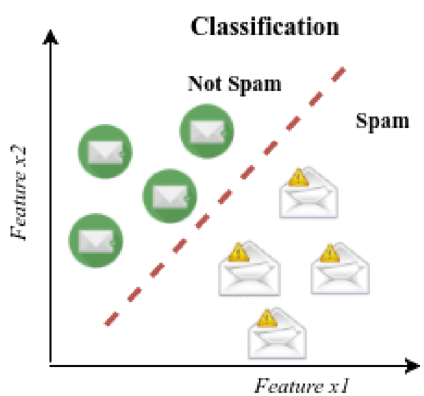
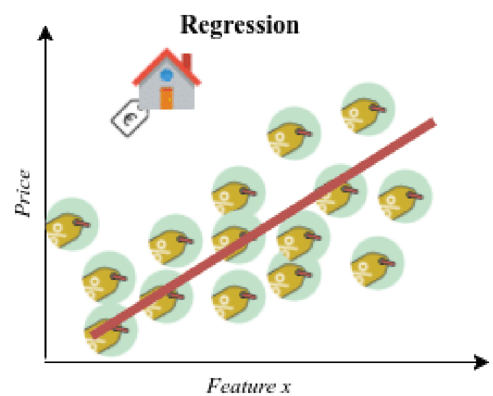
Lecture 12:
Linear and Multiple Regression

In the last lecture

- Up until now
  - Classification: KNN, Decision Tree, Ensemble Trees
  - Clustering: K-Means, Agglomerative Filtering, DBSCAN
- Today
  - Regression: Linear and Multiple regression





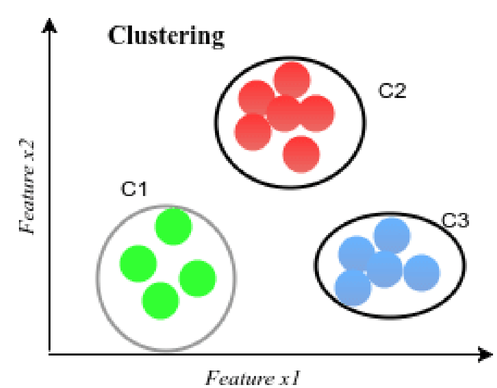


Table of Contents

Understanding regression

- Regression
  - Regression analysis is commonly used for modeling complex relationships among data elements.
    - Examining how populations and individuals vary by their measured characteristics
      - For use in scientific research across fields as diverse as economics, sociology, psychology, physics, and ecology.
    - Quantifying the causal relationship between an event and the response
      - Such as those in clinical drug trials, engineering safety tests, or marketing research.
    - Identifying patterns that can be used to forecast future behavior given known criteria
      - Such as predicting insurance claims, natural disaster damage, election results, and crime rates.

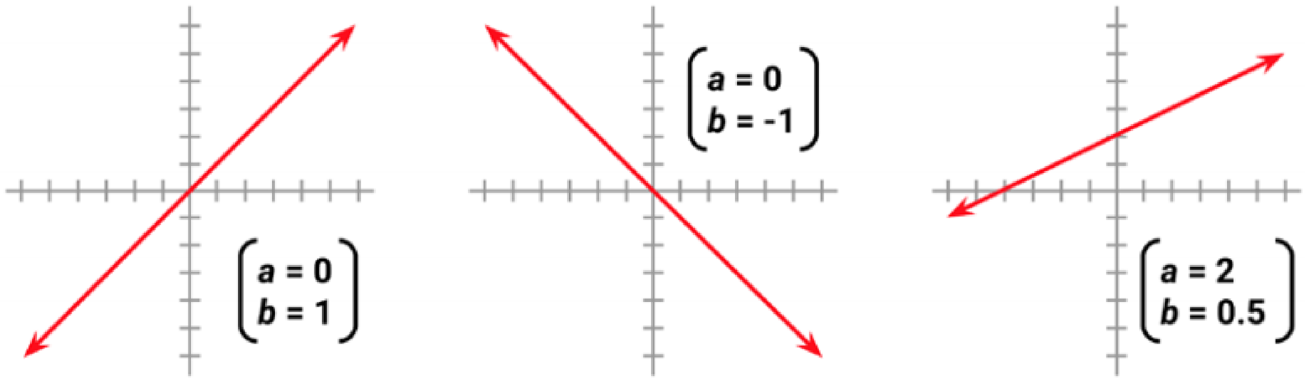
Understanding regression

- Regression
  - Regression is concerned with specifying the relationship between two parameters, such as:
    - a single numeric **dependent variable** (the value to be predicted)
    - one or more numeric **independent variables** (the predictors)
  - The dependent variable depends upon the value of the independent variable or variables.
  - The relationship between the independent and dependent variables follows **a straight line**.

Understanding regression

- Regression
  - Lines can be defined in a slope-intercept form similar to
    - $y = a + bx$
      - y indicates the dependent variable
      - x indicates the independent variable
  - The **slope** term b specifies how much the line rises for each increase in x.
    - Positive values define lines that slope upward
    - Negative values define lines that slope downward
  - The term a is known as the **intercept**
    - The point where the line crosses, or intercepts, the vertical y axis.

Understanding regression



- Regression
  - The machine's job is to identify values of *a* and *b*
    - The specified line is best able to relate the supplied x values to the values of y.
    - The machine must also have some way to quantify the margin of error.

Understanding regression

- Ordinary least squares estimation
  - In order to determine the optimal estimates of a and b, an estimation method known as **Ordinary Least Squares (OLS)** was used.
  - In OLS regression, the slope and intercept are chosen so that they minimize the sum of the **squared errors**
    - The vertical distance between the predicted y value and the actual y value.
  - These errors are known as **residuals**

Understanding regression

- Ordinary least squares estimation
  - It can be shown using calculus that the value of b that results in the minimum squared error is:

$$b = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \qquad b = \frac{\mathrm{Cov}(x,y)}{\mathrm{Var}(x)}$$

  - Denominator
    - The variance finds the average squared deviation from the mean of x.

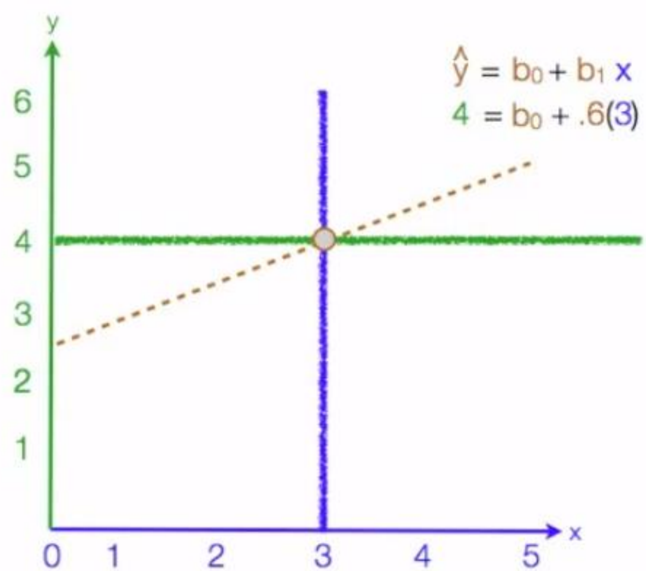
$$\mathrm{Var}(x) = \frac{\sum(x_i - \bar{x})^2}{n}$$

  - Numerator
    - Take the sum of each data point's deviation from the mean x value multiplied by that point's deviation away from the mean y value.

$$\mathrm{Cov}(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n}$$

Understanding regression

- Step-by-step example of regression
  - https://www.youtube.com/watch?v=zPG4NjIkCjc



$b_0 = 2.2$
$b_1 = .6$
$\hat{y} = 2.2 + .6x$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	2	-2	-2	4	4
2	4	-1	0	1	0
3	5	0	1	0	0
4	4	1	0	1	0
5	5	2	1	4	2
				10	6
mean 3	4				

$4 = b_0 + .6(3)$

$4 = b_0 + \cancel{1.8}$

$-1.8 \quad\quad -\cancel{1.8}$

$2.2 = b_0$

$$b_1 = \frac{6}{10} = .6 = \frac{\Sigma(x - \bar{x})(y - \bar{y})}{\Sigma(x - \bar{x})^2}$$

Understanding regression

- Estimating errors
  - R-squared is a goodness-of-fit measure for linear regression models.
  - This statistic indicates the percentage of the variance in the dependent variable that the independent variables explain collectively.
  - R-squared measures the strength of the relationship between your model and the dependent variable on a convenient 0 – 100% scale.
    - the closer the value is to 1.0, the better the model perfectly explains the data

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

- Step-by-step example of regression
  - https://www.youtube.com/watch?v=r-txC-dpI-E&ab_channel=statisticsfun

Understanding regression

- Simple Linear Regression
  - On January 28, 1986, seven crew members of the United States space shuttle Challenger were killed when a rocket booster failed, causing a catastrophic disintegration.
  - https://www.youtube.com/watch?v=j4JOjcDFtBE

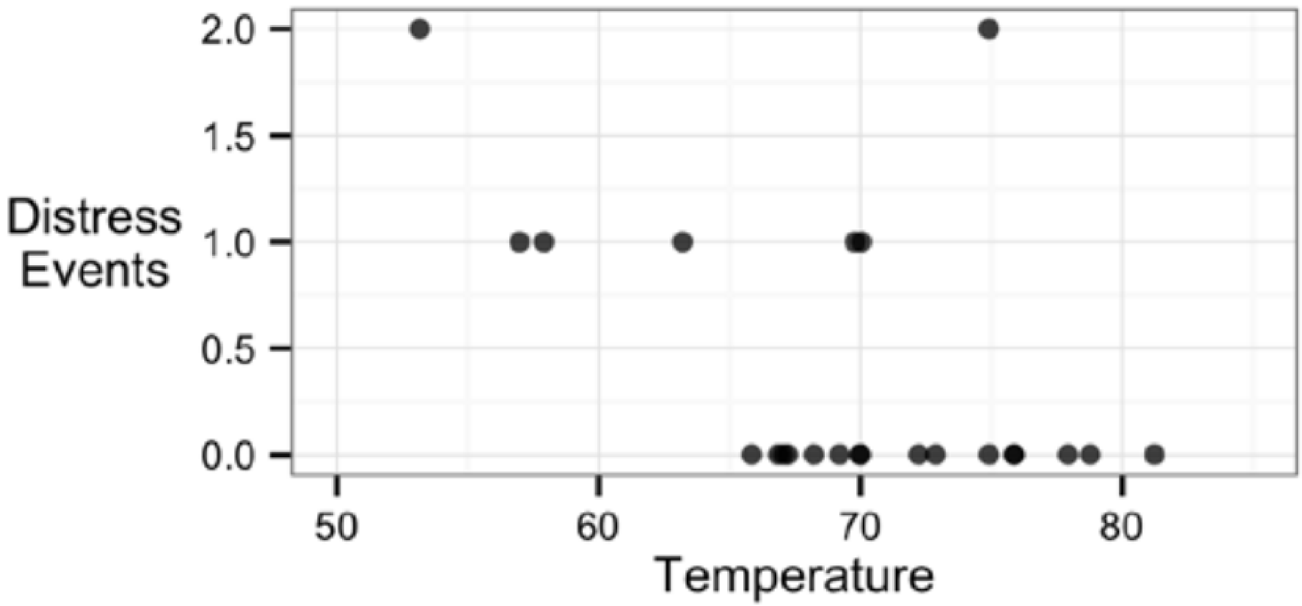
Understanding regression

- Simple Linear Regression
  - In the aftermath, experts focused on the launch temperature as a potential culprit.
  - The rubber O-rings responsible for sealing the rocket joints had never been tested below 40°F (4°C).

  - The weather on the launch day was unusually cold and below freezing.

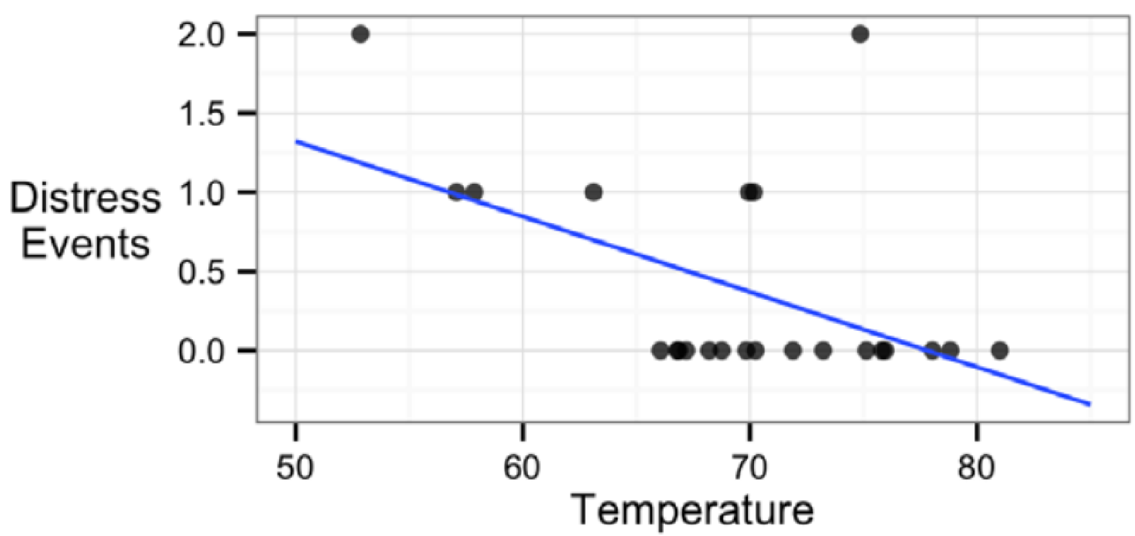
Understanding regression

- Simple Linear Regression
  - The following scatterplot shows a plot of primary O-ring distresses detected for the previous 23 launches



Understanding regression

- Simple Linear Regression
  - Linear Regression
    - The relationship between a dependent variable and a single independent predictor variable using a line defined by an equation in the following form: ***y = a + bx***
    - Suppose: a = 3.70 and b = -0.048



Understanding regression

- Simple Linear Regression
  - As the line shows, at 60 degrees Fahrenheit, we predict just under one O-ring distress.
  - At 70 degrees Fahrenheit, we expect around 0.3 failures.
  - At 31 degrees, would expect about 3.70 - 0.048 * 31 = 2.21 O-ring distress events.
  - Assuming that each O-ring failure is equally likely to cause a catastrophic fuel leak means that the Challenger launch at 31 degrees
    - Nearly three times more risky than the typical launch at 60 degrees
    - Over eight times more risky than a launch at 70 degrees.

Understanding regression

- Multiple linear regression
  - Most real-world analyses have more than one independent variable
  - It is likely that you will be using multiple linear regression for most numeric prediction tasks
  - We can understand multiple regression as an extension of simple linear regression
  - The goal in both cases is similar
    - Find values of beta coefficients that minimize the prediction error of a linear equation

Understanding regression

- Multiple linear regression
  - The strengths and weaknesses of the algorithm are as follows:

Strengths	Weaknesses
• By far the most common approach for modeling numeric data • Can be adapted to model almost any modeling task • Provides estimates of both the strength and size of the relationships among features and the outcome	• Makes strong assumptions about the data • The model's form must be specified by the user in advance • Does not handle missing data • Only works with numeric features, so categorical data requires extra processing • Requires some knowledge of statistics to understand the model

Understanding regression

- Multiple linear regression
  - Multiple regression equations generally follow the form of the following equation

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i + \varepsilon$$

  - Here
    - *y* is dependent variable
    - *a* is an intercept term
    - *β* is estimated value
    - *x* values for each of the i features
    - e is residual

Regression in Python

- There are two ways to build regression models in Python

- Models
  - Simple Linear Regression (SLR)
  - Multiple Linear Regression (MLR)

Regression in Python

- Step 1: Loading Dataset
  - Dataset
    - CO2 emission from a vehicle in Canada

```
import pandas as pd

df = pd.read_csv('D:\\co2.csv')
df.head(5)
```

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
0	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	33	196
1	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	29	221
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	48	136
3	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	25	255
4	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	27	244

Regression in Python

- Step 2: Data Observation
  - df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 12 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Make                              7385 non-null   object 
 1   Model                             7385 non-null   object 
 2   Vehicle Class                     7385 non-null   object 
 3   Engine Size(L)                    7385 non-null   float64
 4   Cylinders                         7385 non-null   int64  
 5   Transmission                      7385 non-null   object 
 6   Fuel Type                         7385 non-null   object 
 7   Fuel Consumption City (L/100 km)  7385 non-null   float64
 8   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 9   Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 10  Fuel Consumption Comb (mpg)       7385 non-null   int64  
 11  CO2 Emissions(g/km)               7385 non-null   int64  
dtypes: float64(4), int64(3), object(5)
memory usage: 692.5+ KB
```

Regression in Python

- Step 2: Data Observation
  - df.describe()

	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
count	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000
mean	3.160068	5.615030	12.556534	9.041706	10.975071	27.481652	250.584699
std	1.354170	1.828307	3.500274	2.224456	2.892506	7.231879	58.512679
min	0.900000	3.000000	4.200000	4.000000	4.100000	11.000000	96.000000
25%	2.000000	4.000000	10.100000	7.500000	8.900000	22.000000	208.000000
50%	3.000000	6.000000	12.100000	8.700000	10.600000	27.000000	246.000000
75%	3.700000	6.000000	14.600000	10.200000	12.600000	32.000000	288.000000
max	8.400000	16.000000	30.600000	20.600000	26.100000	69.000000	522.000000

- Step 3: Exploratory Data Analysis
  - df.drop(['Make','Model','Vehicle Class','Transmission','Fuel Type'], axis = 1, inplace = True)

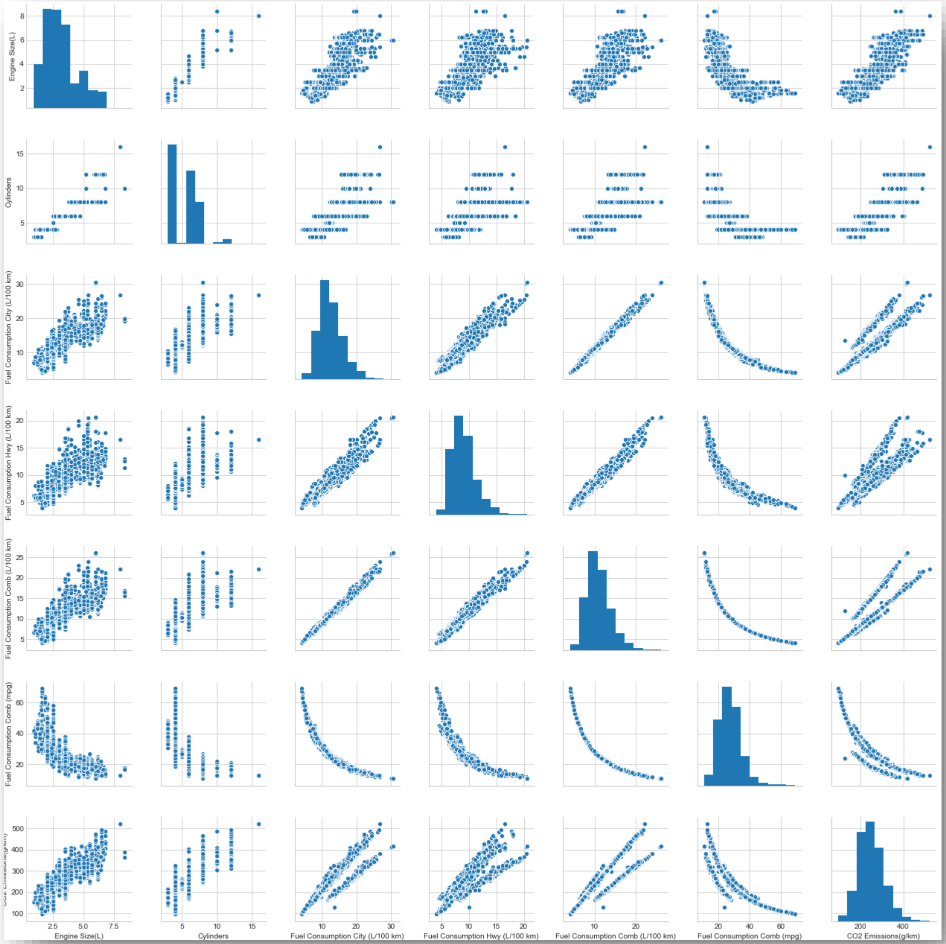
Regression in Python

- Step 3: Exploratory Data Analysis
  - Dependent variable: CO_2 emissions
  - We have to find some positive or negative linear relationships by implementing scatter plots
    - These variables are further used for building our SLR and MLR models

```
import seaborn as sb
import matplotlib.pyplot as plt
from matplotlib import style

style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize'] = (20,10)

sb.pairplot(df)
plt.savefig('pairplor.png')
```

Regression in Python

- Step 3: Exploratory Data Analysis



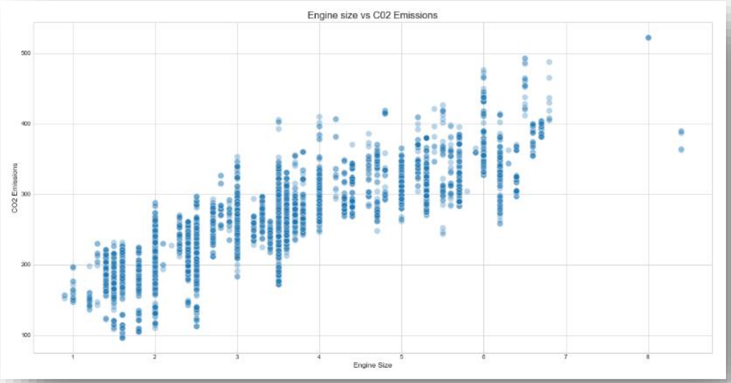
Regression in Python

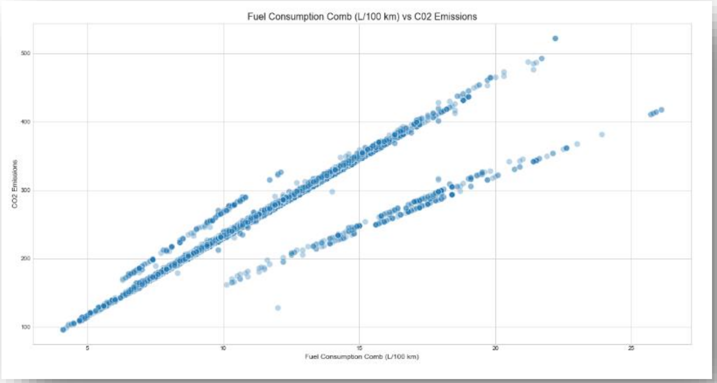
- Step 3: Exploratory Data Analysis
  - Find linear relationships between attributes against CO2
    - Engine size
    - Fuel Consumption Comb
    - Fuel Consumption Hwy (L/100 km)
    - Fuel Consumption City (L/100 km)

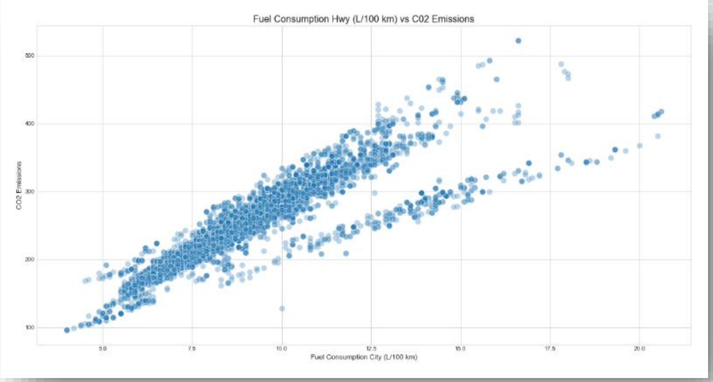
```
plt.scatter(x = 'Engine Size(L)', y = 'CO2 Emissions(g/km)',
                              data = df, s = 100, alpha = 0.3, edgecolor = 'white')

plt.title('Engine size vs C02 Emissions', fontsize = 16)
plt.ylabel('CO2 Emissions', fontsize = 12)
plt.xlabel('Engine Size', fontsize = 12)

plt.savefig('enginesize_co2.png')
```

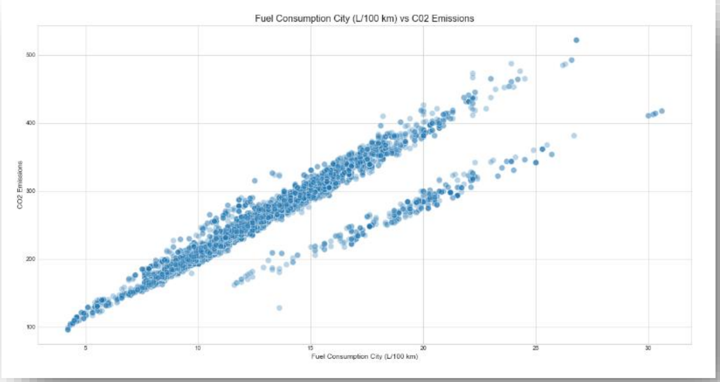
Regression in Python

- Step 3: Exploratory Data Analysis
  - Find linear relationships between attributes against CO2









Regression in Python

- Step 4: Splitting into training and testing datasets (SLR)
  - Using the train_test_split algorithm, we are classifying the training dataset
    - Testing dataset whose size is 30% of the original dataset
    - Training dataset is remaining 70%

```
from sklearn.model_selection import train_test_split

X_var = df[['Engine Size(L)']] # independent variable
y_var = df['CO2 Emissions(g/km)'] # dependent variable

X_train, X_test, y_train, y_test = train_test_split(
            X_var, y_var, test_size = 0.3, random_state = 0)
```

Regression in Python

- Step 5: Training model (SLR)
  - sklearn library for training the dataset using linear model

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)
yhat = lr.predict(X_test)
```

- Step 6: Checking accuracy (SLR)

```
from termcolor import colored as cl

print(cl('R-Squared :', attrs = ['bold']),
          lr.score(X_test, y_test))

#R-Squared : 0.7162770226132333
```

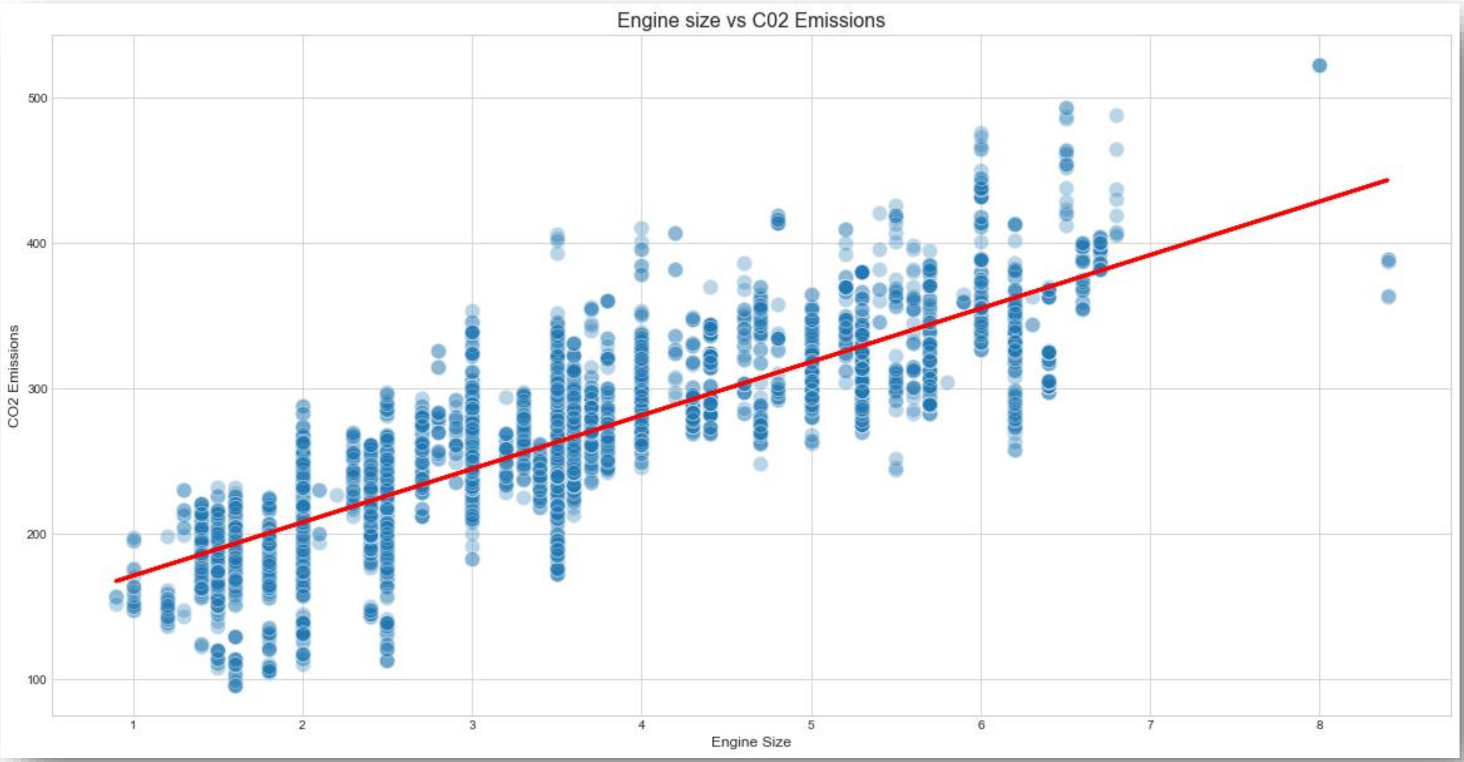
Regression in Python

- Step 6: Checking accuracy (SLR)
  - You can obtain slope and intercept values from the model

```
slr_slope = lr.coef_
slr_intercept = lr.intercept_

sb.scatterplot(x = 'Engine Size(L)', y = 'CO2 Emissions(g/km)',
                    data = df, s = 150, alpha = 0.3, edgecolor = 'white')
plt.plot(df['Engine Size(L)'], slr_slope*df['Engine Size(L)'] + slr_intercept,
                    color = 'r', linewidth = 3)
plt.title('Engine size vs C02 Emissions', fontsize = 16)
plt.ylabel('CO2 Emissions', fontsize = 12)
plt.xlabel('Engine Size', fontsize = 12)

plt.savefig('enginesize_co2_fit.png')
```

Regression in Python

- Step 6: Checking accuracy (SLR)
  - You can obtain slope and intercept values from the model



Regression in Python

- Step 4: Splitting into training and testing datasets (MLR)

```
from sklearn.model_selection import train_test_split

X1_var = df[['Engine Size(L)',
                    'Fuel Consumption Comb (L/100 km)',
                    'Fuel Consumption Hwy (L/100 km)',
                    'Fuel Consumption City (L/100 km)']]
y_var = df['CO2 Emissions(g/km)'] # dependent variable

X_train, X_test, y_train, y_test = train_test_split(
                    X1_var,
                    y_var,
                    test_size = 0.3,
                    random_state = 0)
```

Regression in Python

- Step 5: Training model and checking out accuracy (MLR)

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)
yhat = lr.predict(X_test)
```

- Step 6: Checking accuracy (MLR)

```
from termcolor import colored as cl

print(cl('R-Squared :', attrs = ['bold']),
              lr.score(X_test, y_test))

#R-Squared : 0.8655946234480003
```

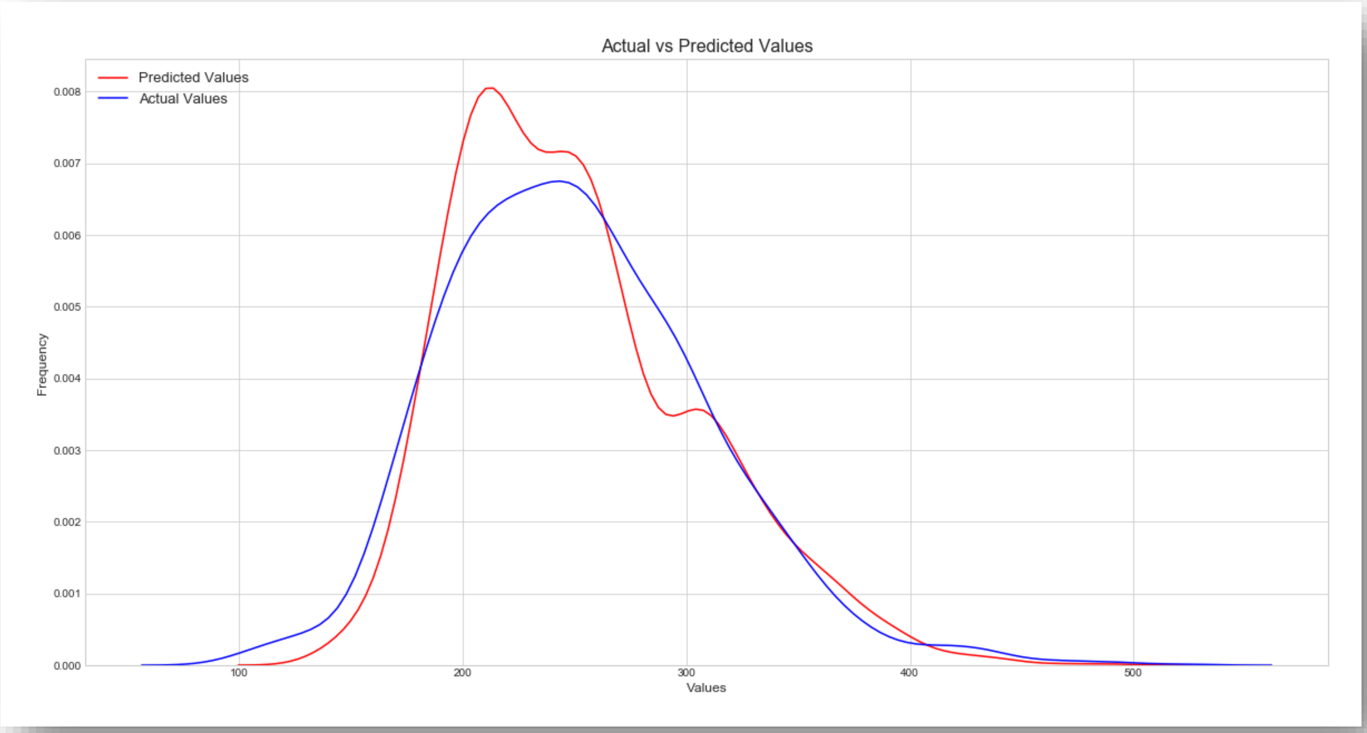
Regression in Python

- Step 6: Checking accuracy (MLR)
  - Constructing a distribution plot by combining the predicted val ues and the actual values

```
sb.distplot(yhat, hist = False, color = 'r', label = 'Predicted Values')
sb.distplot(y_test, hist = False, color = 'b', label = 'Actual Values')
plt.title('Actual vs Predicted Values', fontsize = 16)
plt.xlabel('Values', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)
plt.legend(loc = 'upper left', fontsize = 13)

plt.savefig('ap.png')
```

Regression in Python

- Step 6: Checking accuracy (MLR)
  - Constructing a distribution plot by combining the predicted val ues and the actual values



Regression in Python

- Submit your source code for the following task:
  1. Try all source code in the lecture

- Submission: source code, result screenshots and result explanation

- Deadline: January 24, 2022, 11:59

Q&A

This lecture is supported by Seondo project of the Ministry of Education in Korea.