SIMPLE DATAFRAME OPERATIONS

Accessing data along the column: df[“symbolling”]

Adding 1 to each value along the column: df[“symbolling”] = df[“symbolling”] + 1

DEALING WITH MISSING VALUES IN PYTHON

How to deal with missing data?

- Check with the data collection source
- Drop the missing value:
    - Drop the variable
    - Drop the data entry

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

  With: Axis = 0: drops the entire row; Axis = 1: drops the entire column

  df.replace(missing_value, new_value)

  For example: replace with mean of the data subset:

  mean = df[“normalized-losses”].mean()

  df[“normalized-losses”].replace(np.nan, mean)

- Replace the missing values
    - Replace it with an average (or similar datapoints)
    - Replace it by frequency
    - Replace it based on other functions
- Leave it as missing data

DATA FORMATTING

- Applying calculations to an entire column

  Convert “mpg” to “L/100km” in Car dataset:

  df[“city-mpg”] = 235/df[“city-mpg”]

  Rename the column:

  df.rename(columns = {“city_mpg” : “city-L/100km”}, inplace = True)

- Incorrect data types

  There are many data types in pandas:

  - Objects: “A”, “Hello” …
  - Int64: 1,3,5 …
  - Float64: 2.123, 632.01 …

  Correcting data type

  - To identify data types: df.dtypes()
  - To convert data types: df.astype()

    Example: df[“price”] = df[“price”].astype(“int”)

DATA NORMALIZATION IN PYTHON: uniform the features value with different range

Methods of normalizing data

- Simple feature scaling: resulted values range between 0 and 1

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

  df[“length”] = df[“length”]/df[“length”].max()

- Min – max: resulted values range between 0 and 1

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

df[“length”] = (df[“length”] - df[“length”].min()) / (df[“length”] = df[“length”] - df[“length”].min())

- Z – score: resulted values hover around 0 and mostly in range of -3 and 3

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

df[“length”] = (df[“length”]/df[“length”].mean()) / df[“ength”].std()

BINNING IN PYTHON

Binning: grouping of values into “bins”

- Converts numeric into categorical variables
- Group a set of numerical values into a set of “bins”

Example:

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)

TURNING CATEGORICAL VARIABLES INTO QUANTITATIVE VARIABLES IN PYTHON

Dummy variables in python pandas

- Use pandas.get_dummies() method
- Convert categorical variables to dummy variables (0 or 1)

For example: pd.get(dummies(df[‘fuel’])