Data_Generator

- **UC01 - Generate data**

"The Data_Generator artificially generates historical data based on the established predictors of the theoretical concept. The generation of the data is based on a dataset from a test environment of a shipment tracking system, which was provided by a logistics service provider. Distributing the data, the hypotheses are taken into account, where the linear dependencies between the variables are shown."

Dependencies:

- **Kilometres, stops and duration depend on the region.**
- **Duration also depends on weather conditions and traffic congestion**

In [1]:

```
import pandas as pd
import random

# function to generate a dataset with random values
# Km and Stops depends on Region
# Duration depends on Region, Weather and Traffic
# @param dataset_size: size of the dataset which should be generated for the models
# @return df: generated data frame
def generate_data_set(dataset_size):
    # size of the data frame
    size = range(0,dataset_size)

    # columns of the data frame
    features = ['Duration', 'Region', 'Km', 'Stops', 'Weather Extreme','Traffic']

    # create data frame object with defined size and columns
    df = pd.DataFrame(index = size, columns = features)

    # define weather values
    weather_extreme = ("none","rain", "snow")

    # fill each row
    for x in size:
        # initialize duration
        duration = 0.0
        # set random region
        df.loc[[x], 'Region'] = random.randint(1, 5)

        # set random Km and Stops for Region 1
        if df['Region'][x] == 1:
            df.loc[[x], 'Km'] = random.uniform(0.0, 5.0)
            df.loc[[x], 'Stops'] = random.randint(1, 20)

            # increase duration depending on the region with random value
            duration = duration + random.uniform(0.5, 2.0)

        # set random Km and Stops for Region 2
        if df['Region'][x] == 2:
            df.loc[[x], 'Km'] = random.uniform(4.0, 11.0)
            df.loc[[x], 'Stops'] = random.randint(10, 50)

            # increase duration depending on the region with random value
            duration = duration + random.uniform(1.5, 3.0)

        # set random Km and Stops for Region 3
        if df['Region'][x] == 3:
            df.loc[[x], 'Km'] = random.uniform(10.0, 15.0)
            df.loc[[x], 'Stops'] = random.randint(40, 80)
```

```
            # increase duration depending on the region with random value
            duration = duration + random.uniform(2.5, 4.0)

        # set random Km and Stops for Region 4
        if df['Region'][x] == 4:
            df.loc[[x], 'Km'] = random.uniform(9.0, 18.0)
            df.loc[[x], 'Stops'] = random.randint(70, 110)

            # increase duration depending on the region with random value
            duration = duration + random.uniform(3.5, 5.0)

        # set random Km and Stops for Region 5
        if df['Region'][x] == 5:
            df.loc[[x], 'Km'] = random.uniform(17.0, 20.0)
            df.loc[[x], 'Stops'] = random.randint(100, 140)

            # increase duration depending on the region with random value
            duration = duration + random.uniform(4.5, 6.0)

        # set random weather_extreme
        df.loc[[x], 'Weather Extreme'] = random.choice(weather_extreme)

        # increase duration for none weather extreme with random value
        if df['Weather Extreme'][x] == "none":
            duration = duration + random.uniform(0.0, 0.5)

        # increase duration for rain with random value
        if df['Weather Extreme'][x] == "rain":
            duration = duration + random.uniform(0.0, 1.0)

        # increase duration for snow with random value
        if df['Weather Extreme'][x] == "snow":
            duration = duration + random.uniform(0.0, 1.5)

        # set random value for traffic load (between 0 and 100 percent)
        df.loc[[x], 'Traffic'] = random.uniform(0.0, 100.0)

        # increase duration for traffic load between 0% and 25%
        if 0.0 <= df['Traffic'][x] <= 25.0:
            duration = duration + random.uniform(0.0, 0.5)

        # increase duration for traffic load between 25% and 50%
        if 25.0 <= df['Traffic'][x] <= 50.0:
            duration = duration + random.uniform(0.5, 1.0)

        # increase duration for traffic load between 50% and 75%
        if 50.0 <= df['Traffic'][x] <= 75.0:
            duration = duration + random.uniform(1.0, 1.5)

       # increase duration for traffic load between 75% and 100%
        if 75.0 <= df['Traffic'][x] <= 100.0:
            duration = duration + random.uniform(1.5, 2.0)

        # finally set calculated duration
        df.loc[[x], 'Duration'] = duration

    # return generated data frame
    return df
```