Week 2

Understanding data and Data Wrangling

Some Numpy operations:

np.round()

np.linspace

To understand the data we need to use some descriptive statistics

df.describe() - gives statistical data like mean, median and so on

df.describe(include=’object/all’)

df.info() - gives the null counts and datatypes

df.value\_counts() = gives frequency count (can be used to plot histogram)

df.head(), df.tail() - first and last 5

df.dtypes - displays datatypes

df.isnull() - creates a new dataframe with True and False

Data Wrangling:

changing column names

dummy\_variable\_1.rename(columns={'IPS Panel':'Screen-IPS\_panel', 'Full HD':'Screen-Full\_HD'}, inplace=True) # for some columns

df.columns = columns\_list # for all columns

**How should you deal with missing data?**

1. Drop data  
   a. Drop the whole row  
   b. Drop the whole column
2. Replace data  
   a. Replace it by mean  
   b. Replace it by frequency  
   c. Replace it based on other functions

You should only drop whole columns if most entries in the column are empty. In the data set, none of the columns are empty enough to drop entirely. You have some freedom in choosing which method to replace data; however, some methods may seem more reasonable than others.

df.replace("?", np.nan, inplace = True)

avg\_norm\_loss = df["normalized-losses"].astype("float").mean(axis=0) - finding mean

df['num-of-doors'].value\_counts().idxmax() - finds the most frequent value.

# simply drop whole row with NaN in "price" column

df.dropna(subset=["price"], axis=0, inplace=True)

# reset index, because we droped two rows

df.reset\_index(drop=True, inplace=True)

-drops column

df.drop("fuel-type", axis = 1, inplace=True)

### Fixing the data types

Both "Weight\_kg" and "Screen\_Size\_cm" are seen to have the data type "Object", while both of them should be having a data type of "float". Write a code to fix the data type of these two columns.

# Write your code below and press Shift+Enter to execute

df[["Weight\_kg","Screen\_Size\_cm"]] = df[["Weight\_kg","Screen\_Size\_cm"]].astype('float')

## Data Standardization

You usually collect data from different agencies in different formats. (Data standardization is also a term for a particular type of data normalization where you subtract the mean and divide by the standard deviation.)

**What is standardization?**

Standardization is the process of transforming data into a common format, allowing the researcher to make the meaningful comparison.

**Example**

Transform mpg to L/100km:

In your data set, the fuel consumption columns "city-mpg" and "highway-mpg" are represented by mpg (miles per gallon) unit. Assume you are developing an application in a country that accepts the fuel consumption with L/100km standard.

You will need to apply **data transformation** to transform mpg into L/100km.

Use this formula for unit conversion:

L/100km = 235 / mpg

You can do many mathematical operations directly using Pandas.

## Data Normalization

**Why normalization?**

Normalization is the process of transforming values of several variables into a similar range. Typical normalizations include

1. scaling the variable so the variable average is 0
2. scaling the variable so the variance is 1
3. scaling the variable so the variable values range from 0 to 1

**Example**

To demonstrate normalization, say you want to scale the columns "length", "width" and "height".

**Target:** normalize those variables so their value ranges from 0 to 1

**Approach:** replace the original value by (original value)/(maximum value)

## Binning

**Why binning?**

Binning is a process of transforming continuous numerical variables into discrete categorical 'bins' for grouped analysis.

**Example:**

In your data set, "horsepower" is a real valued variable ranging from 48 to 288 and it has 59 unique values. What if you only care about the price difference between cars with high horsepower, medium horsepower, and little horsepower (3 types)? You can rearrange them into three ‘bins' to simplify analysis.

Use the Pandas method 'cut' to segment the 'horsepower' column into 3 bins.

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

group\_names = ['Low', 'Medium', 'High']

df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group\_names, include\_lowest=True )

## Indicator Variable

**What is an indicator variable?**

An indicator variable (or dummy variable) is a numerical variable used to label categories. They are called 'dummies' because the numbers themselves don't have inherent meaning.

**Why use indicator variables?**

You use indicator variables so you can use categorical variables for regression analysis in the later modules.

Example

The column "fuel-type" has two unique values: "gas" or "diesel". Regression doesn't understand words, only numbers. To use this attribute in regression analysis, you can convert "fuel-type" to indicator variables.

Use the Panda method 'get\_dummies' to assign numerical values to different categories of fuel type.

aspiration\_dummies = pd.get\_dummies(df['aspiration'])

df = pd.concat([df, aspiration\_dummies], axis = 1)

df.drop('aspiration', inplace = True, axis = 1)