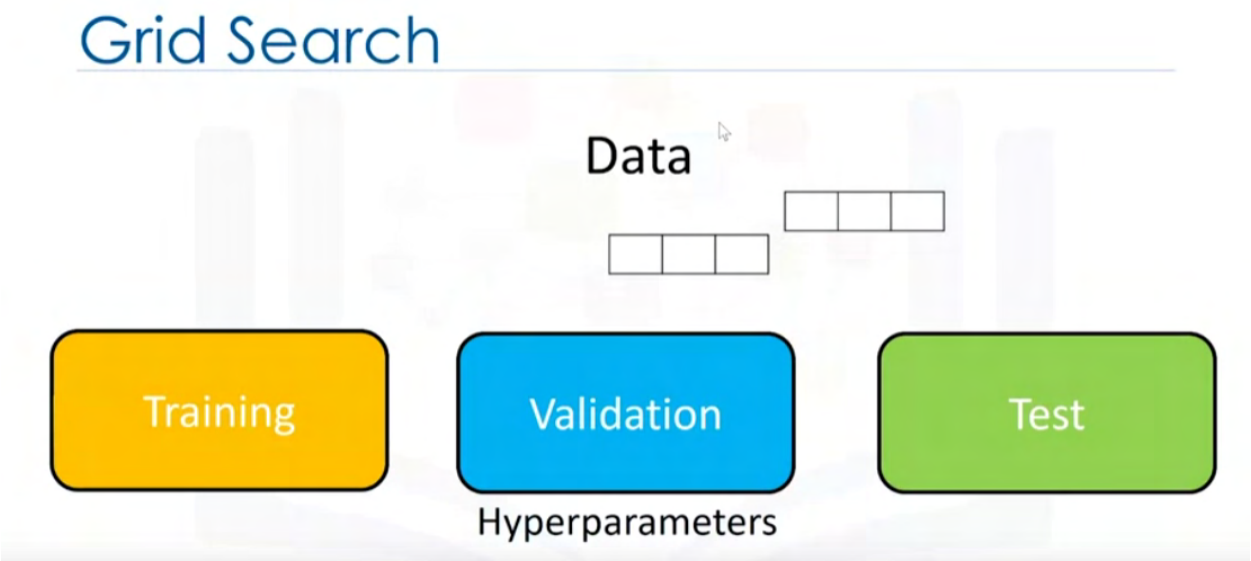


Grid Search takes the model or objects you would like to train and different values of the hyperparameters. It then

calculates the mean square error or R-squared for various hyperparameter values, allowing you to choose the best values.
Let the small circles represent different hyperparameters. We start off with one value for hyperparameters and train the model. We use different hyperparameters to train the model.



We continue the process until we have exhausted the different free parameter values. Each model produces an error. We select the hyperparameter that minimizes the error.
To select the hyperparameter, **we split our dataset into three parts, the training set, validation set, and test set.**



We train the model for different hyperparameters. We use the R-squared or mean square error for each model. We select the hyperparameter that minimizes the mean squared error or maximizes the R-squared on the validation set.

What data do we use to pick the best hyperparameter

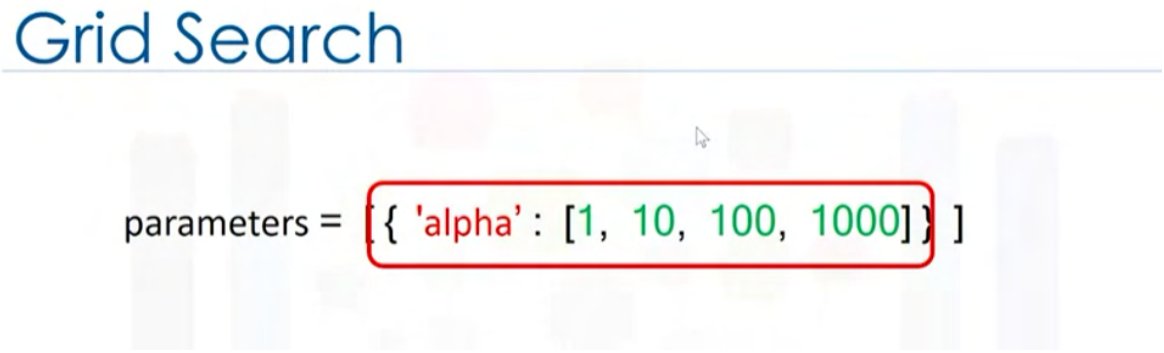
- Training data
- Validation data
- Test data

Correct
correct

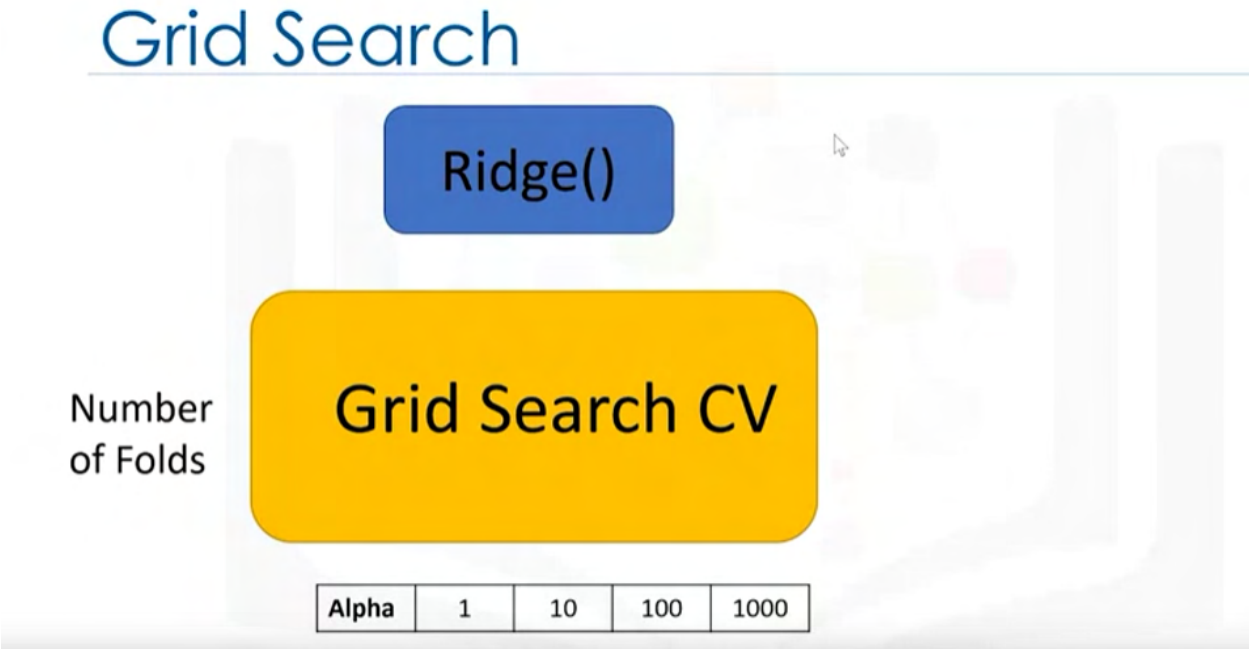
We finally test our model performance using the test data. This is the scikit-learn web page, where the object constructor parameters are given.

It should be noted that the attributes of an object are also called parameters. We will not make the distinction even though some of the options are not hyperparameters per se. In this module, we will focus on the hyperparameter alpha

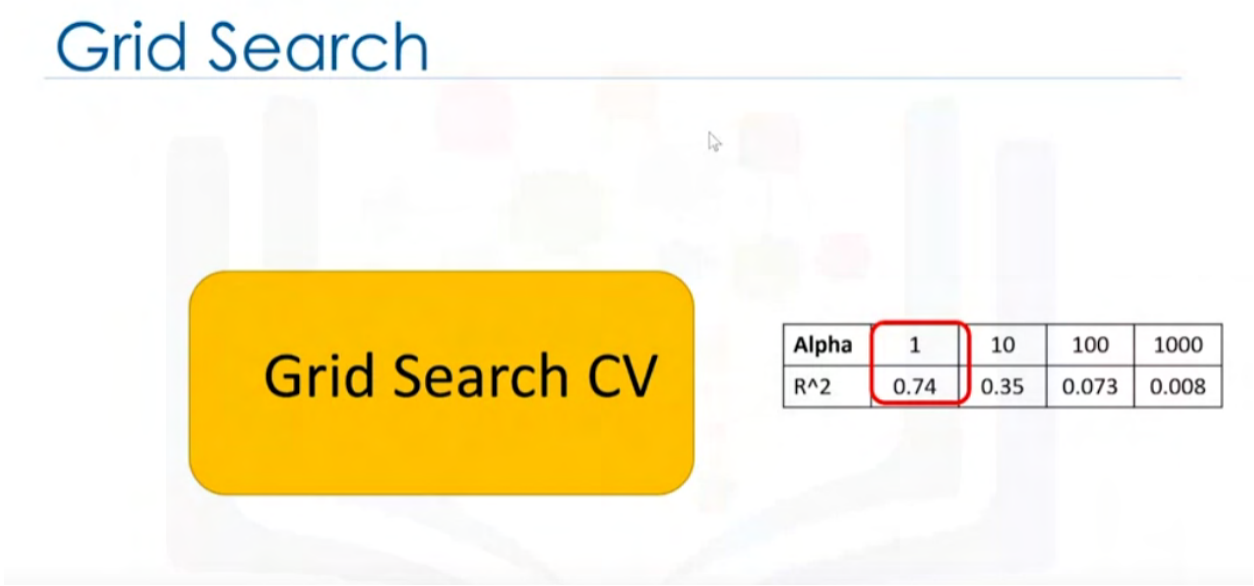
and the normalization parameter. The value of your Grid Search is a Python list that contains a Python dictionary.



The key is the name of the free parameter. The value of the dictionary is the different values of the free parameter.



This can be viewed as a table with various free parameter values. We also have the object or model. The Grid Search takes on the scoring method.

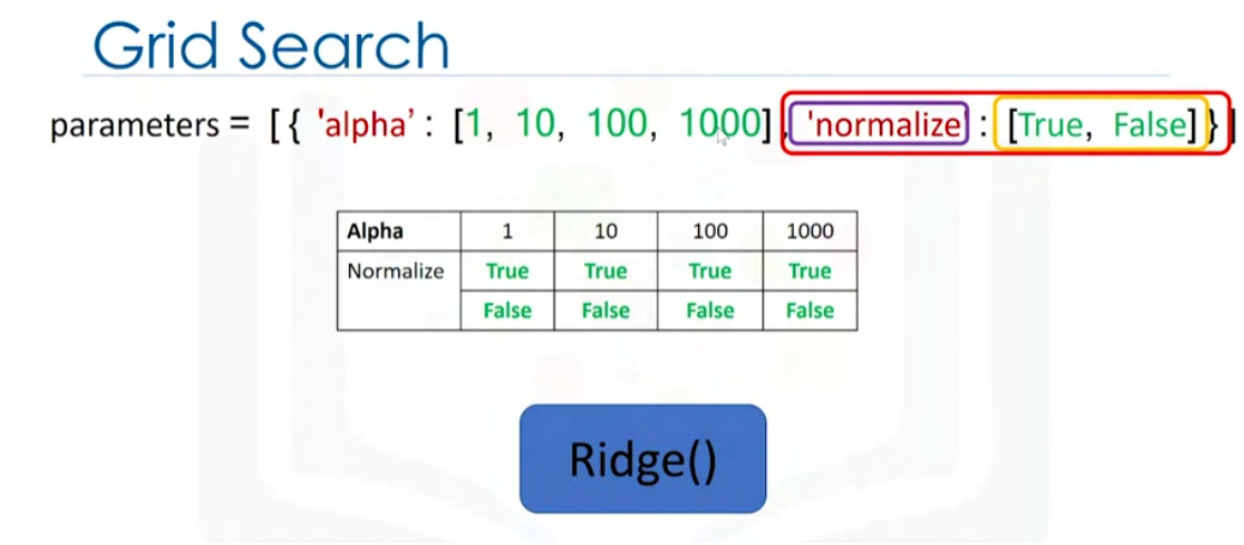


In this case, R-squared the number of folds, the model or object, and the free parameter values. Some of the outputs include the different scores for different free parameter values.

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

parameters1= [{'alpha': [0.001,0.1,1, 10, 100, 1000,10000,100000,100000]}]

RR=Ridge()

Grid1 = GridSearchCV(RR, parameters1,cv=4)

Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y_data)

Grid1.best_estimator_

scores  = Grid1.cv_results_
scores['mean_test_score']
```

In this case, the R-squared along with a free parameter values that have the best score. First, we import the libraries

we need, including GridSearchCV, the dictionary of parameter values



. We create a ridge regression object or model. We then create a GridSearchCV object. The inputs are the ridge regression object, the parameter values, and the number of folds. We will use R-squared. This is the default scoring method. We fit the object. We can find the best values for the free parameters using the attribute best estimator. We can also get information like the mean score on the validation data using the attribute CV result. What are the advantages of Grid Search is how quickly we can test multiple parameters. For example, ridge regression has the option to normalize the data. To see how to standardize, see module four. The term alpha is the first element in the dictionary. The second element is the normalized option. The key is the name of the parameter. The value is the different options in this case because we can either normalize the data or not. The values are True or False

respectively. The dictionary is a table or grid that contains two different values. As before, we need the ridge regression object or model.

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

parameters2= [{'alpha': [0.001,0.1,1, 10, 100], 'normalize' :  [True,  False] }]

RR=Ridge()

Grid1 = GridSearchCV(RR, parameters2,cv=4)

Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y_data)

Grid1.best_estimator_

scores  = Grid1.cv_results_
```

The procedure is similar except that we have a table or grid of different parameter values. The output is the score for all the different combinations of parameter values. The code is also similar. The dictionary contains the different free parameter values. We can find the best value for the free parameters. The resulting scores of the different free parameters are stored in this dictionary, Grid1.cv_results _. We can print out the score for the different free parameter values.

how many types of parameters does the following dictionary contain:

```
parameters= [{'alpha': [0.001,0.1,1, 10, 100],
'normalize' : [True, False] }]
```

2

Lesson Summary

In this lesson, you have learned how to:
Identify overfitting and under-fitting in a predictive model: Overfitting occurs when a function is too closely fit to the training data points and captures the noise of the data. Underfitting refers to a model that can't model the training data or capture the trend of the data.
Apply Ridge Regression to linear regression models: Ridge regression is a regression that is employed in a Multiple regression model when Multicollinearity occurs.
Tune hyper-parameters of an estimator using Grid search: Grid search is a time-efficient tuning technique that exhaustively computes the optimum values of hyperparameters performed on specific parameter values of estimators.

1. What is the code to create a ridge regression object **RR** with an alpha term equal to 10? **1 point**

```
RR=LinearRegression(alpha=10)
```

```
RR=Ridge(alpha=10)
```

```
RR=Ridge(alpha=1)
```

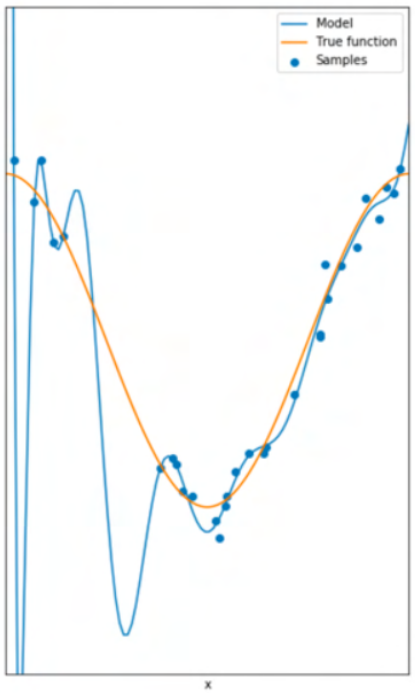
2. What dictionary value would we use to perform a grid search for the following values of alpha? 1,10, 100. No other parameter values should be tested **1 point**

```
alpha=[1,10,100]
```

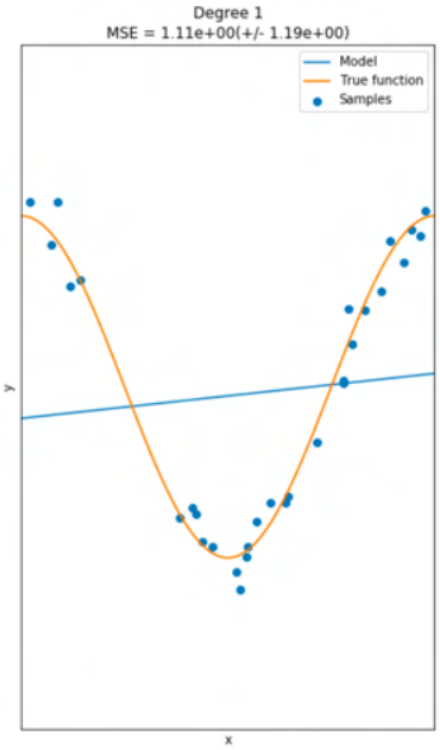
```
[{'alpha': [1,10,100]}]
```

```
[{'alpha': [0.001,0.1,1, 10, 100, 1000,10000,100000,100000],'normalize':[True,False]
```

4. The following is an example of what?



- (●) Overfitting



- ○ Overfitting
- ○ Perfect fit
- ◉ Underfitting

Project Scenario

In this assignment, you are a Data Analyst working at a Real Estate Investment Trust. The Trust would like to start investing in Residential real estate. You are tasked with determining the market price of a house given a set of features. You will analyze and predict housing prices using attributes or features such as square footage, number of bedrooms, number of floors, and so on. A template notebook is provided in the lab; your job is to complete the ten questions. Some hints to the questions are given in the template notebook.

Dataset Used in this Assignment

The dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. It was taken from here. It was also slightly modified for the purposes of this course.

For this project, you will utilize JupyterLab running on the Cloud in Skills Network Labs environment.

Notebook URL: Alternatively, you can work on your local machine or any other environment of choice, by downloading this link : Notebook link House Sales

Lab for Final Project

For this final project, you will utilize Juoyterlite running on the Cloud in Skills Network Labs environment. To launch the lab notebook in a new browser tab, check the box below and click on the "Launch App" button.

To work on your local machine or in any other environment, download the lab notebook (.ipynb file) by clicking HERE .

As you are **completing this notebook**, take and save the screenshots of the final outputs of your solutions (e.g., final charts, tables, calculation results etc.). They will need to be shared in the following Peer Review section of the Final Project module.

Once you have **completed the lab,** you can download the notebook by navigating to "File" and clicking on "Download" button. This will save the (.ipynb) file on your computer.

Once saved, you can upload "House_Sales_in_King_Count_USA.ipynb" file to "My Submission" tab, of the "Peer-graded Assignment" section.

In the next, **"Peer Review"** section of the course, follow the instructions to submit the downloaded "House_Sales_in_King_Count_USA.ipynb" file, along with your screenshots.