

# Lecture 6: Practice Data Pre-processing & EDA

(Data Cleaning, Data Wrangling)



Lecture 6: Practice Data Pre-processing & EDA

# Section 1: Data Pre-processing


(Data Cleaning, Data Wrangling)



# Missing Values



- Missing values occur when no data value is stored for a variable (feature) in an observation
- Could be represented as “?”, “N/A”, 0 or just a blank cell

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	s
0	3		alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	

# How to deal with missing data?



- Check with the data collection source
- Drop the missing values:
  - Drop the variable (feature) – Drop column
  - Drop the data entry – Drop row
- Replacing 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 to drop missing values with Pandas

- Use `df.dropna()` :

highway-mpg	price
...	...
20	23875
22	NaN
29	16430
...	...



highway-mpg	price
...	...
20	23875
29	16430
...	...


- `axis=0` drops the entire **row**
- `axis=1` drops the entire **column**

- **Syntax:** `df.dropna(subset=["price"], axis=0, inplace=True)` :

# How to replace missing values with Pandas

- Use `df.replace(missing_value, new_value)` :

normalized-losses	make
...	...
164	audi
164	audi
NaN	audi
158	audi
...	...



normalized-losses	make
...	...
164	audi
164	audi
162	audi
158	audi
...	...

- `mean = df["normalized-losses"].mean()`
- `df["normalized-losses"].replace(np.nan, mean)`

# How to deal with missing data?



- Check with the data collection source
- Drop the missing values:
  - Drop the variable (feature) – Drop column
  - Drop the data entry – Drop row
- Replacing 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

# Data Normalization

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 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

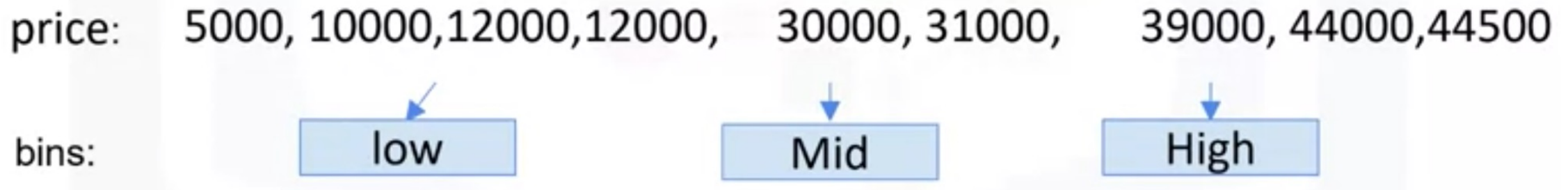
③

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

Z-score

# Data Binning

- Binning: Grouping of values into “bins”
- Converts numeric into categorical variables
- Group a set of numerical values into a set of “bins”
- E.g.: “price” is a feature range from 5,000 to 45,500. => In order to have a better representation of price:



# Binning with Pandas

price
13495
16500
18920
41315
5151
6295
...



price	price-binned
13495	Low
16500	Low
18920	Medium
41315	High
5151	Low
6295	Low
...	...

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

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

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

Lecture 6: Practice Data Pre-processing & EDA

## Section 2: Exploratory Data Analysis (EDA)



# Exploratory Data Analysis (EDA)



## **Preliminary step in data analysis to:**

- Summarize main characteristics of the data
- Gain better understanding of the dataset
- Uncover relationships between variables
- Extract important variables

**Topics:** Descriptive Statistics, GroupBy, Correlation

# Descriptive Statistics



- Describe basic features of data
- Giving short summaries about the sample and measures of the data

# Descriptive Statistics – Describe()



- Summarize statistics using pandas **describe()** method: `df.describe()`

	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	201.000000
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

# Descriptive Statistics – value\_counts()

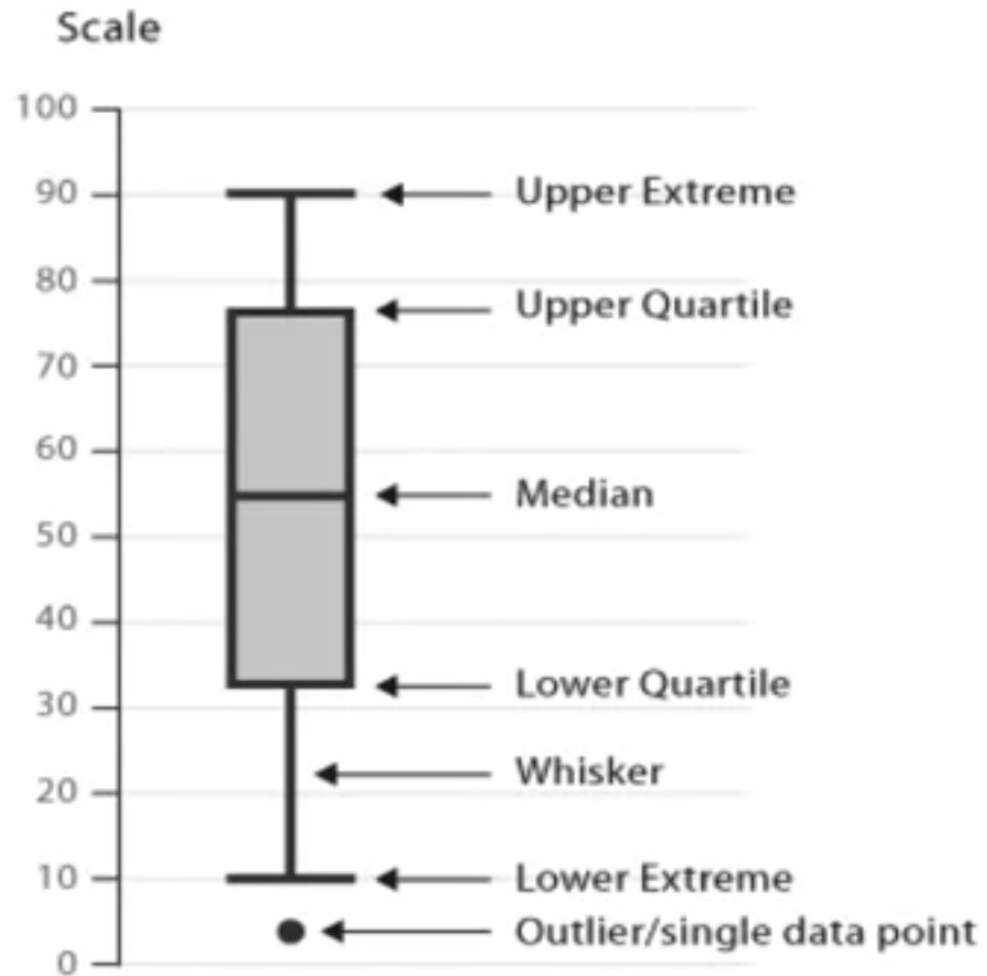
- Summarize the categorical data by using **value\_counts()** method:

```
drive_wheels_counts=df["drive-wheels"].value_counts().to_frame()  
  
drive_wheels_counts.rename(columns={'drive-wheels':'value_counts'}, inplace=True)  
drive_wheels_counts
```

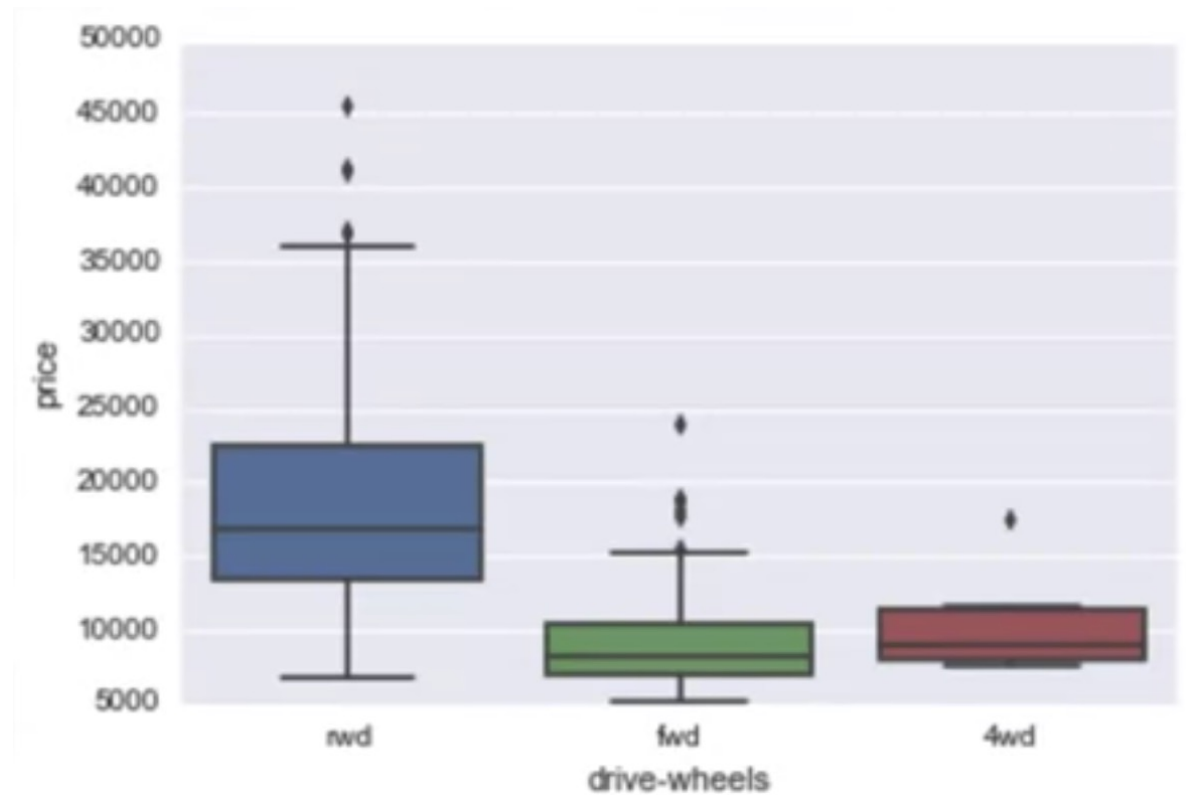
	value_counts
drive-wheels	
fwd	118
rwd	75
4wd	8



# Descriptive Statistics – Box plots



```
sns.boxplot(x= "drive-wheels", y= "price", data=df)
```

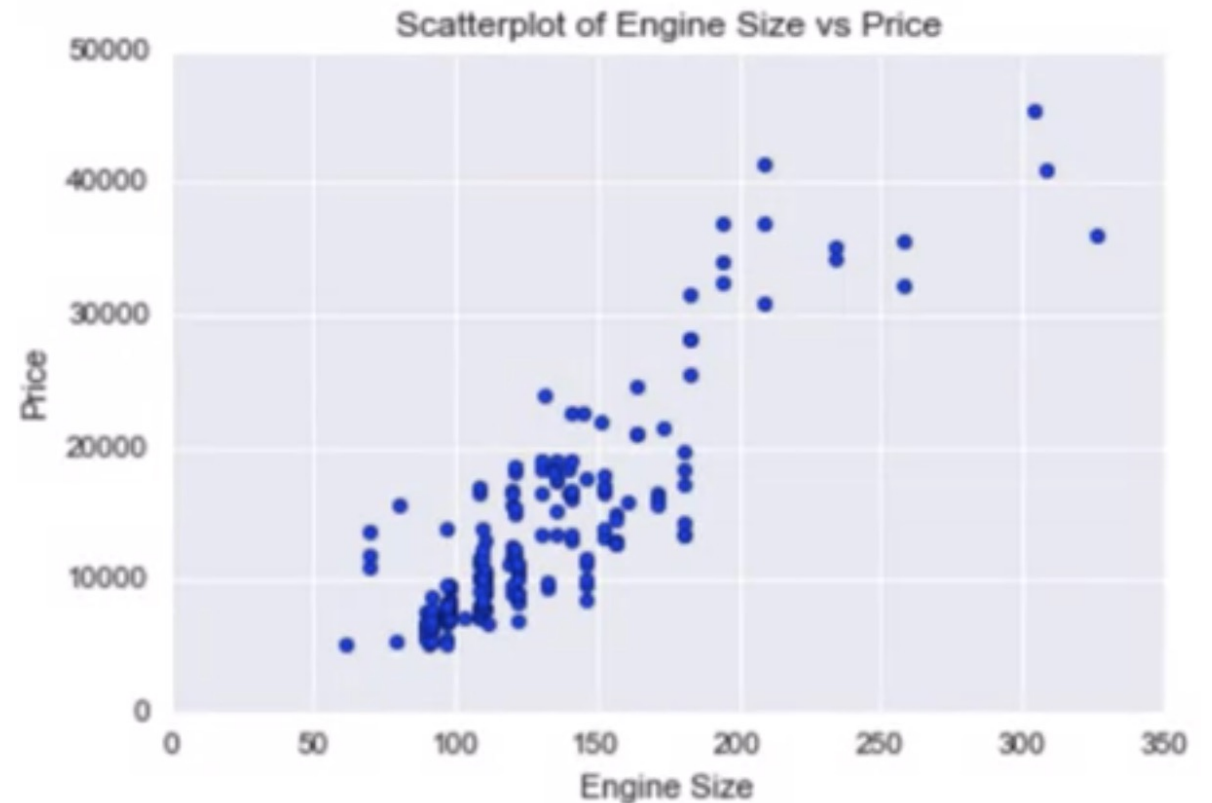


# Descriptive Statistics – Scatter plots



- We can draw scatter plot to show the relationship between two variables:

```
y=df["price"]  
x=df["engine-size"]  
plt.scatter(x,y)  
  
plt.title("Scatterplot of Engine Size vs Price")  
plt.xlabel("Engine Size")  
plt.ylabel("Price")
```



# Correlation



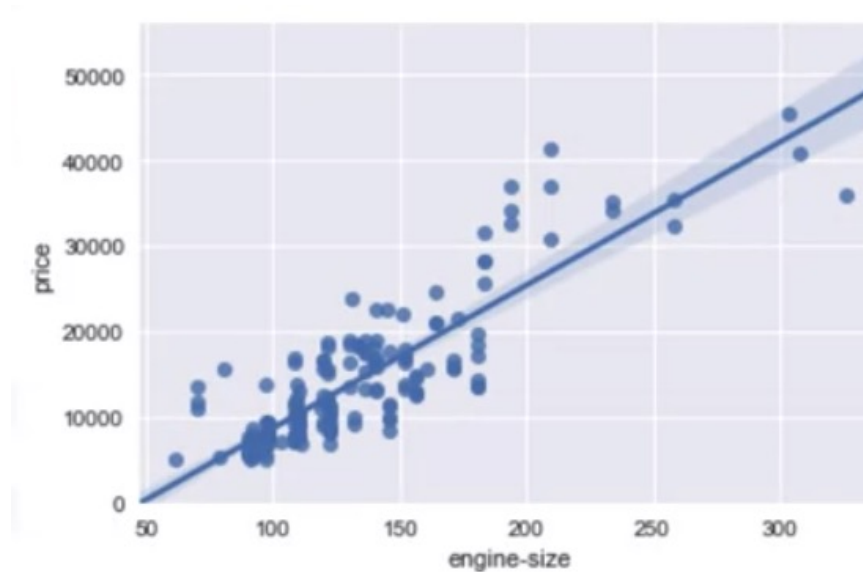
What is correlation?:

- Measures to what extent different variables are interdependent
- For example:
  - Lung cancer -> Smoking
  - Rain -> Umbrella
- In DA, we often deal with correlation

# Correlation – Positive Linear Relationship

Correlation between two features (engine-size and price)

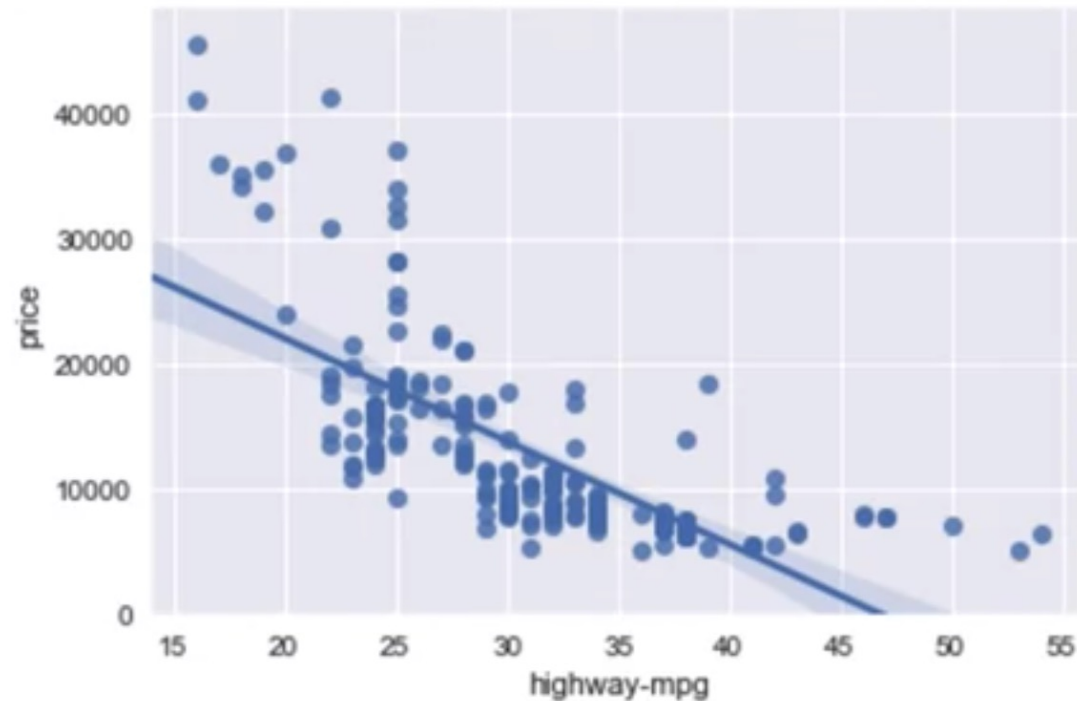
```
sns.regplot(x="engine-size", y="price", data=df)  
plt.ylim(0,)
```



# Correlation – Negative Linear Relationship

Correlation between two features (highway-mpg and price)

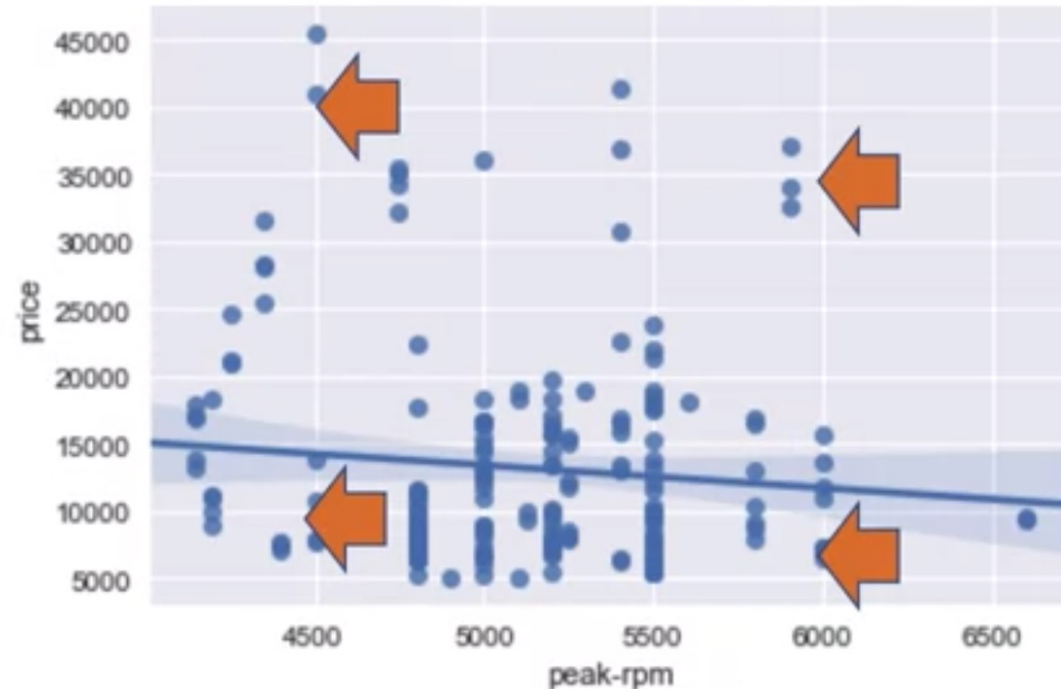
```
sns.regplot(x="highway-mpg", y="price", data=df)  
plt.ylim(0, )
```



# Correlation – Week correlation

Week correlation between two features (peak-mpg and price)

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



# Correlation Statistics– 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 relations
- P-value:
  - $< 0.001$  Strong certainty in the result
  - $< 0.05$  Moderate certainty in the result
  - $< 0.1$  Weak certainty in the result
  - $> 0.1$  No certainty in the result
- Strong Correlation:
  - Correlation coefficient close to 1 or -1
  - P value less than 0.001

# Pearson Correlation – Example

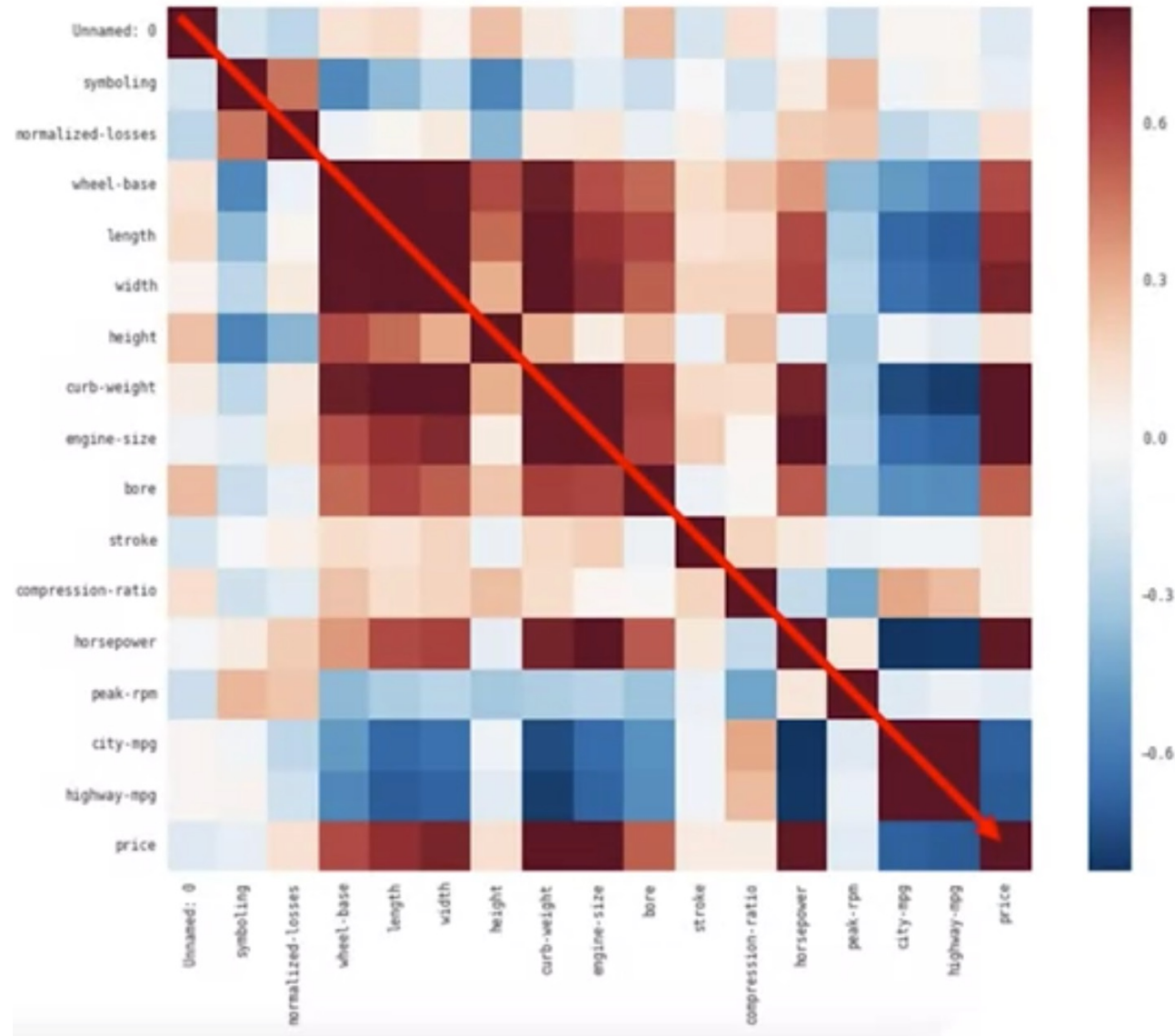


```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
```

- Pearson correlation: 0.81
- P-value: 9.35e-48



# Pearson Correlation – Heatmap



# Summary



In summary, in this lecture, you learned:

- How to do data-preprocessing
- How to perform an EDA on a given dataset