Data-preprocessing_respondents_csv

January 12, 2021

[6]:
```
from sensors.activpal import *
from utils import read_functions
from helpers import pandas_helper as pdh

import matplotlib.pyplot as plt
import seaborn as sns

activpal = Activpal()
```

1 Data Preprocessing

1.1 Renaming columns

[7]:
```
respondents_columns = {
    'ingeschat niveau': 'estimated_level',
    'lengte': 'length_cm',
    'gewicht': 'weight_kg',
    'middelomtrek': 'waist_circumference',
    'geslacht': 'gender',
    'leeftijdscategorie': 'age_category',
    'voldoet aan beweegrichtlijn 2017': 'meets_activity_guidelines',
    'voldoet aan richtlijn bot en spierversterkende activiteiten':␣
 ↪'meets_bone_and_muscle_strengthening_guidelines',
    'voldoet aan richtlijn balansoefeningen': 'meets_balance_guidelines',
    'sporter': 'is_sporter',
    'loop_snelheid_km': 'walking_speed_km',
    'ren_snelheid_km': 'running_speed_km'
}

respondents_df = pdh.read_csv_respondents_speed()
respondents_df.rename(columns = respondents_columns, inplace=True)
```

1.2 Missing values

[8]:
```
respondents_df.isnull().sum()
```

[8]:
```
estimated_level                                 0
length_cm                                       0
weight_kg                                       0
waist_circumference                             2
gender                                          0
age_category                                    0
meets_activity_guidelines                       0
meets_bone_and_muscle_strengthening_guidelines  0
meets_balance_guidelines                        0
is_sporter                                      0
walking_speed_km                                0
running_speed_km                                0
dtype: int64
```

Since there are only two null values in one column, fill null values with mean waist circumference

[9]:
```
respondents_df.waist_circumference.fillna(respondents_df.waist_circumference.
 ↪mean(), inplace=True)
respondents_df.waist_circumference.isnull().sum()
```

[9]:
```
0
```

1.3 Convert values to numerical

[10]:
```
def convert_respondents_values_to_numerical(respondents_df):
    if not np.issubdtype(respondents_df.estimated_level.dtype, int):
        respondents_df.estimated_level = respondents_df.estimated_level.
 ↪apply(lambda x: x == 'actief').astype(int)

    if not np.issubdtype(respondents_df.gender.dtype, int):
        respondents_df.gender = respondents_df.gender.apply(lambda x: x ==␣
 ↪'man').astype(int)

    if not np.issubdtype(respondents_df.meets_activity_guidelines.dtype, int):
        respondents_df.meets_activity_guidelines = respondents_df.
 ↪meets_activity_guidelines.apply(lambda x: x == 'ja').astype(int)

    if not np.issubdtype(respondents_df.
 ↪meets_bone_and_muscle_strengthening_guidelines.dtype, int):
        respondents_df.meets_bone_and_muscle_strengthening_guidelines =␣
 ↪respondents_df.meets_bone_and_muscle_strengthening_guidelines.apply(lambda x:
 ↪ x == 'ja').astype(int)
```

```
    if not np.issubdtype(respondents_df.meets_balance_guidelines.dtype, int):
        respondents_df.meets_balance_guidelines = respondents_df.
↪meets_balance_guidelines.apply(lambda x: x == 'ja').astype(int)

    if not np.issubdtype(respondents_df.age_category.dtype, int):
        age_category = {
            "15-19": 0, "20-24": 1, "25-29": 2, "30-34": 3,
            "35-39": 4,  "40-44": 5, "45-49": 6, "50-54": 7,
            "55-59": 8,  "60-64": 9, "65-69": 10,  "70-74": 11, "75-79": 12
        }
        respondents_df.age_category.replace(age_category, inplace=True)

    return respondents_df

respondents_df = convert_respondents_values_to_numerical(respondents_df)
respondents_df.dtypes
```

[10]:
```
estimated_level                                 int64
length_cm                                     float64
weight_kg                                     float64
waist_circumference                           float64
gender                                          int64
age_category                                    int64
meets_activity_guidelines                       int64
meets_bone_and_muscle_strengthening_guidelines  int64
meets_balance_guidelines                        int64
is_sporter                                      int64
walking_speed_km                              float64
running_speed_km                              float64
dtype: object
```

1.4 Remove low variance

[11]:
```
from sklearn.feature_selection import VarianceThreshold

selector = VarianceThreshold(threshold=(.8 * (1 - .8)))
selector.fit(respondents_df)
support = selector.get_support()
respondents_df = respondents_df.loc[:,support]
respondents_df.columns
```

[11]:
```
Index(['estimated_level', 'length_cm', 'weight_kg', 'waist_circumference',
       'gender', 'age_category', 'running_speed_km'],
      dtype='object')
```

1.4.1 Save DataFrame

[49]:
```
respondents_df.to_csv('../../data/respondents-cleaned.csv')
```