07_calculate_probabilty

May 26, 2024

1 Calculate Automation Probabilities of Jobs on the Swiss Labour Market

1.1 Content

1. Import Data
2. Normalize the Data
3. Calculate Weighted Sum from Coefficients
4. Calculate Probability of Automation
5. Display Results
6. Save Results

[723]:
```
# imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import math as m
```

Import Data

[724]:
```
# import files
df_swiss_jobs = pd.read_csv('files/switzerland_occupations.csv')
coefficients_outside_95 = pd.read_csv('files/coefficients.csv')
intercept = pd.read_csv('files/intercept.csv').iloc[0, 0]
not_automatable = [item[0] for item in pd.read_csv("files/not_automatable.csv").
 ↪values.tolist()]
```

[725]:
```
df_swiss_jobs.head(2)
```

[725]:
```
   isco08       Name_de  Berufshauptgruppe    s1    s2    s3    s4    s5  \
0    2655  Schauspieler                  2  0.72  0.72  0.69  0.69  0.50
1    2612       Richter                  2  0.81  0.75  0.81  0.72  0.81

     s6    s7  …  a43   a44  a45  a46  a47  a48  a49  a50  a51  a52
0  0.50  0.50  …  0.0  0.00  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.66  0.56  …  0.0  0.31  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[2 rows x 90 columns]
```

[726]:
```
coefficients_outside_95.head(2)
```

[726]:
```
    id  coefficient
0  a12    -0.809382
1   s6    -0.701232
```

[727]:
```
cols_to_keep = not_automatable
cols_to_keep.extend(['isco08', 'Name_de', 'Berufshauptgruppe'])

# Drop the other columns
df_swiss_jobs = df_swiss_jobs[cols_to_keep]

# Display the first few rows of the DataFrame
df_swiss_jobs.head(2)
```

[727]:
```
    a12   s4  s27   s15   s8  s31  s26   s24  isco08       Name_de  \
0  0.47  0.69  0.0  0.38  0.47  0.0  0.0  0.06    2655  Schauspieler
1  0.78  0.72  0.0  0.56  0.72  0.0  0.0  0.19    2612       Richter

   Berufshauptgruppe
0                  2
1                  2
```

Normalize the Data

[728]:
```
# Initialize a scaler
scaler = StandardScaler()

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

# Scale only these columns
df_swiss_jobs_scaled = df_swiss_jobs.copy()
df_swiss_jobs_scaled[cols_to_scale] = scaler.
 ↪fit_transform(df_swiss_jobs[cols_to_scale])

df_swiss_jobs_scaled.head(2)
```

[728]:
```
        a12       s4      s27       s15       s8       s31      s26  \
0 -1.135638  1.329989 -0.478152 -0.154766 -0.453849 -1.464599 -1.08397
1  1.552699  1.584277 -0.478152  1.142521  1.502201 -1.464599 -1.08397

        s24  isco08       Name_de  Berufshauptgruppe
0 -1.190280    2655  Schauspieler                  2
1  0.329845    2612       Richter                  2
```

Calculate Weighted Sum from Coefficients

[729]:
```
# Create a dictionary from the coefficients DataFrame
coef_dict = coefficients_outside_95.set_index('id')['coefficient'].to_dict()

print(coef_dict["s31"])
```

```
0.0280082924110774
```

[730]:
```
# calculate weighted sums
ignore_cols = ['Name_de', 'isco08', 'Berufshauptgruppe']
weighted_sums = []

for index, row in df_swiss_jobs_scaled.iterrows():
    sum = 0
    for col_name, col_value in row.items():
        if col_name not in ignore_cols:
            sum += coef_dict[col_name] * col_value
    weighted_sums.append(sum)

# Create a new DataFrame for the weighted sums
df_swiss_jobs_scaled["weighted_sum"] = pd.DataFrame(weighted_sums)

df_swiss_jobs_scaled.head(2)
```

[730]:

	a12	s4	s27	s15	s8	s31	s26	\
0	-1.135638	1.329989	-0.478152	-0.154766	-0.453849	-1.464599	-1.08397	
1	1.552699	1.584277	-0.478152	1.142521	1.502201	-1.464599	-1.08397	

	s24	isco08	Name_de	Berufshauptgruppe	weighted_sum
0	-1.190280	2655	Schauspieler	2	0.050777
1	0.329845	2612	Richter	2	-0.822573

Calculate Probability of Automation

[731]:
```
# Calculate the probabilities
prob_list = [m.exp(x_i + intercept) / (1 + m.exp(x_i + intercept)) for x_i in␣
 ↪df_swiss_jobs_scaled['weighted_sum']]
```

[732]:
```
# Add the probabilities to the DataFrame
df_swiss_jobs_scaled['probability'] = prob_list

df_swiss_jobs_scaled.head(5)
```

[732]:

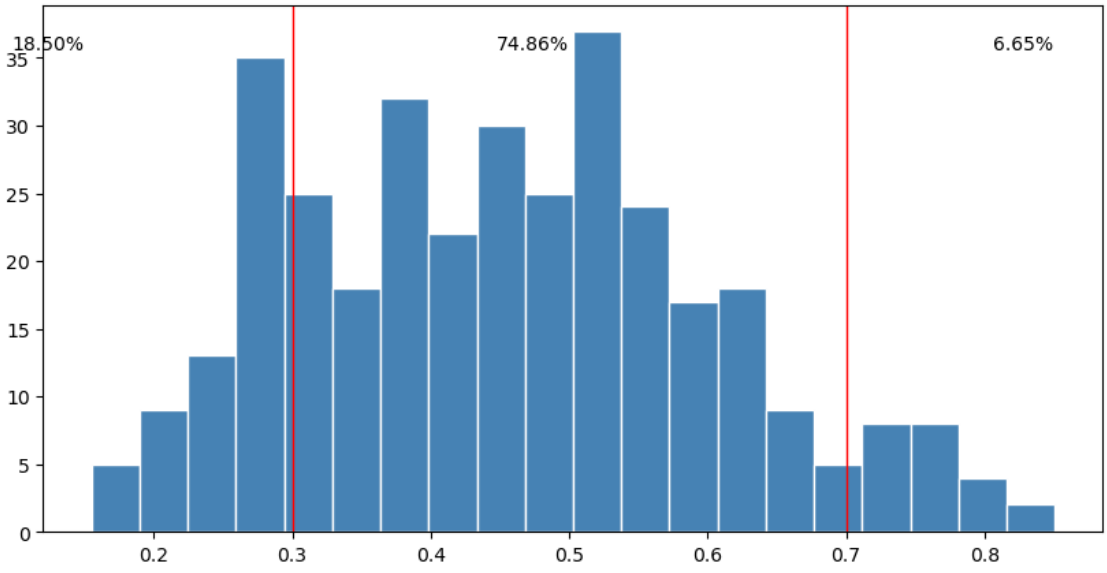
	a12	s4	s27	s15	s8	s31	s26	\
0	-1.135638	1.329989	-0.478152	-0.154766	-0.453849	-1.464599	-1.083970	
1	1.552699	1.584277	-0.478152	1.142521	1.502201	-1.464599	-1.083970	
2	1.032376	-0.534788	2.095368	-0.154766	0.798023	0.792444	1.302311	
3	1.292537	-0.280500	-0.478152	0.277663	1.736927	-1.464599	-1.083970	
4	1.032376	1.838564	-0.478152	0.926306	0.798023	-1.464599	-0.909364	

```
        s24  isco08                                              Name_de  \
0 -1.190280    2655                                          Schauspieler
1  0.329845    2612                                               Richter
2  1.382239    3115  Maschinenbautechniker, Techniker im Bereich Sy…
3  0.680643    2120              Mathematiker, Aktuare und Statistiker
4  0.329845    1222  Führungskräfte in Werbung und Öffentlichkeitsa…

   Berufshauptgruppe  weighted_sum  probability
0                  2      0.050777     0.461774
1                  2     -0.822573     0.263753
2                  3     -0.342688     0.366638
3                  2     -0.044931     0.438091
4                  1     -0.671868     0.294039
```

Display Results

[733]:
```
# Get the values
x = df_swiss_jobs_scaled["probability"].values

# Set the size of the plot
plt.figure(figsize=(10, 5))

# Calculate the histogram
hist, bins = np.histogram(x, bins=20)

# Calculate the width and center of the bars
width = 1 * (bins[1] - bins[0])
center = (bins[:-1] + bins[1:]) / 2

# Plot the histogram
plt.bar(center, hist, align='center', width=width, color='#4682B4',␣
 ↪edgecolor='white')

# Add lines at 0.3 and 0.7
plt.axvline(0.3, color='r', linewidth=1)
plt.axvline(0.7, color='r', linewidth=1)

# Calculate the percentage of jobs between 0 and 0.3, 0.3 and 0.7, 0.7 and 1
percentage_0_03 = (x <= 0.3).mean() * 100
percentage_03_07 = ((0.3 < x) & (x < 0.7)).mean() * 100
percentage_07_1 = (x >= 0.7).mean() * 100

# Add the percentages to the plot
plt.text(0.15, max(hist) - 1, f'{percentage_0_03:.2f}%', ha='right',␣
 ↪va='center', color='black')
```

```
plt.text(0.5, max(hist) - 1, f'{percentage_03_07:.2f}%', ha='right',␣
 ↪va='center', color='black')
plt.text(0.85, max(hist) - 1, f'{percentage_07_1:.2f}%', ha='right',␣
 ↪va='center', color='black')

# Show the plot
plt.show()
```



[734]:
```
# Jobs with probability below 0.3
jobs_below_03 = df_swiss_jobs_scaled[df_swiss_jobs_scaled['probability'] <= 0.3]
print(f"Number of jobs with probability below 0.3: {len(jobs_below_03)}")

# Jobs with probability between 0.3 and 0.7
jobs_between_03_and_07 =␣
 ↪df_swiss_jobs_scaled[(df_swiss_jobs_scaled['probability'] > 0.3) &␣
 ↪(df_swiss_jobs_scaled['probability'] <= 0.7)]
print(f"Number of jobs with probability between 0.3 and 0.7:␣
 ↪{len(jobs_between_03_and_07)}")

# Jobs with probability above 0.7
jobs_above_07 = df_swiss_jobs_scaled[df_swiss_jobs_scaled['probability'] > 0.7]
print(f"Number of jobs with probability above 0.7: {len(jobs_above_07)}")
```

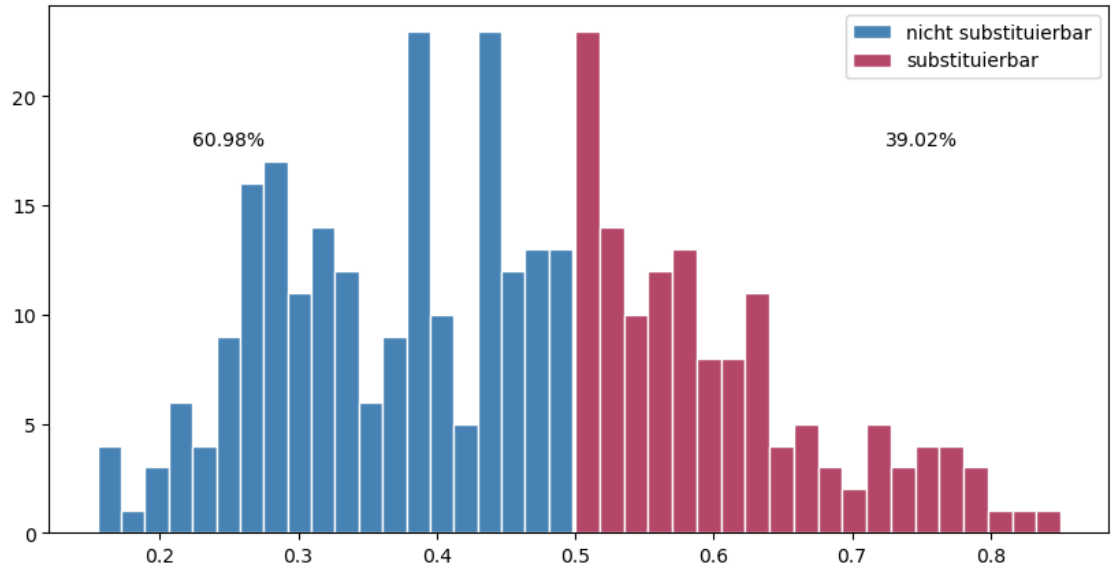
```
Number of jobs with probability below 0.3: 64
Number of jobs with probability between 0.3 and 0.7: 259
Number of jobs with probability above 0.7: 23
```

[735]:
```
# Percentage of jobs with probability above 0.5
percentage_above_05 = (df_swiss_jobs_scaled['probability'] > 0.5).mean() * 100
print(f"Percentage of jobs with probability above 0.5: {percentage_above_05:.
↪2f}%")
```

```
Percentage of jobs with probability above 0.5: 39.02%
```

[736]:
```
# Get the values
x = df_swiss_jobs_scaled["probability"].values

# Create subsets for probabilities above and below 0.5
x_below_05 = x[x <= 0.5]
x_above_05 = x[x > 0.5]

# Set the size of the plot
plt.figure(figsize=(10, 5))

# Calculate the histograms
hist_below_05, bins_below_05 = np.histogram(x_below_05, bins=20)
hist_above_05, bins_above_05 = np.histogram(x_above_05, bins=20)

# Calculate the width and center of the bars
width_below_05 = 1 * (bins_below_05[1] - bins_below_05[0])
center_below_05 = (bins_below_05[:-1] + bins_below_05[1:]) / 2

width_above_05 = 1 * (bins_above_05[1] - bins_above_05[0])
center_above_05 = (bins_above_05[:-1] + bins_above_05[1:]) / 2

# Plot the histograms
plt.bar(center_below_05, hist_below_05, align='center', width=width_below_05,␣
↪color='#4682B4', edgecolor='white', label='nicht substituierbar')
plt.bar(center_above_05, hist_above_05, align='center', width=width_above_05,␣
↪color='#B44668', edgecolor='white', label='substituierbar')

# Calculate the percentages
percentage_below_05 = len(x_below_05) / len(x) * 100
percentage_above_05 = len(x_above_05) / len(x) * 100

# Add the percentages to the plot
plt.text(0.25, max(hist_below_05) - 5, f'{percentage_below_05:.2f}%',␣
↪ha='center', va='center')
plt.text(0.75, max(hist_above_05) - 5, f'{percentage_above_05:.2f}%',␣
↪ha='center', va='center')

# Add a legend
plt.legend()
```

```
# Show the plot
plt.show()
```



[737]:
```
# Group by 'berufshauptgruppe' and calculate the mean probability
df_probability_per_berufshauptgruppe = df_swiss_jobs_scaled.
  ↪groupby('Berufshauptgruppe')['probability'].mean().reset_index()
```

Save Results

[738]:
```
df_probabilities = df_swiss_jobs_scaled[['Name_de', 'Berufshauptgruppe',␣
  ↪'probability', "isco08"]].copy()
df_pytorch_model = df_swiss_jobs_scaled.drop(columns=["weighted_sum"])
```

[739]:
```
df_probabilities['probability'] = df_probabilities['probability'].round(4)
df_probabilities.to_csv('files/switzerland_probabilities.csv', index=False)
df_probability_per_berufshauptgruppe.to_csv('files/
  ↪switzerland_probability_berufshauptgruppe.csv', index=False)
```