DA Capstone (AutoScout) Intro_new_DE

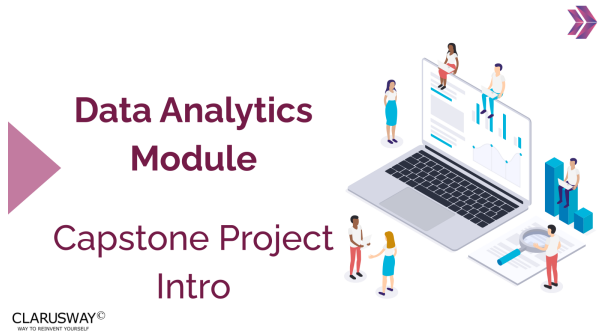
Pear Deck Session
Training Clarusway
Pear Deck - March 10, 2023 at 10:47AM

Part 1 - Summary

Use this space to summarize your thoughts on the lesson

Part 2 - Responses

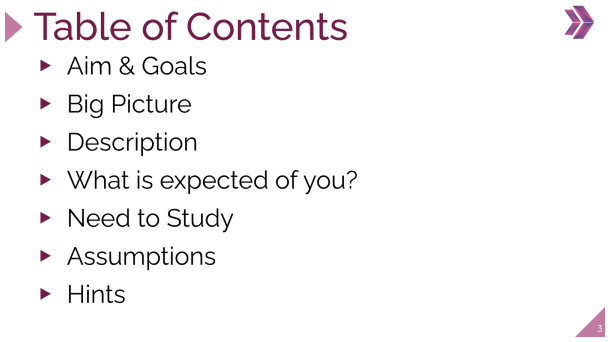
Slide 1



Use this space to take notes:

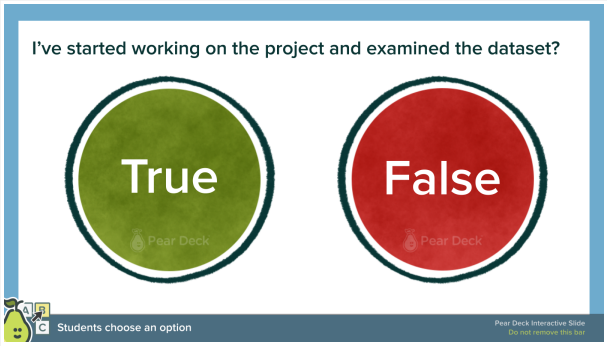
Slide 2


Use this space to take notes:

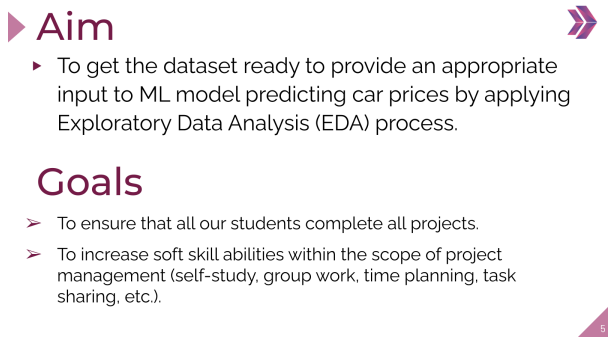
Slide 3


Use this space to take notes:

Slide 4	Your Response
	You Chose

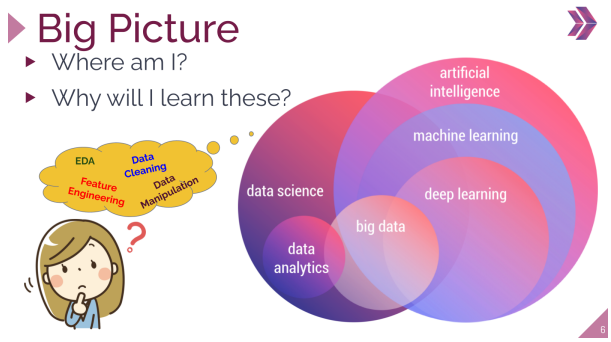
Slide 4	Your Response
	• **False** Other Choices • True

Use this space to take notes:

Slide 5


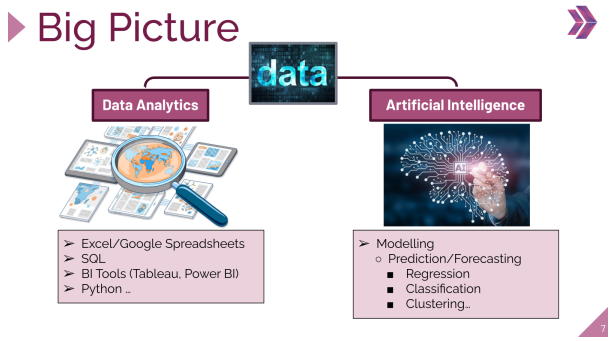
Use this space to take notes:

Slide 6



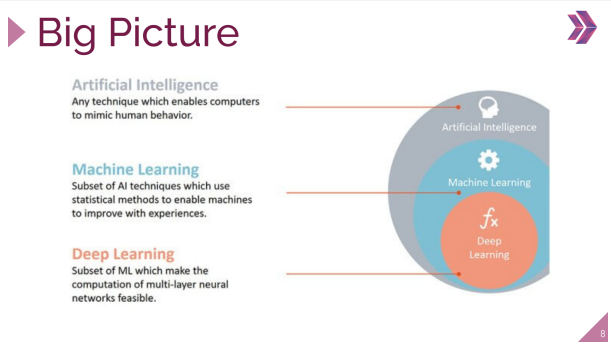
Use this space to take notes:

Slide 7



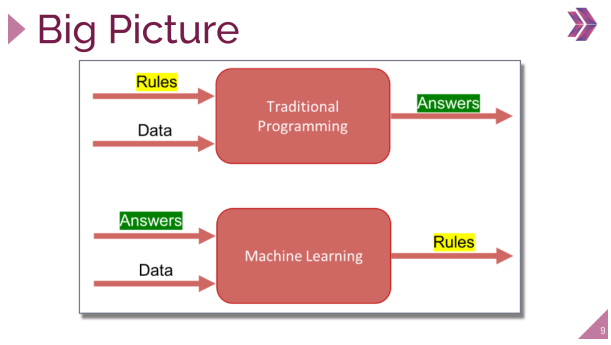
Use this space to take notes:

Slide 8



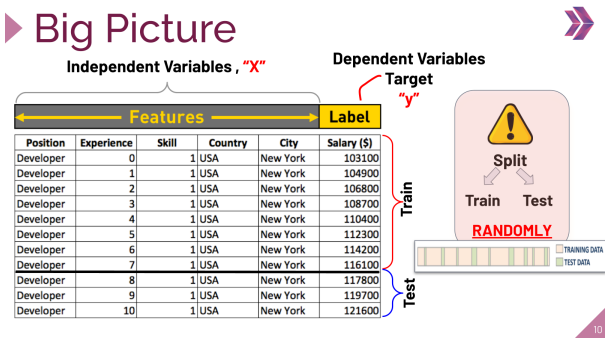
Use this space to take notes:

Slide 9



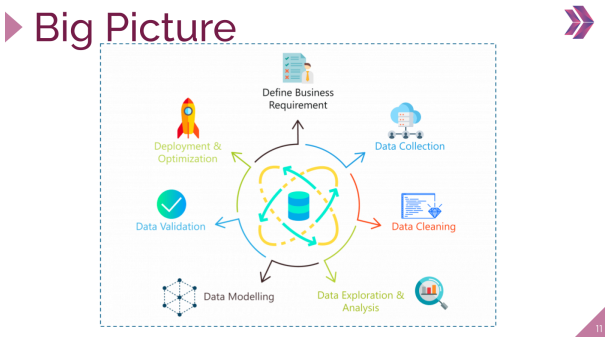
Use this space to take notes:

Slide 10



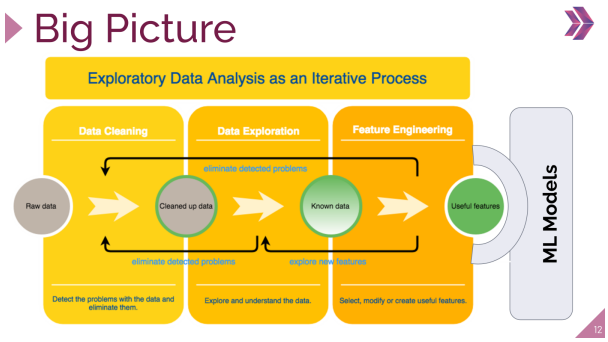
Use this space to take notes:

Slide 11



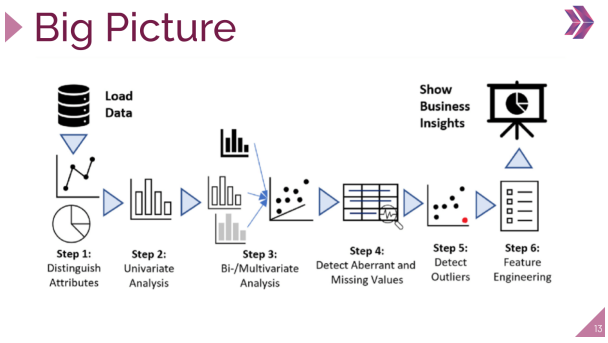
Use this space to take notes:

Slide 12



Use this space to take notes:

Slide 13



Use this space to take notes:

Slide 14

Description

- A ``.json`` file containing a dataset consisting of **29480 rows and 58 columns** is provided.
- This dataset, scraped from the online car trading company in 2022, contains many features of **various number of car models**.
- The features (variables) of this dataset are **too messy and distorted**.

14

Use this space to take notes:

Slide 15

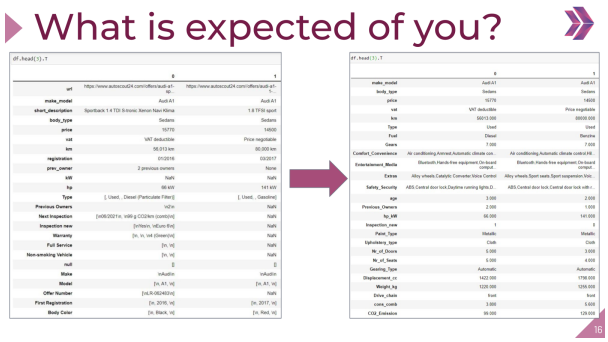
What is expected of you?

- Read the "**.json**" file and assign the dataset into a "**DataFrame**" using "pandas".
- Implement all aspects of the "**EDA process**" to the dataset.
  - Fix corrupted **data formats**
  - Handle with **missing values and outliers**
    - **Domain knowledge** (automobiles) is important
    - Always use the **internet** to do the research that you need (Domain Knowledge)
    - Think carefully to decide whether a data is **outliers or not**
  - **Drop the columns/rows** you determined unnecessary as a result of your analysis
  - Use **visualization tools** while doing all these processes

15

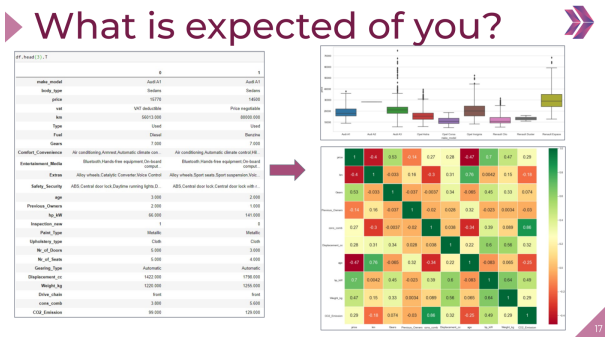
Use this space to take notes:

Slide 16



Use this space to take notes:

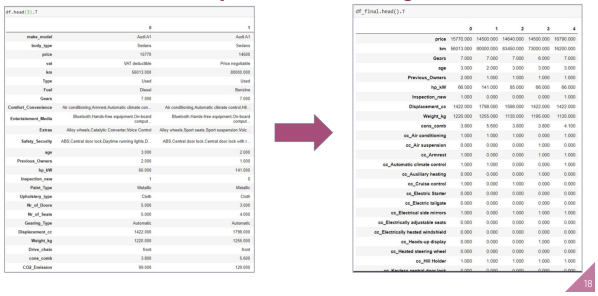
Slide 17



Use this space to take notes:

Slide 18

What is expected of you?



18

Use this space to take notes:

Slide 19

Need to Study

- str.method
- contains()
- extract()
- get_dummies()
- add_prefix()
- sample()
- to_numeric()
- isin()
- apply()
- replace()
- split()
- join()
- regex
- def
- lambda

19

Use this space to take notes:

Slide 20

Assumptions

- Assume the year you are currently in is 2022

20

Use this space to take notes:

Slide 21

Hints

- Domain Knowledge is one of the most important things to evaluate your data.
- You have to evaluate each column by target label.

21

Use this space to take notes:

Slide 22

Hints

- Check the column names.
  (You can change the column names to something more useful.)
- Check the percentage of null values for each column.
  (You can drop columns having more than %... null value.)
- Check the value_counts of each column, evaluate them and take notes about what you'll do.
  (drop, similarity between columns, how to clean, define the pattern etc.)

22

Use this space to take notes:

Slide 23

Hints

- How to exclude each value of columns from list.

```
\n1\n                                                                                                  8101
NaN                                                                                                    6640
\n2\n                                                                                                   766
\n0\n                                                                                                   163
\n3\n                                                                                                    17
                                                                                                       ...
[\n1\n, \n96 g CO2/km (comb)\n]                                                                          1
[\n1\n, \n181 g CO2/km (comb)\n]                                                                         1
[\n1\n, \n, 6 l/100 km (comb), \n, 8 l/100 km (city), \n, 4.9 l/100 km (country), \n]                    1
[\n1\n, \n, 6.7 l/100 km (comb), \n, 8.6 l/100 km (city), \n, 5.6 l/100 km (country), \n]               1
[\n1\n, \n102 g CO2/km (comb)\n]                                                                         1
Name: Previous Owners, Length: 103, dtype: int64
```

```
df["Previous_Owners"] = [item[0] if type(item) == list else item for item in df["Previous Owners"]]
df["Previous_Owners"]
```

```
...
```

```
df["Previous_Owners2"] = df["Previous Owners"].apply(lambda item: item[0] if type(item) == list else item)
df["Previous_Owners2"]
```

23

Use this space to take notes:

Slide 24

Hints

- You can create functions to clean values

```
df["Fuel"].value_counts(dropna=False)
```

```
Diesel (Particulate Filter)          4315
Super 95                             4100
Gasoline                             3175
Diesel                               2984
Regular                               503
Super E10 95                          402
Super 95 (Particulate Filter)         268
```

```
benzine = ["Gasoline", "Super 95","Regular","Super E10 95","Super Plus 98","Super Plus E10 98", "Others"]
lpg = ["LPG","Liquid petroleum gas", "CNG", "Biogas", "Domestic gas H"]
def fueltype(x):
    if x in benzine:
        return "Benzine"
    elif x in lpg:
        return "LPG/CNG"
    else:
        return x
df["Fuel"] = df.Fuel.apply(fueltype)
```

24

Use this space to take notes:

Slide 25

Hints

- Relatively hard-to-handle columns
  - Consumption

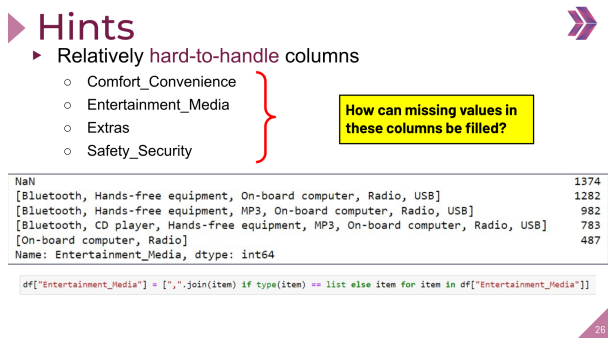
    To create separate columns, define the patterns for each consumption type. Then evaluate which one is enough to ML Model.

```
NaN                                                                                1906
[[3.9 l/100 km (comb)], [4.1 l/100 km (city)], [3.7 l/100 km (country)]]            304
[[4.2 l/100 km (comb)], [5 l/100 km (city)], [3.7 l/100 km (country)]]              276
[[5.4 l/100 km (comb)], [6.8 l/100 km (city)], [4.5 l/100 km (country)]]            257
[[3.8 l/100 km (comb)], [4.3 l/100 km (city)], [3.5 l/100 km (country)]]            253
                                                                                   ...
[[3.6 l/100 km (comb)], [], [4.4 l/100 km (country)]]                                1
[\n, 4.8 l/100 km (comb), \n, 5.6 l/100 km (city), \n, 4.3 l/100 km (country), \n]   1
[[7.6 l/100 km (comb)], [], []]                                                      1
[[5.6 l/100 km (comb)], [7.6 l/100 km (city)], [4.4 l/100 km (country)]]             1
[\n, 4.7 l/100 km (comb), \n, \n, \n]                                                1
Name: Consumption, Length: 882, dtype: int64
```

25

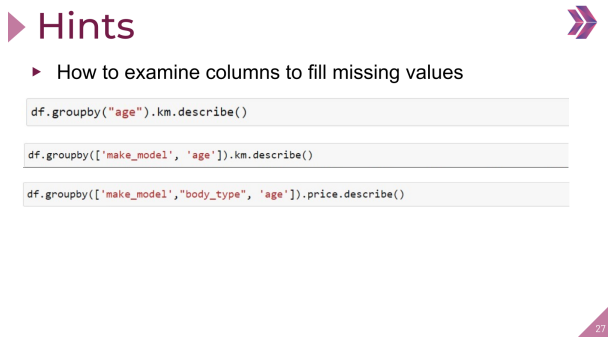
Use this space to take notes:

Slide 26



Use this space to take notes:

Slide 27



Use this space to take notes:

Slide 28

Hints

- How to fill missing values by groups

Example-1

```
#Step-1
#df["body_type"].fillna(df["body_type"].mode()[0])

#Step-2
#df.loc[df["make_model"]=="Audi A1", "body_type"].fillna(df[df["make_model"]=="Audi A1"]["body_type"].mode()[0])

#Step-3
for group in list(df["make_model"].unique()):
    cond = df["make_model"]==group
    mode = list(df[cond]["body_type"].mode())
    if mode != []:
        df.loc[cond, "body_type"] = df.loc[cond, "body_type"].fillna(df[cond]["body_type"].mode()[0])
    else:
        df.loc[cond, "body_type"] = df.loc[cond, "body_type"].fillna(df["body_type"].mode()[0])
```

You can generalize this loop to create your own function in your analysis

28

Use this space to take notes:

Slide 29

Hints

- How to fill missing values by groups

Example-2

```
#Step-1
#df.["Previous_Owners"].fillna(method="ffill")

#Step-2
#df.loc[df["age"]==0, "Previous_Owners"].fillna(method="ffill")

#Step-3
for group in list(df["age"].unique()):
    cond = df["age"]==group
    df.loc[cond, "Previous_Owners"] = df.loc[cond, "Previous_Owners"].fillna(method="ffill").fillna(method="bfill")
df["Previous_Owners"] = df["Previous_Owners"].fillna(method="ffill").fillna(method="bfill")
```

You can generalize this loop to create your own function

29

Use this space to take notes:

Slide 30

Hints

- How to fill missing values by groups

Example-3

```
# Step-1

# df.["Paint_Type"].fillna(method="ffill")

# Step-2
# df.loc[df["make_model"]=="Audi A1", "Paint_Type"].fillna(method="ffill")

# Step-3
# for group in list(df["make_model"].unique()):
#     cond = df["make_model"]==group
#     df.loc[cond, "Paint_Type"] = df.loc[cond, "Paint_Type"].fillna(method="ffill").fillna(method="bfill")
# df["Paint_Type"] = df["Paint_Type"].fillna(method="ffill").fillna(method="bfill")

# Step-4

for group1 in df["make_model"].unique():
    for group2 in list(df["body_type"].unique()):
        cond2 = (df["make_model"]==group1) & (df["body_type"]==group2)
        df.loc[cond2, "Paint_Type"] = df.loc[cond2, "Paint_Type"].fillna(method="ffill").fillna(method="bfill")

for group1 in list(df["make_model"].unique()):
    cond1 = df["make_model"]==group1
    df.loc[cond1, "Paint_Type"] = df.loc[cond1, "Paint_Type"].fillna(method="ffill").fillna(method="bfill")

df["Paint_Type"] = df["Paint_Type"].fillna(method="ffill").fillna(method="bfill")
```

30

Use this space to take notes:

Slide 31

Hints

- Dummy Operation

(The get_dummies function has 2 different uses)

pd.get_dummies(df)

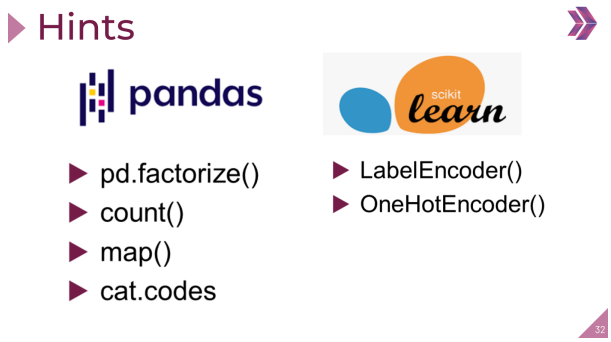
need to be research

df["col_name"].str.get_dummies(sep = " , ")

31

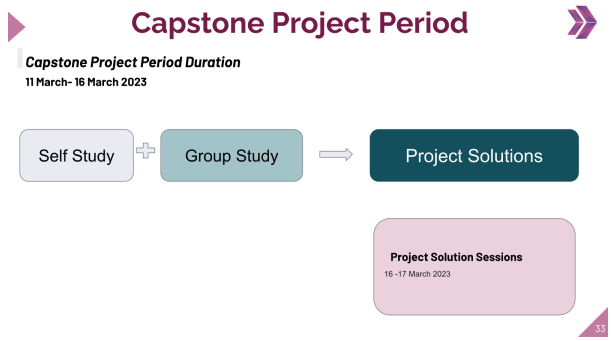
Use this space to take notes:

Slide 32



Use this space to take notes:

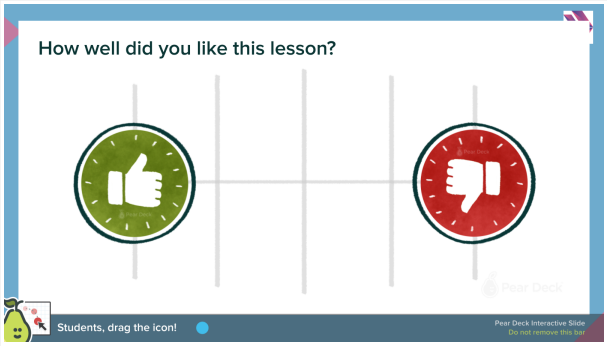
Slide 33



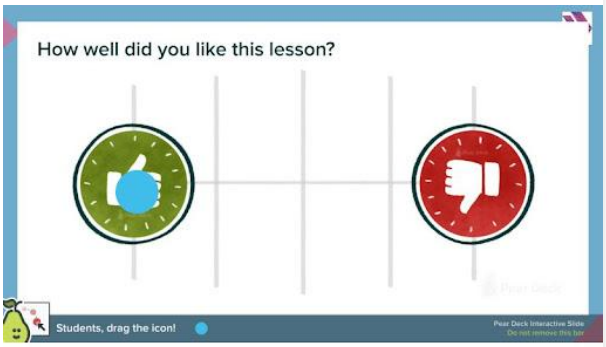
Use this space to take notes:

Slide 34	Your Response

Slide 34

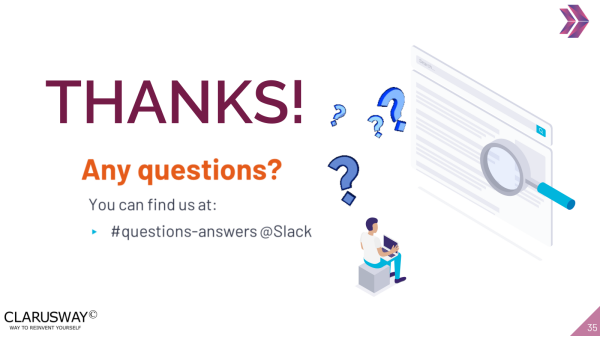


Your Response



Use this space to take notes:

Slide 35



Use this space to take notes: