05b_calculate_weights

May 26, 2024

1 Calculate Normalized Weights for Probability Calculation

1.1 Content

1. Import files
2. Normalize Data
3. Calculate Weights
4. Save Data

[31]:

```
# imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from collections import Counter
from sklearn.metrics import *
from sklearn.utils import resample
import matplotlib.colors as mcolors
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

import

[32]:

```
# import files
df = pd.read_csv("files/fo_smote.csv")

print(df.shape[0])
df.head(5)
```

```
356
```

[32]:

```
   isco08  Berufshauptgruppe    s1    s2    s3    s4    s5    s6    s7    s8  \
0    2655                  2  0.72  0.72  0.69  0.69  0.50  0.50  0.50  0.47
1    2612                  2  0.81  0.75  0.81  0.72  0.81  0.66  0.56  0.72
2    3115                  3  0.69  0.66  0.66  0.47  0.72  0.53  0.53  0.63
```

```
3    2120                    2  0.81  0.72  0.75  0.50  0.81  0.53  0.50  0.75
4    1222                    1  0.69  0.75  0.78  0.75  0.75  0.56  0.63  0.63

   …  a45  a46   a47   a48   a49   a50   a51  a52  fo_probability  \
0  …  0.0  0.0  0.00  0.00  0.00  0.00  0.00  0.0           0.370
1  …  0.0  0.0  0.00  0.00  0.00  0.00  0.00  0.0           0.400
2  …  0.0  0.0  0.25  0.28  0.19  0.03  0.03  0.0           0.240
3  …  0.0  0.0  0.00  0.00  0.00  0.00  0.00  0.0           0.035
4  …  0.0  0.0  0.00  0.00  0.00  0.00  0.00  0.0           0.015

   fo_computerisation
0                   0
1                   0
2                   0
3                   0
4                   0

[5 rows x 91 columns]
```

## Normalize Data

[33]:
```python
# Initialize a scaler
scaler = StandardScaler()

# Define the columns to scale
cols_to_scale = [col for col in df.columns if col not in ['isco08',␣
 ↪'Name_de',"Berufshauptgruppe","fo_probability","fo_computerisation"]]

# Scale only these columns
df_scaled = df.copy()
df_scaled[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])

df_scaled.head(5)
```

[33]:
```
   isco08  Berufshauptgruppe        s1        s2        s3        s4  \
0    2655                  2  0.961125  0.835289  0.542122  1.384220
1    2612                  2  1.613999  1.077415  1.550860  1.642504
2    3115                  3  0.743500  0.351039  0.289938 -0.509866
3    2120                  2  1.613999  0.835289  1.046491 -0.251582
4    1222                  1  0.743500  1.077415  1.298676  1.900789

         s5        s6        s7        s8  …       a45       a46       a47  \
0 -0.858716 -0.681654 -0.205927 -0.418321  … -0.869331 -0.897504 -1.124347
1  1.845918  0.835128  0.428009  1.555735  … -0.869331 -0.897504 -1.124347
2  1.060702 -0.397258  0.111041  0.845075  … -0.869331 -0.897504  0.099171
3  1.845918 -0.397258 -0.205927  1.792622  … -0.869331 -0.897504 -1.124347
4  1.322441 -0.112861  1.167601  0.845075  … -0.869331 -0.897504 -1.124347
```

```
        a48       a49       a50       a51       a52  fo_probability  \
0 -1.183707 -1.150578 -0.899383 -1.054109 -1.010659           0.370
1 -1.183707 -1.150578 -0.899383 -1.054109 -1.010659           0.400
2  0.078750 -0.122511 -0.658196 -0.853614 -1.010659           0.240
3 -1.183707 -1.150578 -0.899383 -1.054109 -1.010659           0.035
4 -1.183707 -1.150578 -0.899383 -1.054109 -1.010659           0.015

   fo_computerisation
0                   0
1                   0
2                   0
3                   0
4                   0

[5 rows x 91 columns]
```

## Calculate Coefficients

[34]:

```
# Select columns that start with 's' or 'a' and the 'fo_probability' column
df_selected = df_scaled.filter(regex='^(s|a|fo_computerisation)')

# Define the dependent variable
y = df_selected['fo_computerisation']

# Define the independent variables
X = df_selected.drop('fo_computerisation', axis=1)

# Fit the model
model = LogisticRegression(C=0.3, penalty='l2', solver='newton-cg',␣
 ↪random_state=42, max_iter=1000)
model.fit(X, y)

intercept = model.intercept_

# Get the regression coefficients
coefficients = pd.Series(model.coef_[0], index=X.columns)
```

[35]:

```
# Create a dataframe from the coefficients outside the 95% and 99% confidence␣
 ↪intervals with headers
df_coefficients = pd.DataFrame(list(coefficients.items()), columns=['id',␣
 ↪'coefficient'])

df_coefficients.head(2)
```

[35]:

```
   id  coefficient
0  s1     0.101763
1  s2    -0.190055
```

Save the data

[36]:

```
# Save the dataframes to csv files
df_coefficients.reset_index().rename(columns={'index': 'Variable', 0: 'Coefficient'}).sort_values('coefficient').drop(columns=["Variable"]).to_csv('files/coefficients.csv', index=False)
pd.DataFrame([intercept], columns=['intercept']).to_csv('files/intercept.csv', index=False)
```