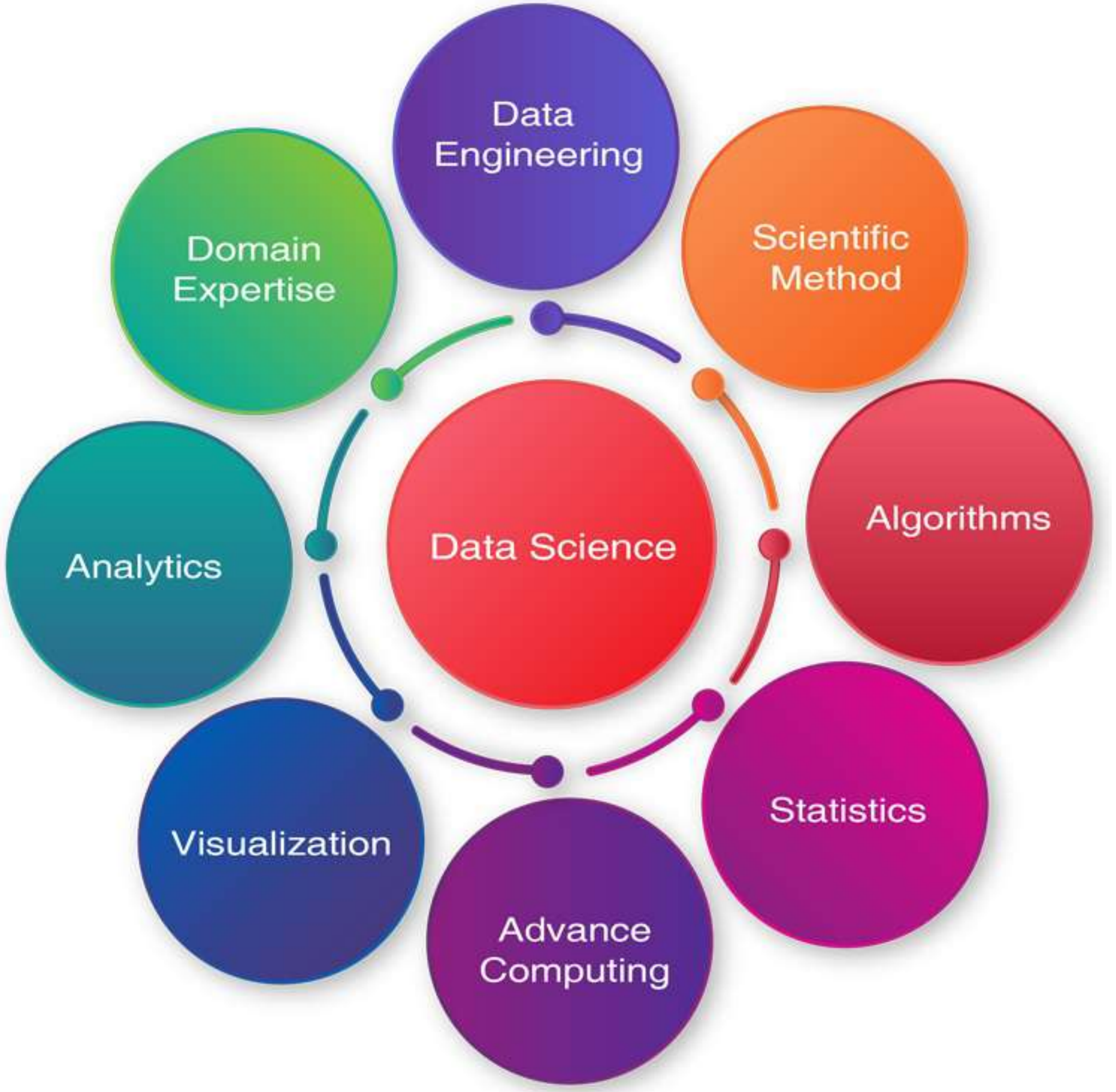
# Exploratory Data Analysis

**M Octaviano Pratama, S.Kom., M.Kom**

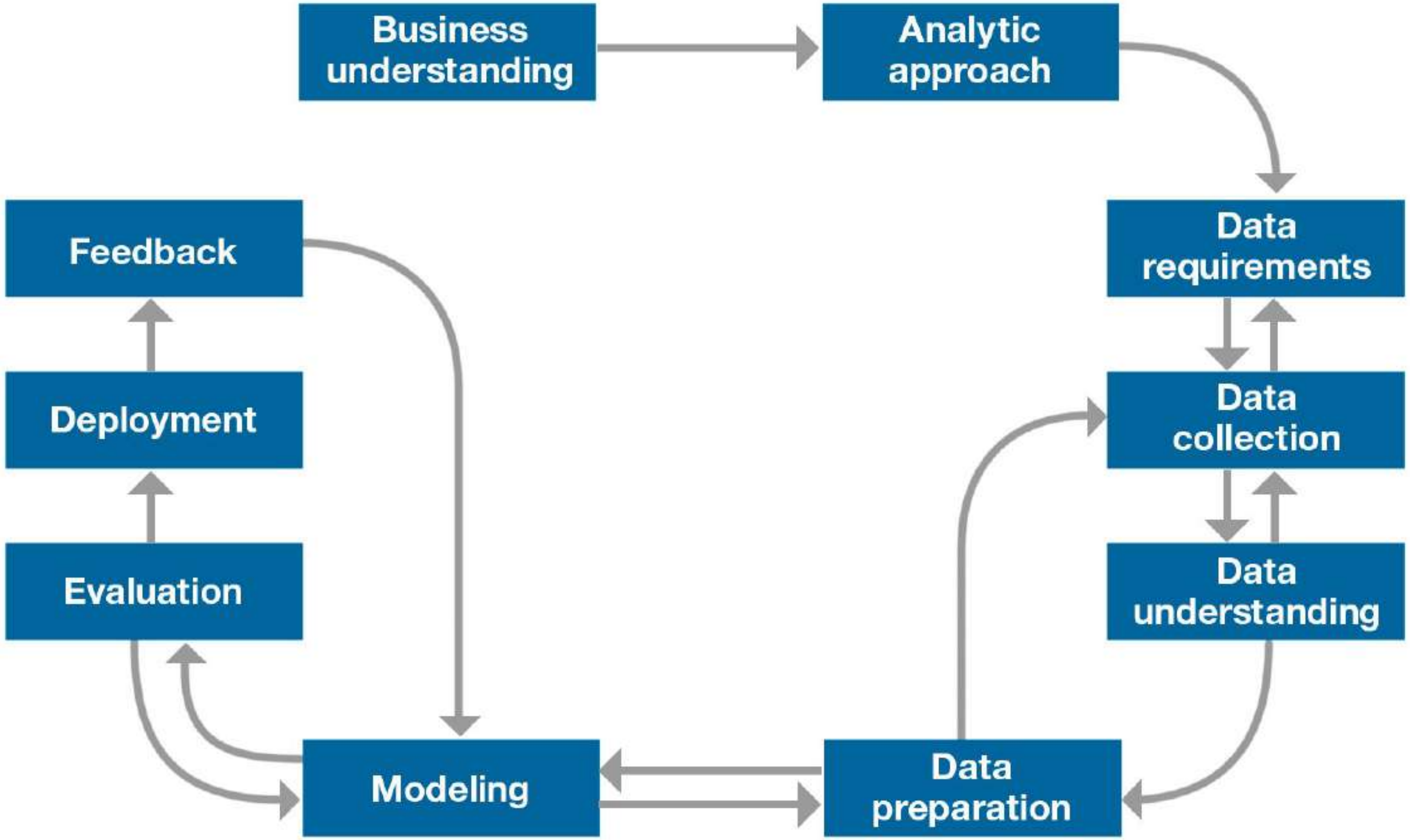
Director BISA AI ACADEMY

(PT BISA ARTIFISIAL INDONESIA)

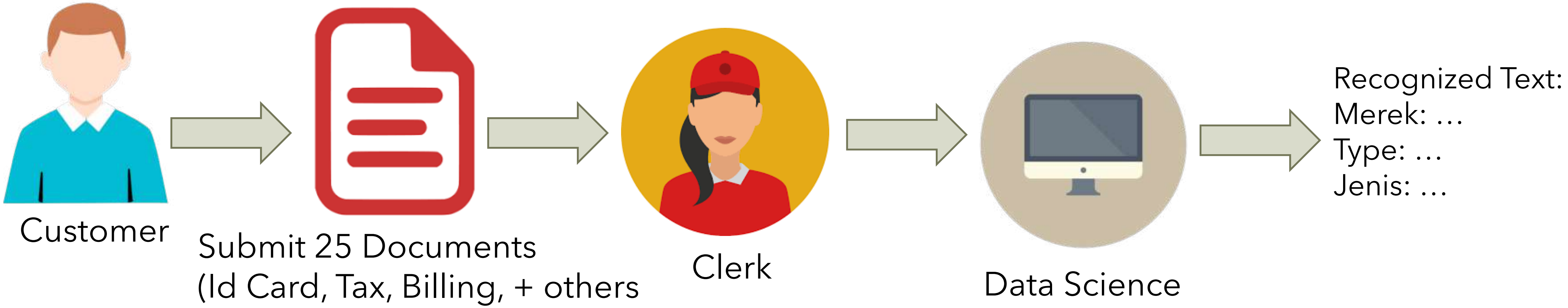
# Data Science



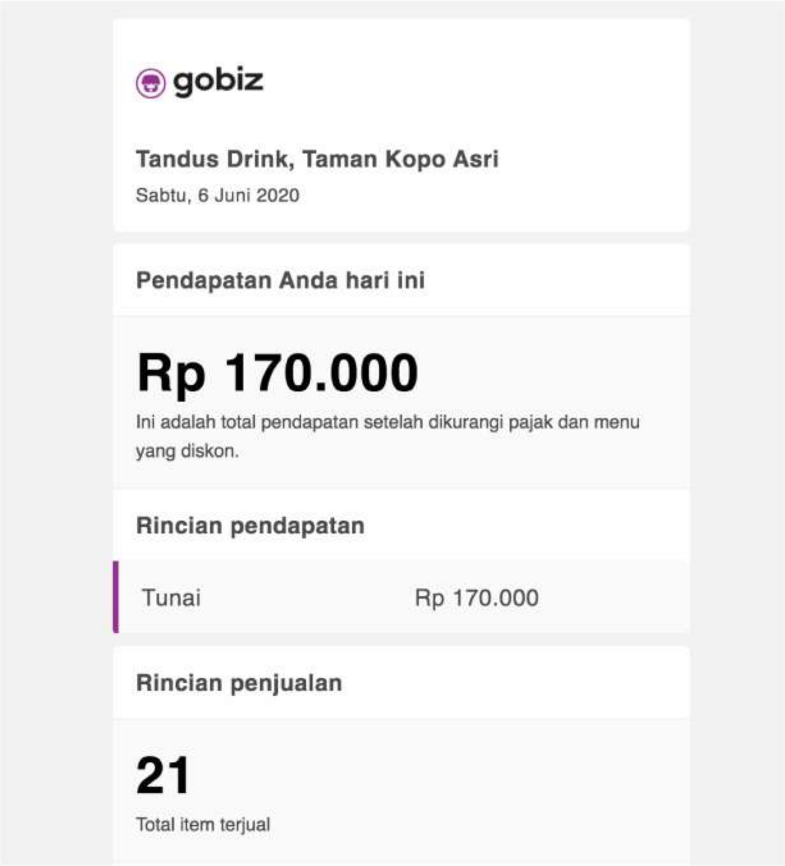
# Data Science Methodology

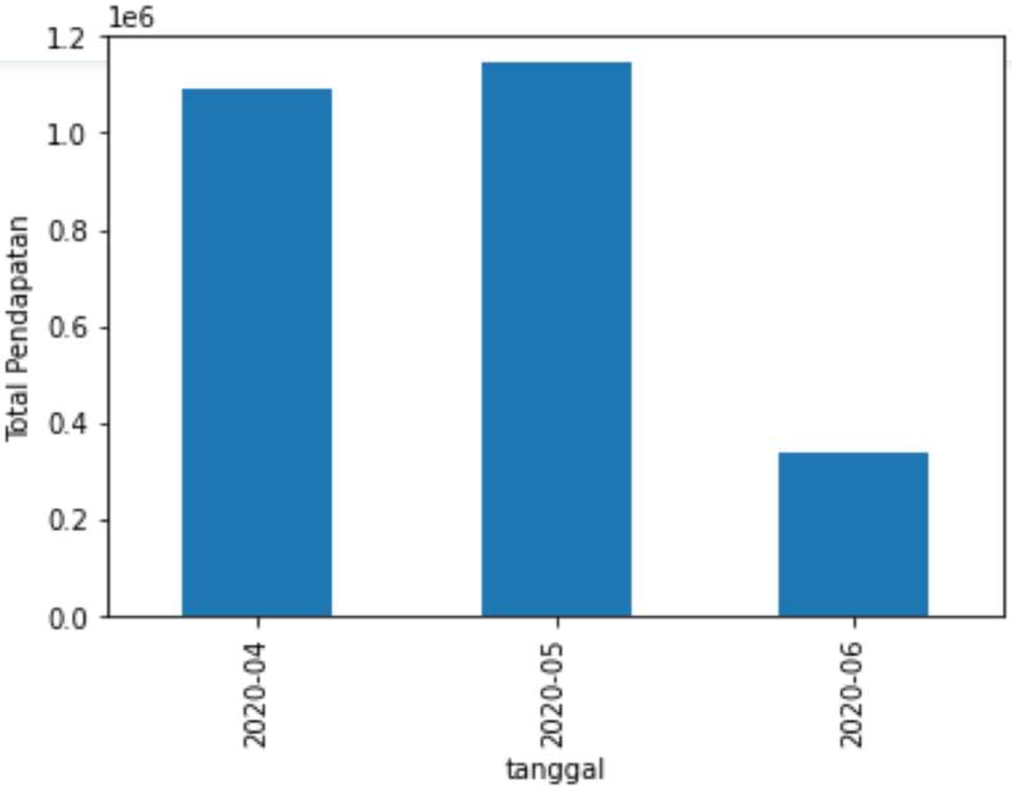


# Contoh Kasus Data Science



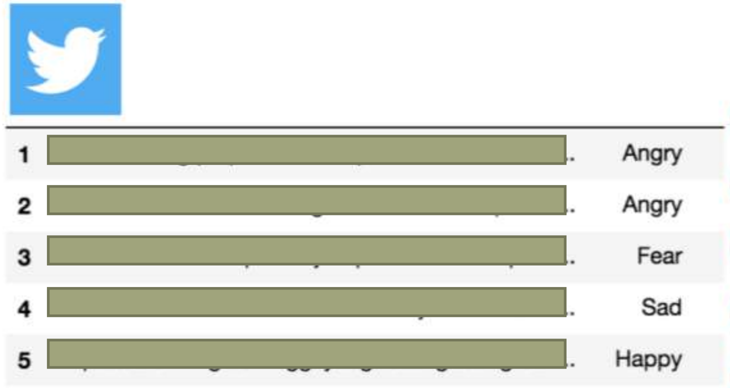
# Contoh Kasus Data Science



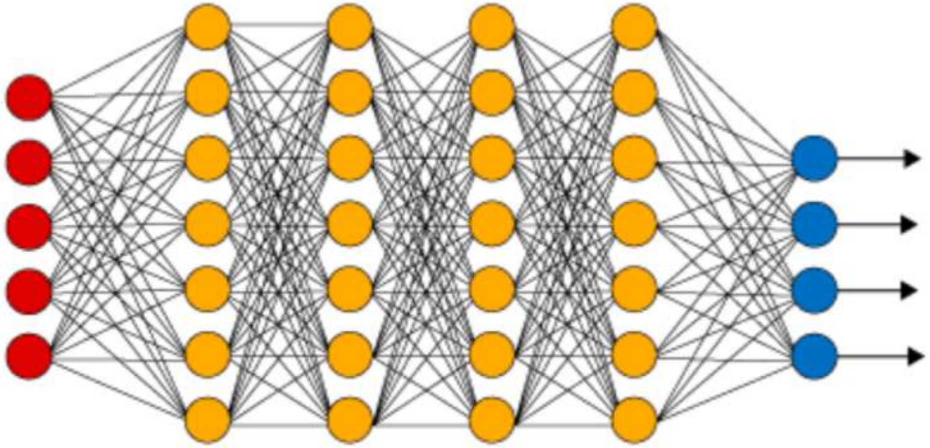



# Contoh Kasus Data Science

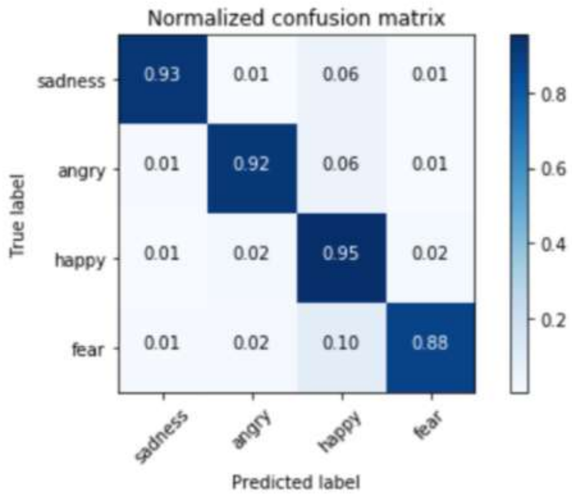
**Sentiment analysis is useful for detecting emotion or sentiment from topics or products. Data is gathered from social media like Twitter, then Machine Learning is performed to detect emotion or sentiment in topics or products.**




Crawling from Twitter Data



**Deep Learning Architecture**

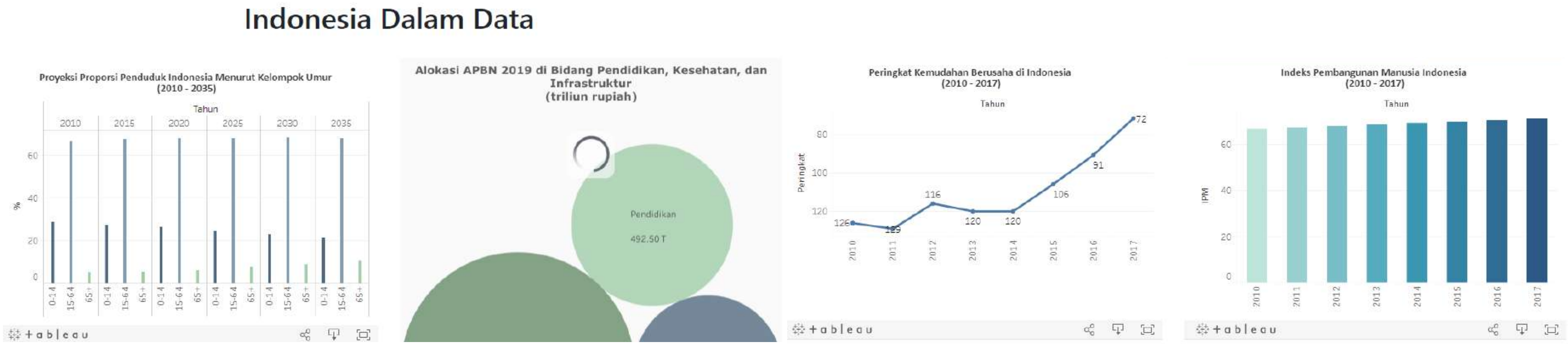



Sentiment or Emotion Score

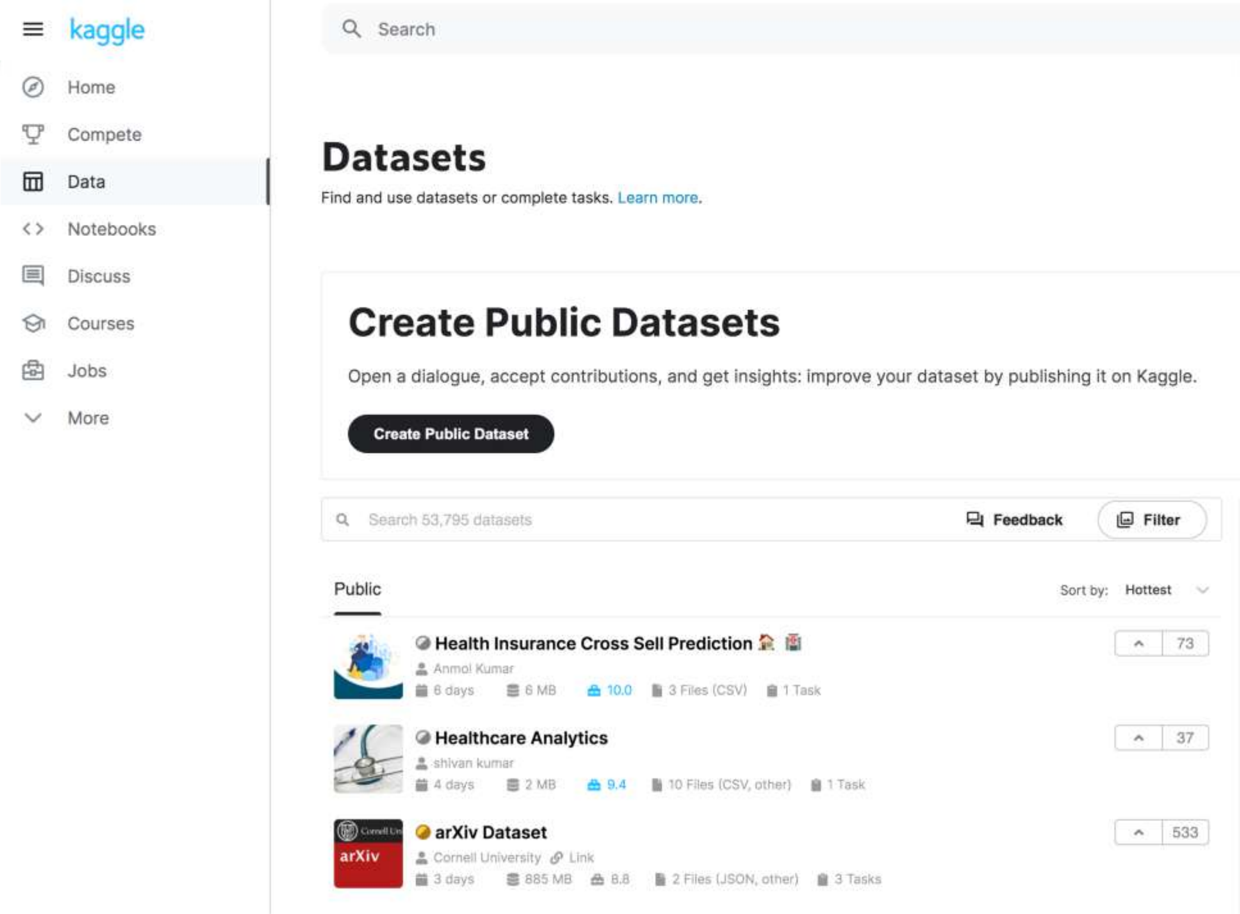
# Get Data

- From database or Big Data
- From public dataset
- Create by yourself

# https://data.go.id/

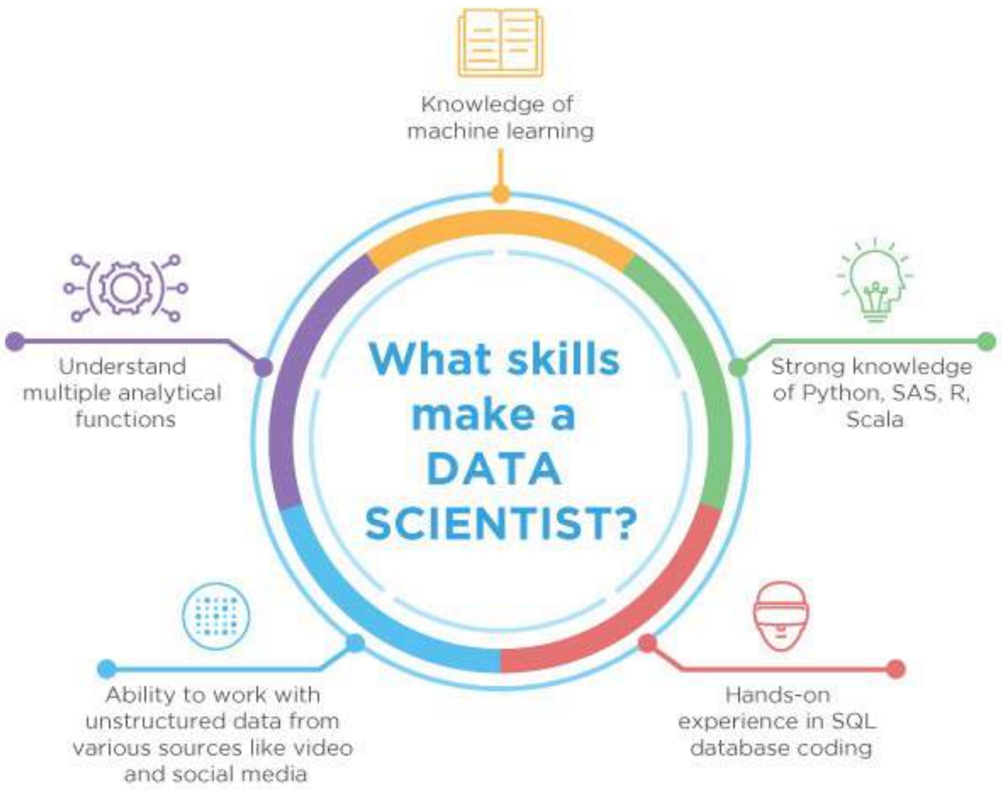


# kaggle

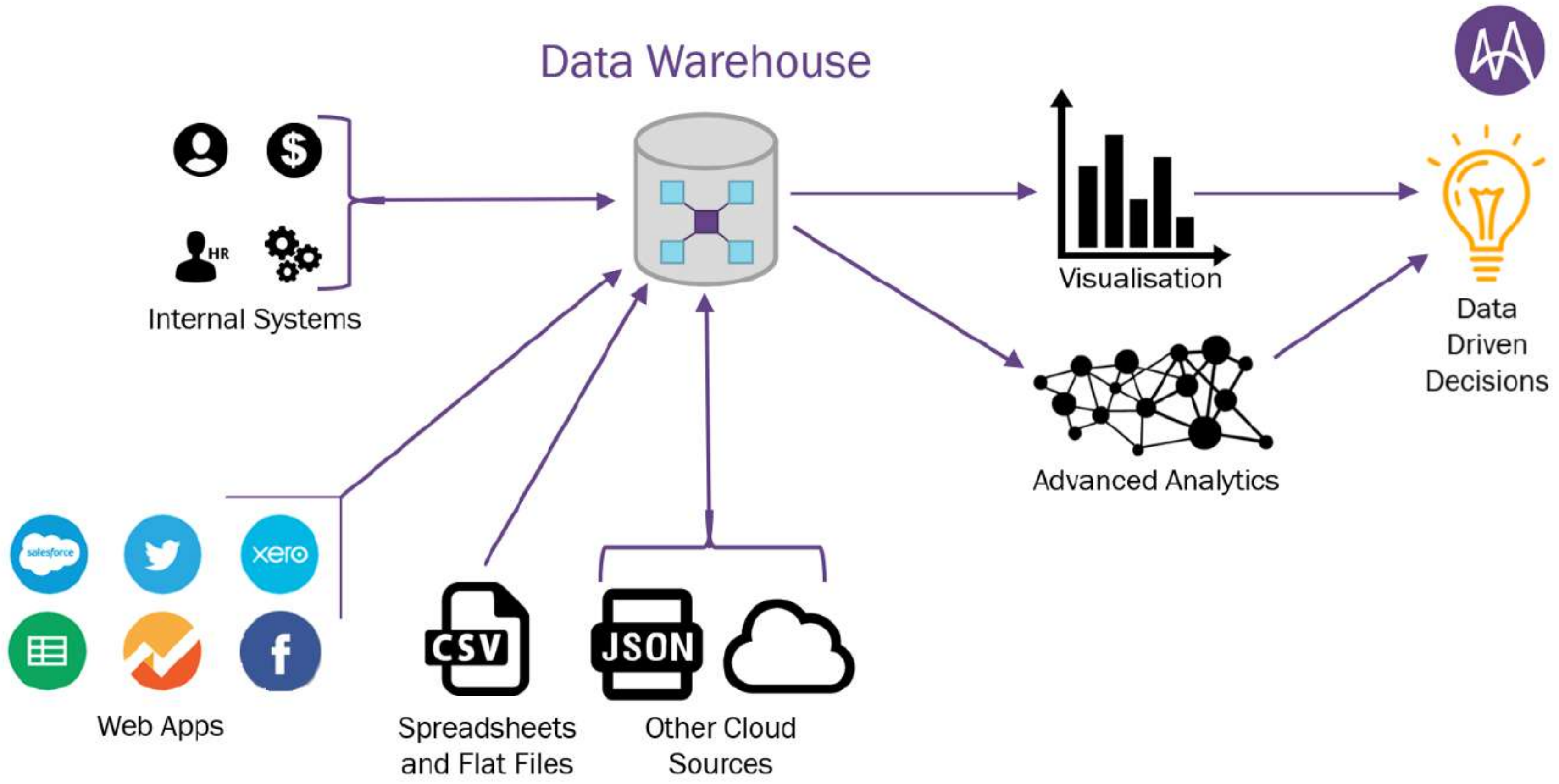


kaggle
Search
Home
Compete
Data
Notebooks
Discuss
Courses
Jobs
More
Datasets
Find and use datasets or complete tasks. Learn more.
Create Public Datasets
Open a dialogue, accept contributions, and get insights: improve your dataset by publishing it on Kaggle.
Create Public Dataset
Search 53,795 datasets
Feedback
Filter
Public
Sort by: Hottest
Health Insurance Cross Sell Prediction
Anmol Kumar
6 days 6 MB 10.0 3 Files (CSV) 1 Task
73
Healthcare Analytics
shivan kumar
4 days 2 MB 9.4 10 Files (CSV, other) 1 Task
37
arXiv Dataset
arXiv
Cornell University Link
3 days 885 MB 8.8 2 Files (JSON, other) 3 Tasks
533

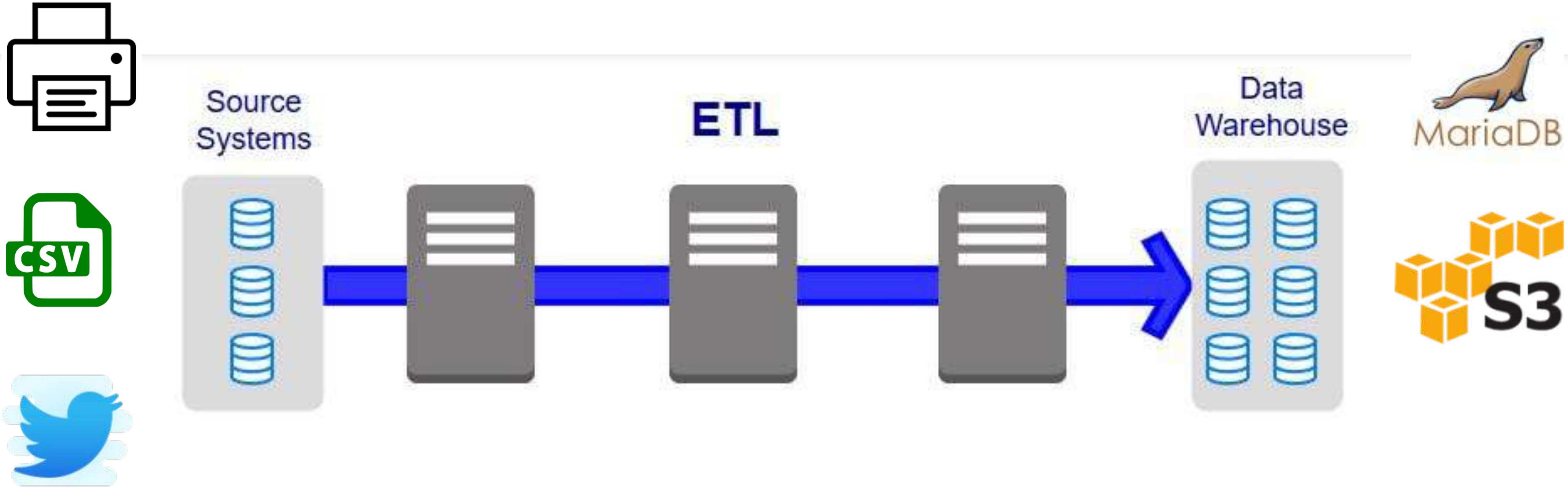
# Data Science Requirement



# Data Warehouse



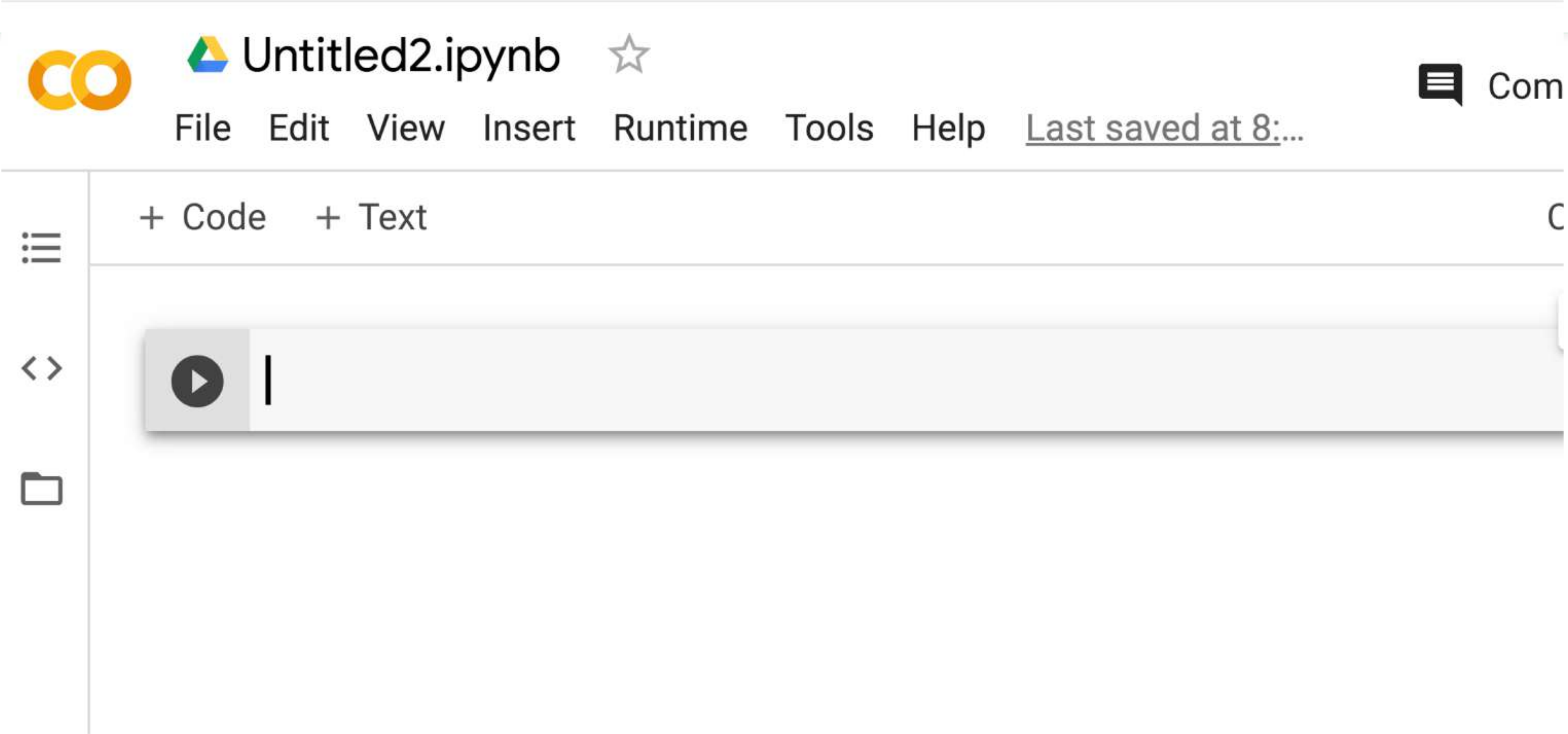
# Extract Transform Load



# Python-libraries for ETL

- Pandas (General Format / Tabular Data)
- OpenCV (Image)
- Librosa (Speech/Music)
- Scikit-learn (Text)
- Mysql connector

# Running python via colab Google



# Getting Started: Pandas

```
In [ ]: #Read csv file
        df = pd.read_csv(”/drive/My Drive/heart.csv")
```

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])

pd.read_stata('myfile.dta')

pd.read_sas('myfile.sas7bdat')

pd.read_hdf('myfile.h5','df')
```

# Exploring data frames

```
In [3]: df_data_1.head()
```

Out[3]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |

# Data Frames: Selecting rows

If we need to select a range of rows, we can specify the range using ":"

```
In [ ]: #Select rows by their position:
        df[10:20]
```

Notice that the first row has a position 0, and the last value in the range is omitted:
So for 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9

# Data Frames: method iloc

If we need to select a range of rows and/or columns, using their positions we can use method iloc:

In [ ]:
```
#Select rows by their labels:
df_sub.iloc[10:20,[0, 3, 4, 5]]
```

Out[ ]:

| | age | trestbps | chol | fbs |
|---|---|---|---|---|
| **10** | 54 | 140 | 239 | 0 |
| **11** | 48 | 130 | 275 | 0 |
| **12** | 49 | 130 | 266 | 0 |
| **13** | 64 | 110 | 211 | 0 |
| **14** | 58 | 150 | 283 | 1 |
| **15** | 50 | 120 | 219 | 0 |
| **16** | 58 | 120 | 340 | 0 |
| **17** | 66 | 150 | 226 | 0 |
| **18** | 43 | 150 | 247 | 0 |
| **19** | 69 | 140 | 239 | 0 |

# Exploratory Data Analysis

Exploratory Data Analysis (EDA) are very important steps in any analysis task.

get to know your data!

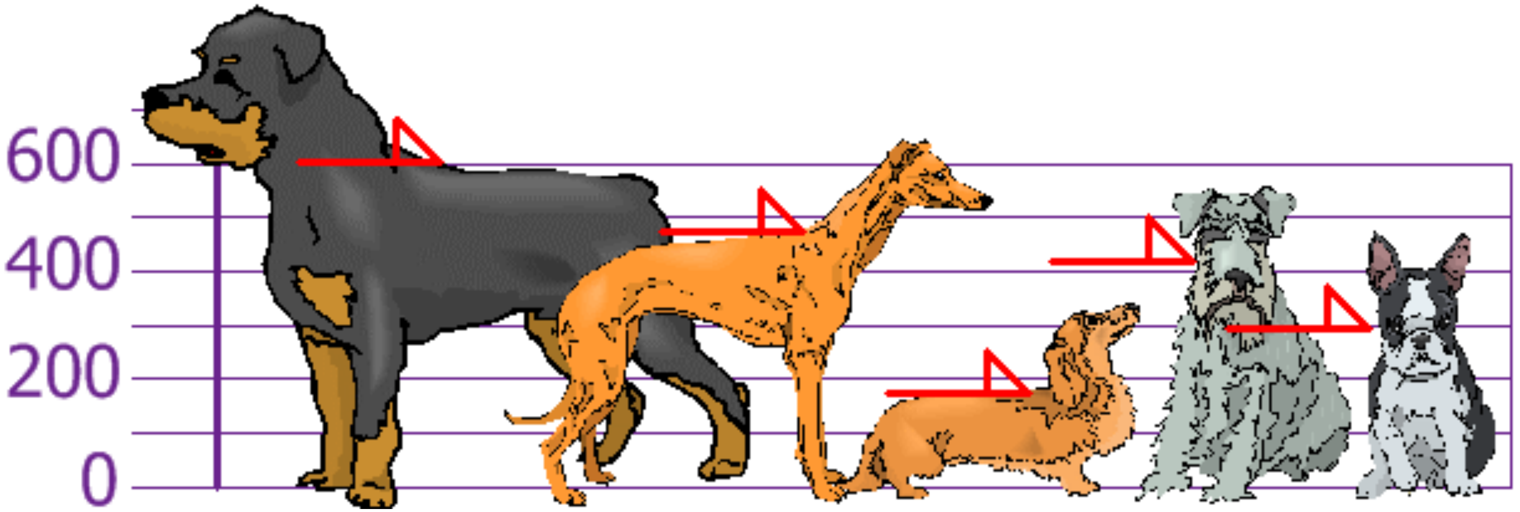
- distributions (symmetric, normal, skewed)
- data quality problems
- outliers
- correlations and inter-relationships
- subsets of interest
- suggest functional relationships

# Summary Statistics

- mean: $\mu = \sum_i X_i / n$
- mode: most common value in X
- median: $\mathbf{X}$=sort(X), median = $\mathbf{X}_{n/2}$ (half below, half above)
- variance: $\sigma^2 = \sum_i (X_i - \mu)^2 / n$

| | Unnamed: 0 | symboling | normalized-losses | wheel-base | length | width | height | curb-weight | engine-size | bore | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 201.000000 | 201.000000 | 164.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.00000C |
| mean | 100.000000 | 0.840796 | 122.000000 | 98.797015 | 174.200995 | 65.889055 | 53.766667 | 2555.666667 | 126.875622 | 3.319154 | 3.256766 |
| std | 58.167861 | 1.254802 | 35.442168 | 6.066366 | 12.322175 | 2.101471 | 2.447822 | 517.296727 | 41.546834 | 0.280130 | 0.316049 |
| min | 0.000000 | -2.000000 | 65.000000 | 86.600000 | 141.100000 | 60.300000 | 47.800000 | 1488.000000 | 61.000000 | 2.540000 | 2.070000 |
| 25% | 50.000000 | 0.000000 | NaN | 94.500000 | 166.800000 | 64.100000 | 52.000000 | 2169.000000 | 98.000000 | 3.150000 | 3.110000 |
| 50% | 100.000000 | 1.000000 | NaN | 97.000000 | 173.200000 | 65.500000 | 54.100000 | 2414.000000 | 120.000000 | 3.310000 | 3.290000 |
| 75% | 150.000000 | 2.000000 | NaN | 102.400000 | 183.500000 | 66.600000 | 55.500000 | 2926.000000 | 141.000000 | 3.580000 | 3.410000 |
| max | 200.000000 | 3.000000 | 256.000000 | 120.900000 | 208.100000 | 72.000000 | 59.800000 | 4066.000000 | 326.000000 | 3.940000 | 4.170000 |

# Mean

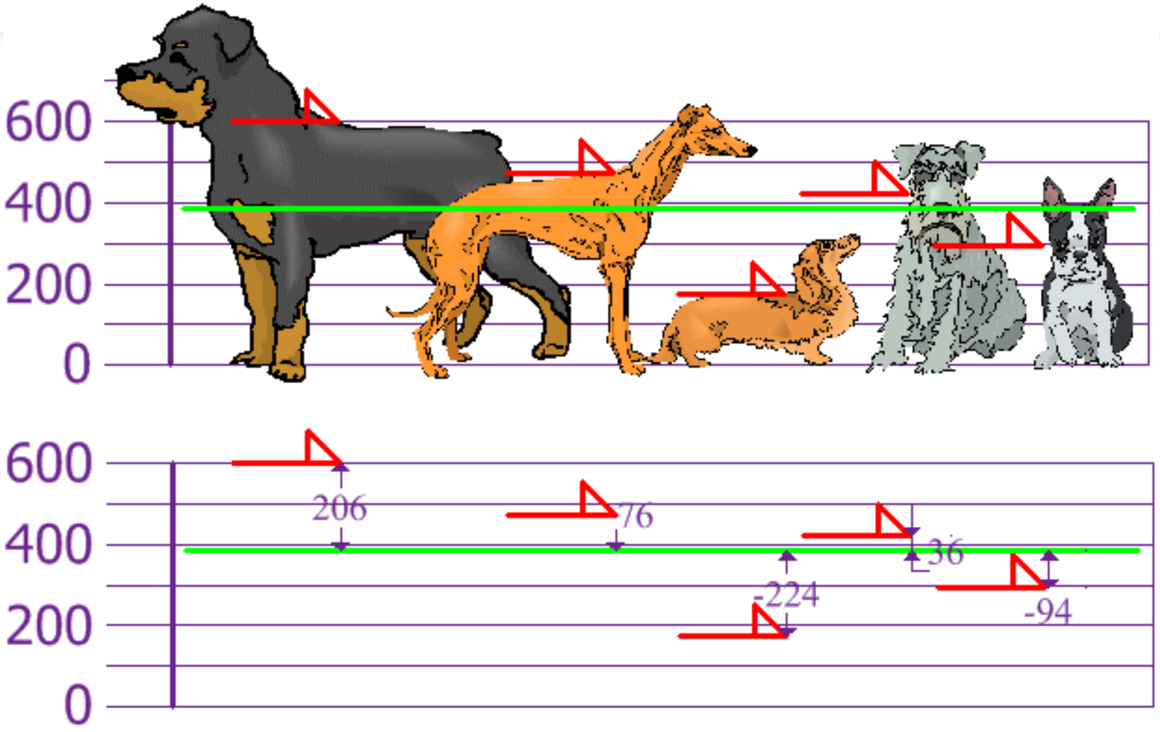



$$\text{Mean} = \frac{600 + 470 + 170 + 430 + 300}{5}$$

$$= \frac{1970}{5}$$

$$= 394$$

# Variance




**Variance**

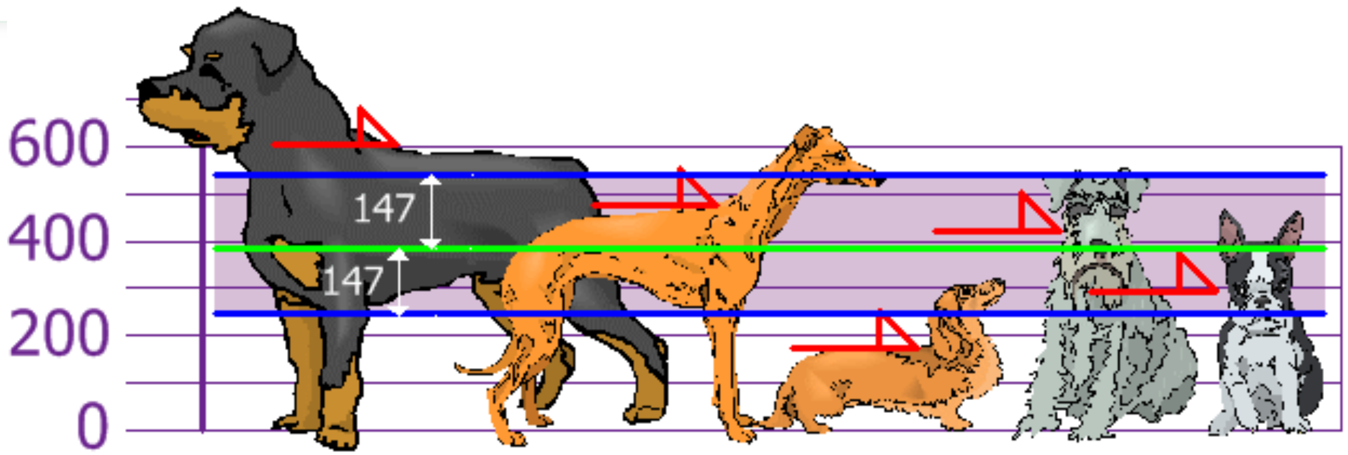
$$\sigma^2 = \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5}$$

$$= \frac{42436 + 5776 + 50176 + 1296 + 8836}{5}$$

$$= \frac{108520}{5}$$

$$= 21704$$

# Standard deviation




**Standard Deviation**

$\sigma = \sqrt{21704}$

$= 147.32...$

$= \mathbf{147}$ *(to the nearest mm)*

# Data Normalization

- Simple Feature Scaling
- Min-Max
- Z-score

| age | income |
|---|---|
| 20 | 100000 |
| 30 | 20000 |
| 40 | 500000 |

**Not-normalized**

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

| age | income |
|---|---|
| 0.2 | 0.2 |
| 0.3 | 0.04 |
| 0.4 | 1 |

**Normalized**

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

# Data Normalization: Simple Feature Scaling

- The first method, called “simple feature scaling”, just divides each value by the maximum value for that feature.
- This makes the new values range between 0 and 1.

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

# Data Normalization: Min Max Scaling

- The second method, called “Min-Max”, takes each value, X_old, subtracted from the minimum value of that feature, then divides by the range of that feature.
- Again, the resulting new values range between 0 and 1.

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

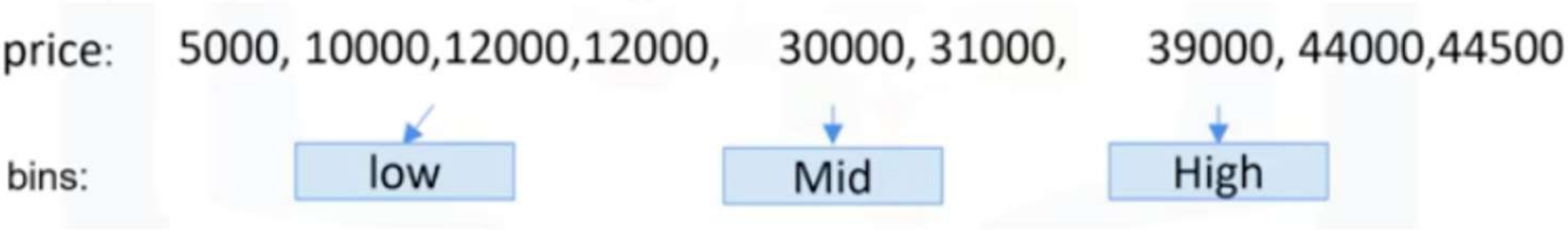
# Data Normalization: Z Score

- The third method is called “z-score” or “standard score”. In this formula, for each value, you subtract the Mu which is the average of the feature, and then divide by the standard deviation (sigma).
- The resulting values hover around 0, and typically range between -3 and +3, but can be higher or lower.

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

# Binning

- Binning is when you group values together into “bins”
- Convert numeric into categorical variables
- For example, you can bin “age” into [0 to 5], [6 to 10], [11 to 15] a



# Binning Python

```
bins = np.linspace(min(df["price"]), max(df["price"]), 4)
```

Start

Stop

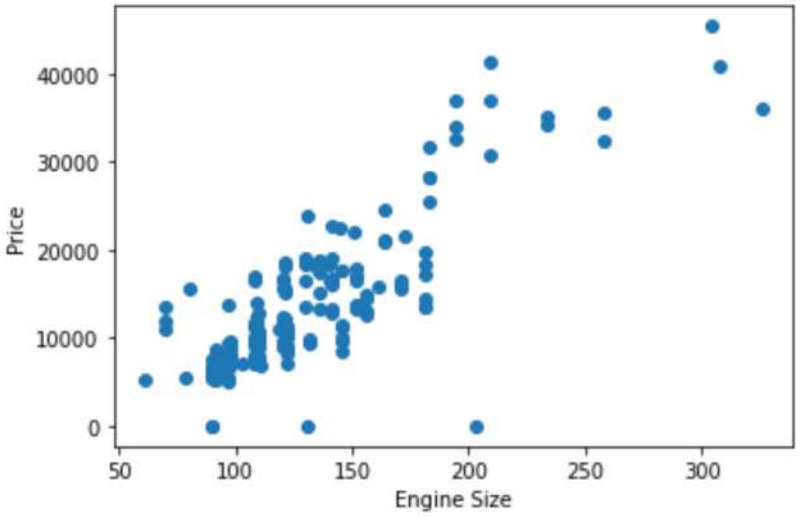
Number of dividers

```
group_names = ["Low", "Medium", "High"]
```

```
df["price-binned"] = pd.cut(df["price"], bins, labels=group_names, include_lowest=True )
```

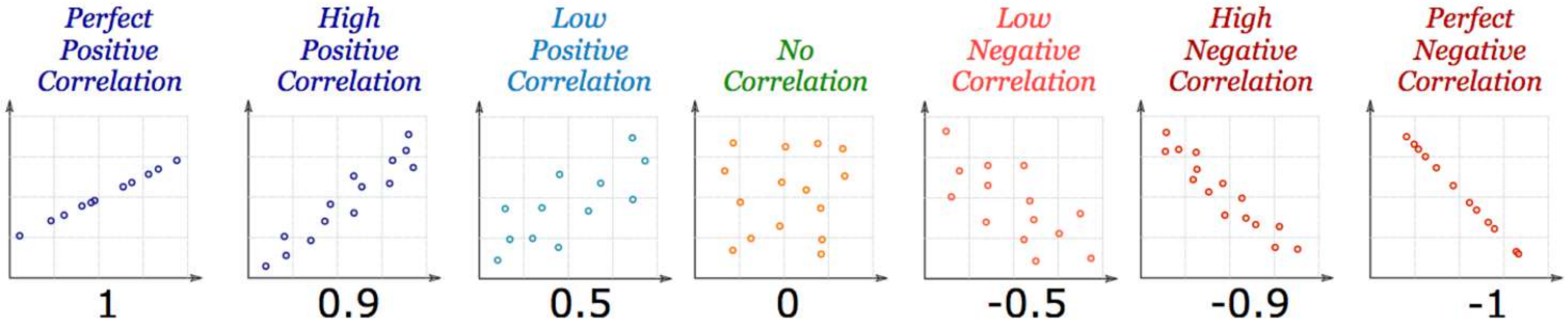
# Scatter Plot

- This is giving us an initial indication that there is a positive linear relationship between these two variables.



# Correlation

- Correlation is **Positive** when the values **increase** together, and
- Correlation is **Negative** when one value **decreases** as the other increases

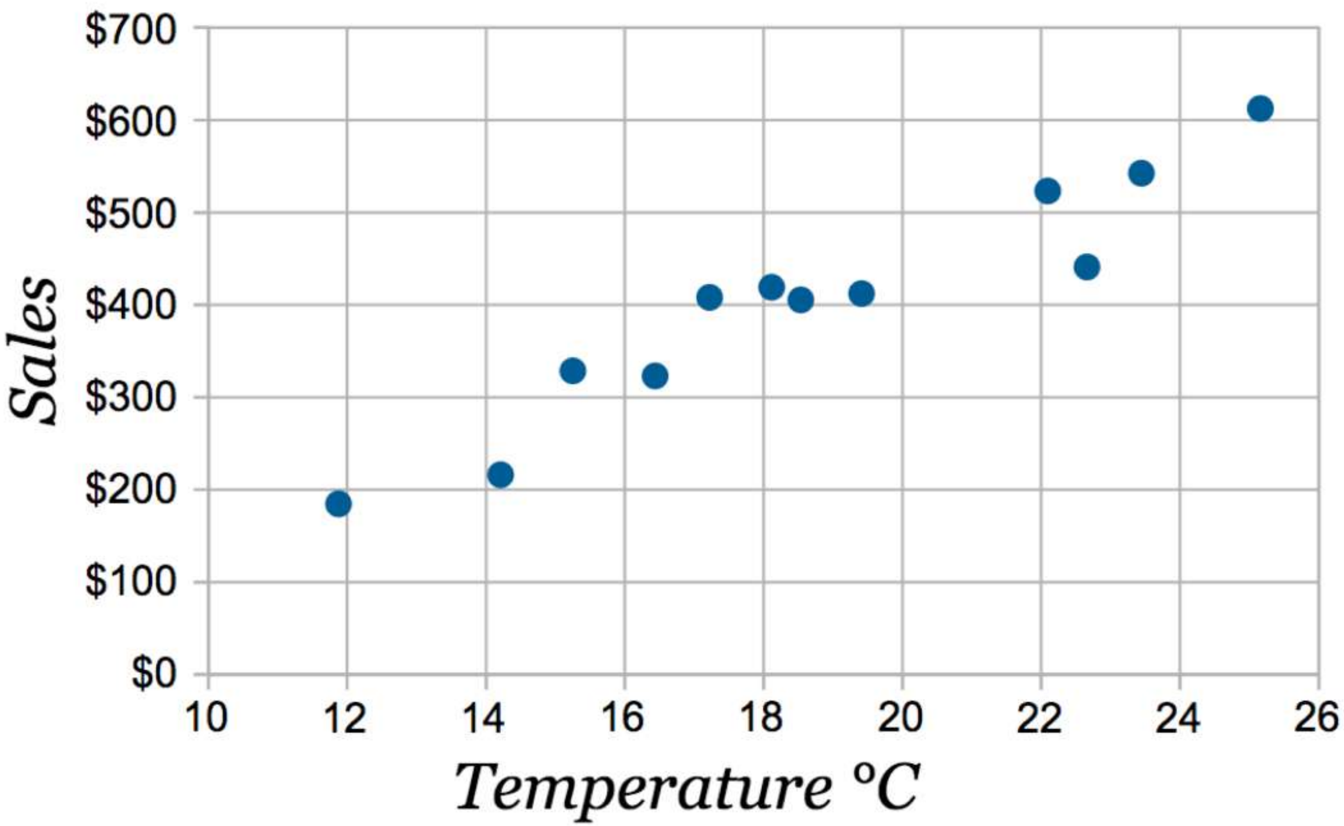


# Correlation

- 1 is a perfect positive correlation
- 0 is no correlation (the values don't seem linked at all)
- -1 is a perfect negative correlation

# Case Study

| Ice Cream Sales vs Temperature | |
|---|---|
| Temperature °C | Ice Cream Sales |
| 14.2° | $215 |
| 16.4° | $325 |
| 11.9° | $185 |
| 15.2° | $332 |
| 18.5° | $406 |
| 22.1° | $522 |
| 19.4° | $412 |
| 25.1° | $614 |
| 23.4° | $544 |
| 18.1° | $421 |
| 22.6° | $445 |
| 17.2° | $408 |



# Case Study: Pearson correlation

- Step 1: Find the mean of **x**, and the mean of **y**
- Step 2: Subtract the mean of x from every x value (call them "**a**"), and subtract the mean of y from every y value (call them "**b**")
- Step 3: Calculate: **ab**, $\mathbf{a^2}$ and $\mathbf{b^2}$ for every value
- Step 4: Sum up **ab**, sum up $\mathbf{a^2}$ and sum up $\mathbf{b^2}$
- Step 5: Divide the sum of ab by the square root of [(sum of $a^2$) × (sum of $b^2$)]

# Case Study: Pearson correlation

**2** *Subtract Mean*

**3** *Calculate ab, $a^2$ and $b^2$*

| Temp °C | Sales | "a" | "b" | a×b | $a^2$ | $b^2$ |
|---|---|---|---|---|---|---|
| 14.2 | $215 | -4.5 | -$187 | 842 | 20.3 | 34,969 |
| 16.4 | $325 | -2.3 | -$77 | 177 | 5.3 | 5,929 |
| 11.9 | $185 | -6.8 | -$217 | 1,476 | 46.2 | 47,089 |
| 15.2 | $332 | -3.5 | -$70 | 245 | 12.3 | 4,900 |
| 18.5 | $406 | -0.2 | $4 | -1 | 0.0 | 16 |
| 22.1 | $522 | 3.4 | $120 | 408 | 11.6 | 14,400 |
| 19.4 | $412 | 0.7 | $10 | 7 | 0.5 | 100 |
| 25.1 | $614 | 6.4 | $212 | 1,357 | 41.0 | 44,944 |
| 23.4 | $544 | 4.7 | $142 | 667 | 22.1 | 20,164 |
| 18.1 | $421 | -0.6 | $19 | -11 | 0.4 | 361 |
| 22.6 | $445 | 3.9 | $43 | 168 | 15.2 | 1,849 |
| 17.2 | $408 | -1.5 | $6 | -9 | 2.3 | 36 |
| **18.7** | **$402** | | | **5,325** | **177.0** | **174,757** |

**1** *Calculate Means*

**4** *Sum Up*

**5** $$\frac{5,325}{\sqrt{177.0 \times 174,757}} = 0.9575$$

# Case: Prediksi Harga Mobil

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

## Automobile Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: From 1985 Ward's Automotive Yearbook

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 205 | **Area:** | N/A |
| **Attribute Characteristics:** | Categorical, Integer, Real | **Number of Attributes:** | 26 | **Date Donated** | 1987-05-19 |
| **Associated Tasks:** | Regression | **Missing Values?** | Yes | **Number of Web Hits:** | 544484 |

https://archive.ics.uci.edu/ml/datasets/automobile

# Dataset

| | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | ? | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 |
| **1** | 3 | ? | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 |
| **2** | 1 | ? | alfa-romero | gas | std | two | hatchback | rwd | front | 94.5 | 171.2 | 65.5 | 52.4 |
| **3** | 2 | 164 | audi | gas | std | four | sedan | fwd | front | 99.8 | 176.6 | 66.2 | 54.3 |
| **4** | 2 | 164 | audi | gas | std | four | sedan | 4wd | front | 99.4 | 176.6 | 66.4 | 54.3 |

```
import pandas as pd #lib pandas:

df = pd.read_csv("/content/drive/My Drive/dataset/mobil/data.data",
df
```

# Pre-Processing

```
df['price'] = df['price'].replace(to_replace ="?",value =df['price'].mean())
df['price'] = df['price'].astype("int64") #jadikan object --> int64
```

```
#simple normalization
df['length'] = df['length'] / df['length'].max()
```

```
#Min Max normalization
df['wheel-base'] = (df['wheel-base'] - df['wheel-base'].min() / df['wheel-base'].max() - df['wheel-base'].min())
```

```
#z score
df['width'] = (df['width'] - df['wheel-base'].mean()) / df['width'].std()
```

# Pre-Processing

```
import numpy as np #numeric

print("MAX: " + str(df_new['compression-ratio'].max()))
print("MIN: " + str(df_new['compression-ratio'].min()))

bins = np.linspace(min(df['compression-ratio']), max(df['compression-ratio']), 4)
group_names = ["low","medium","high"]
df["compression-ratio"] = pd.cut(df["compression-ratio"], bins, labels=group_names, include_lowest=True)
```

# Statistics Descriptive

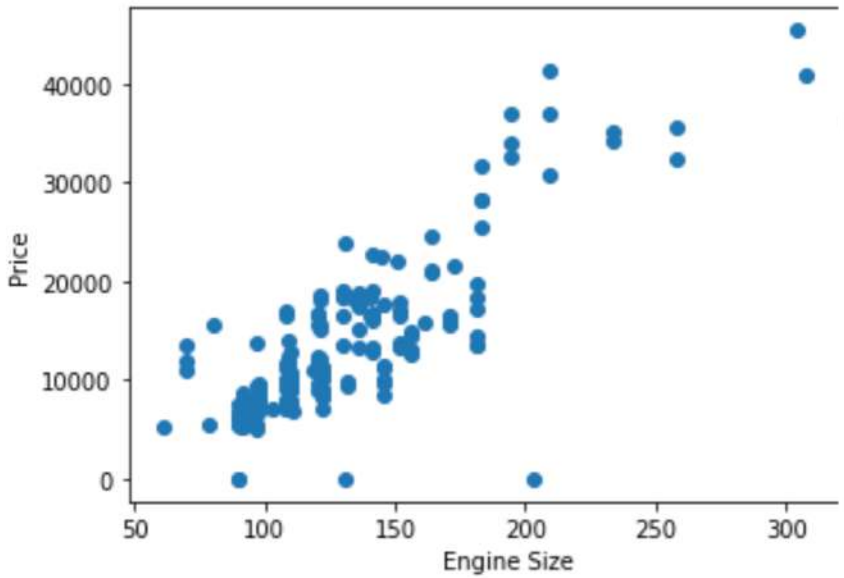
```
df.describe()
```

| | 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 | 2 |
| **mean** | 0.834146 | 98.756585 | 174.049268 | 65.907805 | 53.724878 | 2555.565854 | 126.907317 | 10.142537 | 25.219512 | 30.751220 | 129 |
| **std** | 1.245307 | 6.021776 | 12.337289 | 2.145204 | 2.443522 | 520.680204 | 41.642693 | 3.972040 | 6.542142 | 6.886443 | 80 |
| **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 | 76 |
| **50%** | 1.000000 | 97.000000 | 173.200000 | 65.500000 | 54.100000 | 2414.000000 | 120.000000 | 9.000000 | 24.000000 | 30.000000 | 101 |
| **75%** | 2.000000 | 102.400000 | 183.100000 | 66.900000 | 55.500000 | 2935.000000 | 141.000000 | 9.400000 | 30.000000 | 34.000000 | 165 |
| **max** | 3.000000 | 120.900000 | 208.100000 | 72.300000 | 59.800000 | 4066.000000 | 326.000000 | 23.000000 | 49.000000 | 54.000000 | 454 |

# Scatter Plot

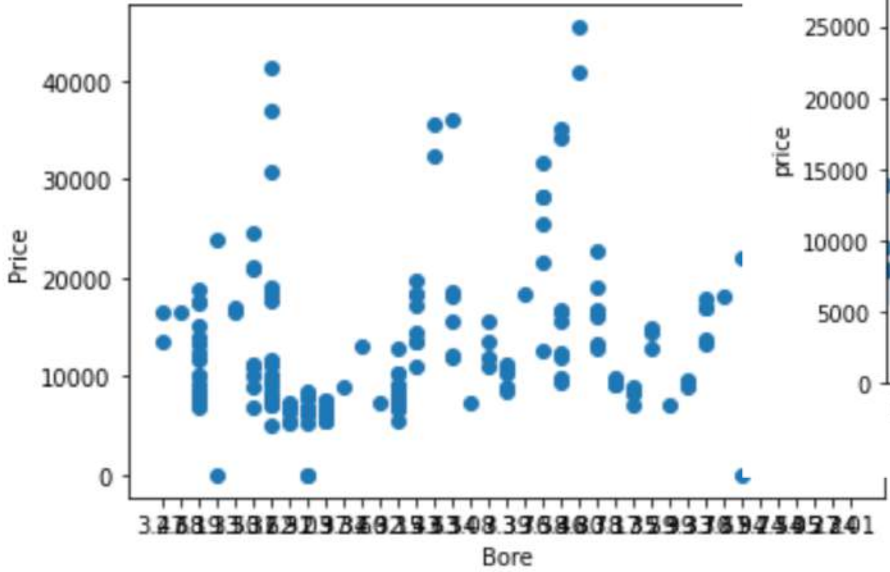
```python
import matplotlib.pyplot as plt

plt.scatter(df['engine-size'], df['price'])
plt.xlabel("Engine Size")
plt.ylabel("Price")
plt.show()
```



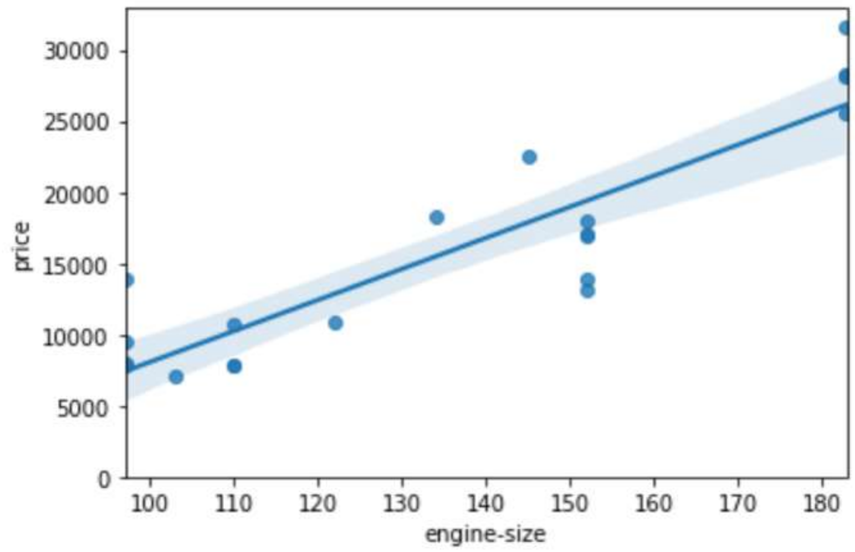

```python
import matplotlib.pyplot as plt

plt.scatter(df['bore'], df['price'])
plt.xlabel("Bore")
plt.ylabel("Price")
plt.show()
```




```python
sns.regplot(x='engine-size',y='price',data=df_new)
plt.ylim(0,)
```

```
(0.0, 32915.73541112133)
```



# Pearson Correlation

```
import scipy

x, y = scipy.stats.pearsonr(df['engine-size'],df['price'])
print("pearson Coef: " + str(x))
print("p value: " + str(y))
```

```
pearson Coef: 0.838097285838633
p value: 2.489808772739823 7e-55
```

```
df['bore'] = df['bore'].replace(to_replace ="?",value ="0")
df['bore'] = df['bore'].astype("float")

x, y = scipy.stats.pearsonr(df['bore'],df['price'])
print("pearson Coef: " + str(x))
print("p value: " + str(y))
```

```
pearson Coef: 0.264096030122132 1
p value: 0.000130075990655641 13
```

# References


- https://medium.com/wripolinema/yuk-kenalan-dengan-data-science-dalam-pengolahan-suatu-data-aa1141c10a43
- Bisa.ai
- Digitalent Scholarship Kominfo course Artificial Intelligence 2019