Data Science: Predictive Analytics, Übung 2

Henry Haustein

Data Preparation

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  from datetime import datetime
4  import time
5  from sklearn.impute import KNNImputer
6  import numpy as np
7  from sklearn.preprocessing import LabelEncoder
8  from sklearn.model_selection import train_test_split
9
10 # von Übung 1
11 def fixCols(cols):
12   liste = []
13   for col in cols:
14     newName = col.split("(")[0].strip().replace(" ", "_")
15     liste.append(newName)
16   return liste
17
18 data = pd.read_csv("dataset.csv", delimiter = ";", header = 0,
       index_col = 0, decimal = ",")
19 data.columns = fixCols(data.columns)
20 # print(data.dtypes)
21
22 # Übung 2
23
24 # 1
25 data["Date"] = pd.to_datetime(data["Date"])
26
27 # 2
28 data["weekday"] = data["Date"].dt.day_name()
29 dataDays = pd.DataFrame(data.groupby("weekday").sum()["
       Rented_Bike_Count"])
30 dataDays["daynumber"] = [time.strptime(x, "%A").tm_wday for x in
       list(dataDays.index.values)]
31 dataDays.sort_values("daynumber").iloc[:,0].plot()
32 plt.show()
33
34 # 3
35 def time2day(hour):
```

```
36    if 6 <= hour and hour <= 22:
37      return "day"
38    else:
39      return "night"
40
41  data["dayNight"] = data["Hour"].apply(time2day)
42  data.groupby("dayNight").sum()["Rented_Bike_Count"].plot.bar(x = "
        dayNight", y = "Rented_Bike_Count")
43  plt.show()
44
45  # 4
46  # wozu soll bitte die Tautemperatur notwendig sein? Also weg damit
47  data = data.drop(columns = ["Hotness", "Dew_point_temperature"])
48
49  # 5 + 6
50  data["Solar_Radiation"] = data["Solar_Radiation"].apply(lambda x:
        np.NaN if (x == "failure") else(float(x.replace(",", "."))))
51  # print(data.isna().sum())
52  imputer = KNNImputer(n_neighbors = 3, weights = "distance")
53  data[["Temperature", "Humidity", "Wind_speed", "Visibility", "
        Rainfall", "Snowfall", "Solar_Radiation"]] = imputer.
        fit_transform(data[["Temperature", "Humidity", "Wind_speed", "
        Visibility", "Rainfall", "Snowfall", "Solar_Radiation"]])
54  # print(data.isna().sum())
55
56  # 7
57  def fixTemp(temp):
58    if temp > 100:
59      return temp/10
60    else:
61      return temp
62
63  data["Temperature"] = data["Temperature"].apply(fixTemp)
64
65  # 8
66  data["Functioning_Day"] = data["Functioning_Day"].astype(bool)
67  data.loc[data["Seasons"] == "Herbst","Seasons"] = "Autumn"
68
69  # 9
70  encoder = LabelEncoder()
71  data["Seasons"] = encoder.fit_transform(data["Seasons"])
72  data["weekday"] = encoder.fit_transform(data["weekday"])
73  data["dayNight"] = encoder.fit_transform(data["dayNight"])
74  data["Holiday"] = encoder.fit_transform(data["Holiday"])
75  X = data[["Hour", "Temperature", "Wind_speed", "Humidity", "
        Visibility", "Solar_Radiation", "Rainfall", "Snowfall", "
        Seasons", "weekday", "dayNight", "Holiday"]]
76  y = data["Rented_Bike_Count"]
77  x_train, x_test, y_train, y_test = train_test_split(X, y,
        test_size = 0.3)
78
```

```
79  # print(data)
```