Progress Report

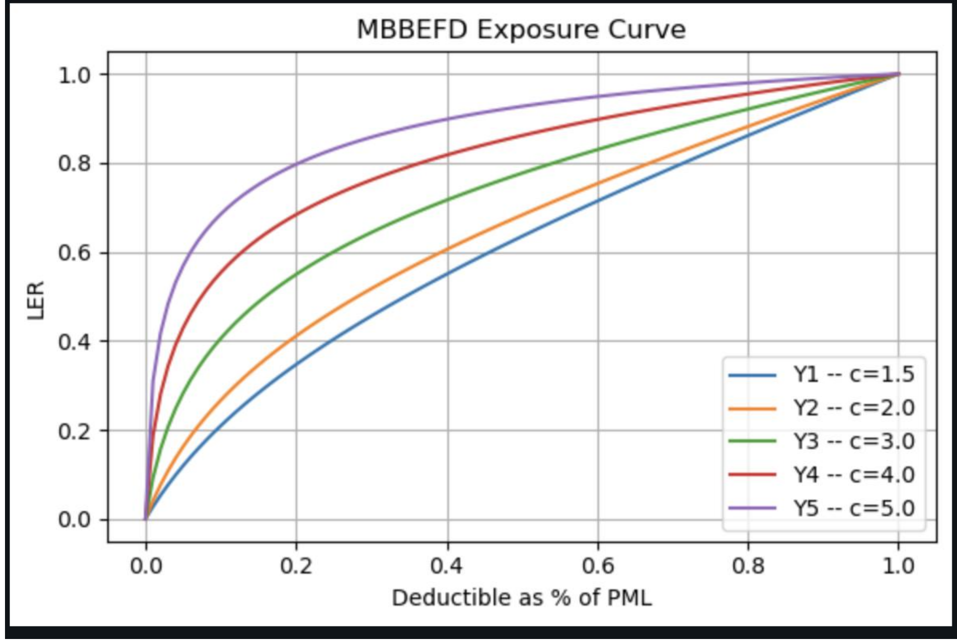
7-19-2024

01 - Implement MBBEFD & Swiss Re Curve with Python

```python
def swissRe(c):
    """
    Input: c (1.5, 2, 3, 4, 5)
    Returns: (b, g) parameters for the MBBEFD curves Y1, Y2, Y3, Y4, and Lloyd's curve
    """
    b = np.exp(3.1-0.15*(1+c)*c)
    g = np.exp((0.78+0.12*c)*c)

    return b, g

def MBBEFD(x, b, g):
    if g==1 or b==0:
        return x
    if b==1 and g>1:
        return np.log(1+(g-1)*x)/np.log(g)
    if b*g==1 and g>1:
        return (1-b**x)/(1-b)
    if b>0 and b!=1 and b*g!=1 and g>1:
        return np.log(((g-1)*b+(1-g*b)*(b**x))/(1-b))/np.log(g*b)
    return "Error. Please check input b and g."

# Define a range of c values (deductible as % of PML)
c_values = np.array([1.5, 2, 3, 4, 5])

x=np.linspace(0,1,101)
curves={}
for c in c_values:
    b, g = swissRe(c)
    y=MBBEFD(x, b, g)
    curves[c]=y
```

```python
curves=pd.DataFrame(curves,index=x)
curves

# Plot b and g parameters
plt.figure(figsize=(6,4))
plt.subplot()
plt.title('MBBEFD Exposure Curve')
plt.plot(curves)
plt.xlabel('Deductible as % of PML')
plt.ylabel('LER')
plt.legend([f'Y{i+1} -- c={c_values[i]}' for i in range(5)])
plt.grid(True)

plt.tight_layout()
plt.show()
```

✓ 0.1s



02 - Pending Revision of Clustering Results

Here are 2 changes added to the first version of clustering results (sent by Luyang on 7/18).

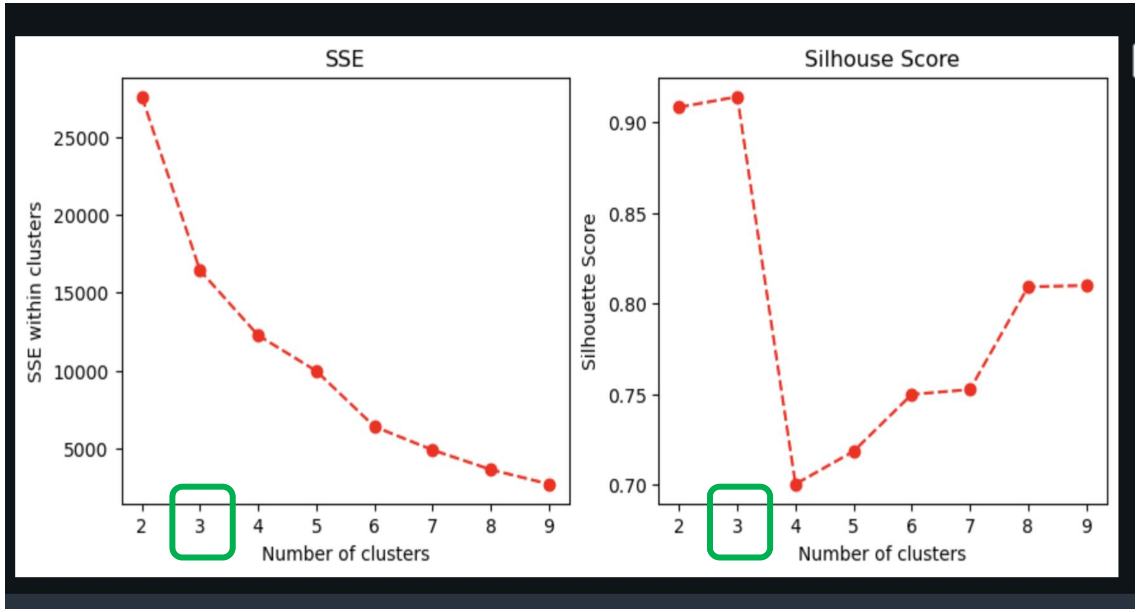
1. Include “propery_type” as a feature, using OneHot Encoding.

sedan	SUV	sports car	marine cargo	turboprop aircraft	light sport aircraft	property_value	pml	deductible	cv
0	0	0	0	0	0	-0.078419	-0.158192	0.009751	0.070653
0	0	0	0	0	0	-0.074858	-0.155793	-0.153713	0.070653
0	0	0	0	0	0	-0.193779	-0.237351	-0.112847	0.070653
0	0	0	0	0	0	-0.161734	-0.215762	-0.174146	0.070653
0	0	0	0	0	0	-0.256444	-0.278130	0.009751	0.070653
...	...	...	...	...	...	...	...	...	...

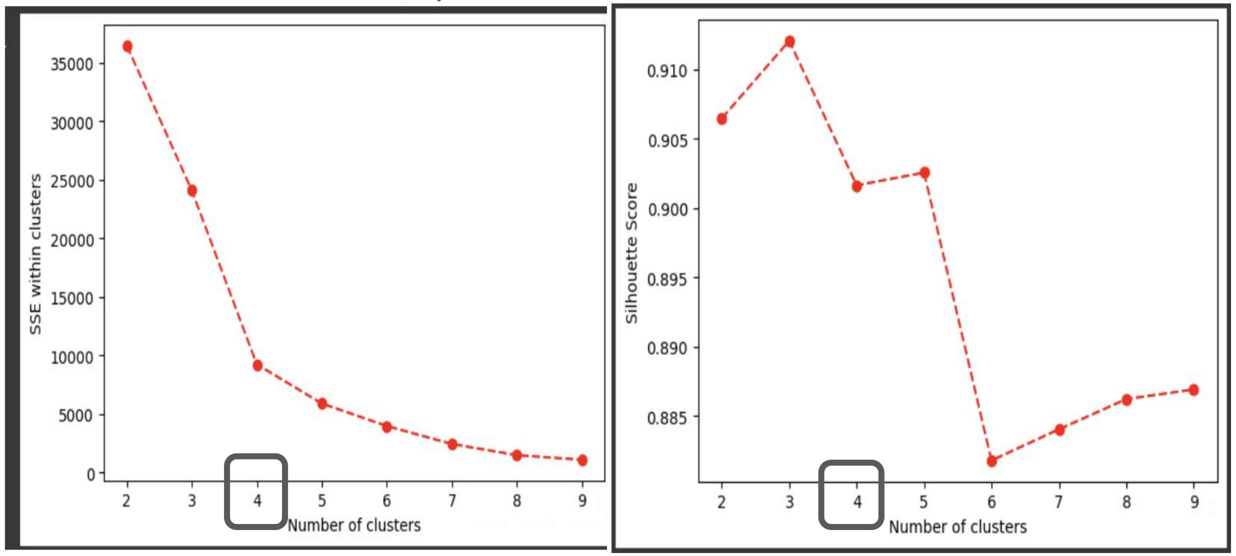
(old features)

	property_value	pml	deductible	cv
0	-0.078419	-0.158192	0.009751	-0.321627
1	-0.074858	-0.155793	-0.153713	-0.321627
2	-0.193779	-0.237351	-0.112847	-0.321627
3	-0.161734	-0.215762	-0.174146	-0.321627
4	-0.256444	-0.278130	0.009751	-0.321627
...	...	...	...	...

2. The results seems better when k=3.(instead of 4) *The reasoning for changing k to 4 is explained in the next page.*



New Results



Old Results

02 - Pending Revision of Clustering Results

The reasoning for changing k to 4:

(According to ChatGPT) when there’s a conflict between SSE and Silhouse Score:

When to Trust Silhouette Score

- **Quality of Clusters**: If you care more about the quality and distinctiveness of clusters rather than merely compactness, the Silhouette Score is more informative.
- **Interpretability**: Silhouette Score provides a more interpretable measure of how well-defined clusters are.

When to Trust SSE

- **Compactness**: If the goal is to minimize within-cluster variance and create very tight clusters, SSE might be a more appropriate metric.
- **Objective Function**: If you are using an algorithm like K-Means, which directly optimizes SSE, then SSE can be a more relevant metric.

Combining Both Metrics

Sometimes, it might be beneficial to consider both metrics together:

- Use SSE to identify a reasonable range for the number of clusters.
- Within this range, use the Silhouette Score to select the optimal number of clusters and validate cluster quality.

∵

1. It seems like SSE(k=4) is no much smaller than SSE(k=3)
2. SilScore(k=4) is way lower than SilScore(k=3)
3. K=3 has better interpretability:

    3 clusters represent 3 risk levels(“high-mid-low”)

∴

k=3 will be easier to justify in the Presentation.