vector-analysis

April 6, 2023

1 Set-up

1.1 Globals

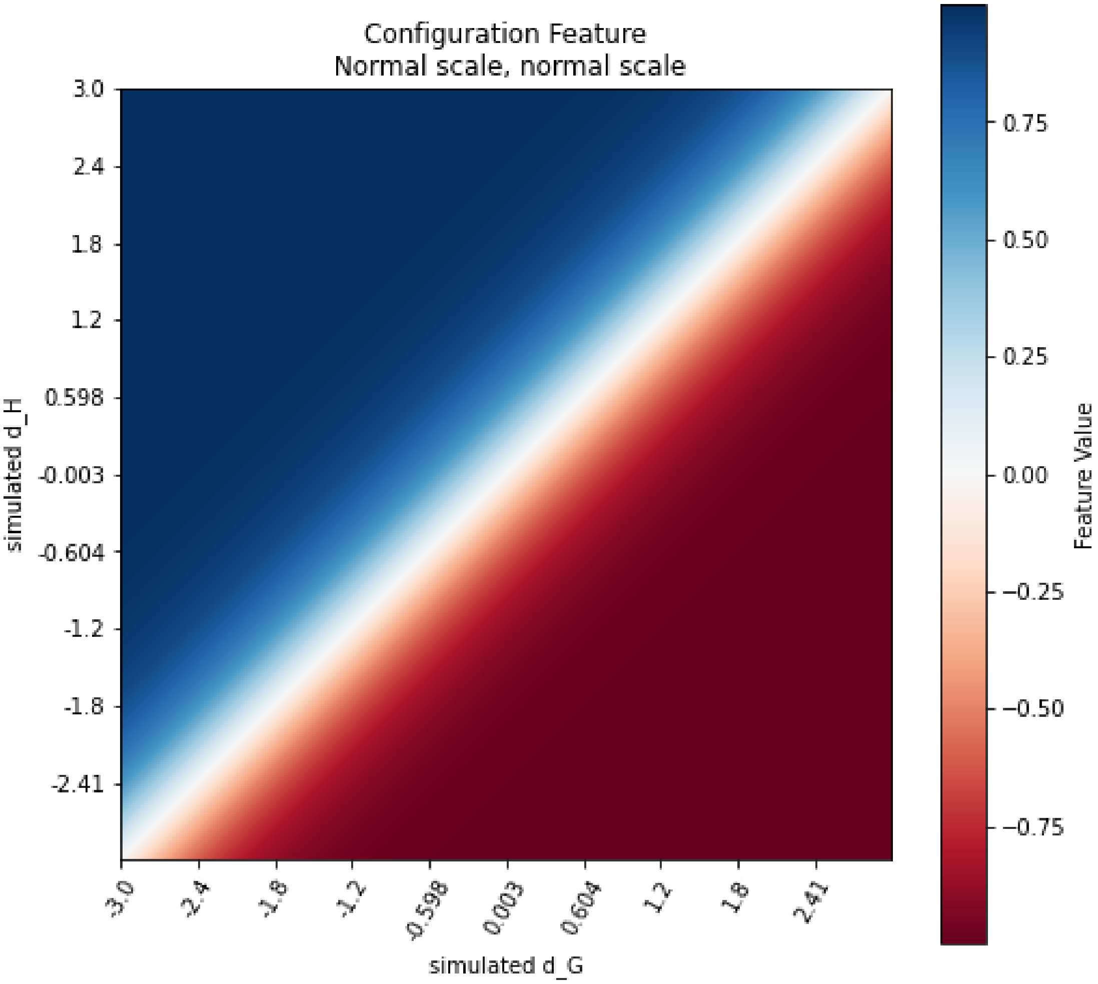
1.2 Loading cache and applying transformations

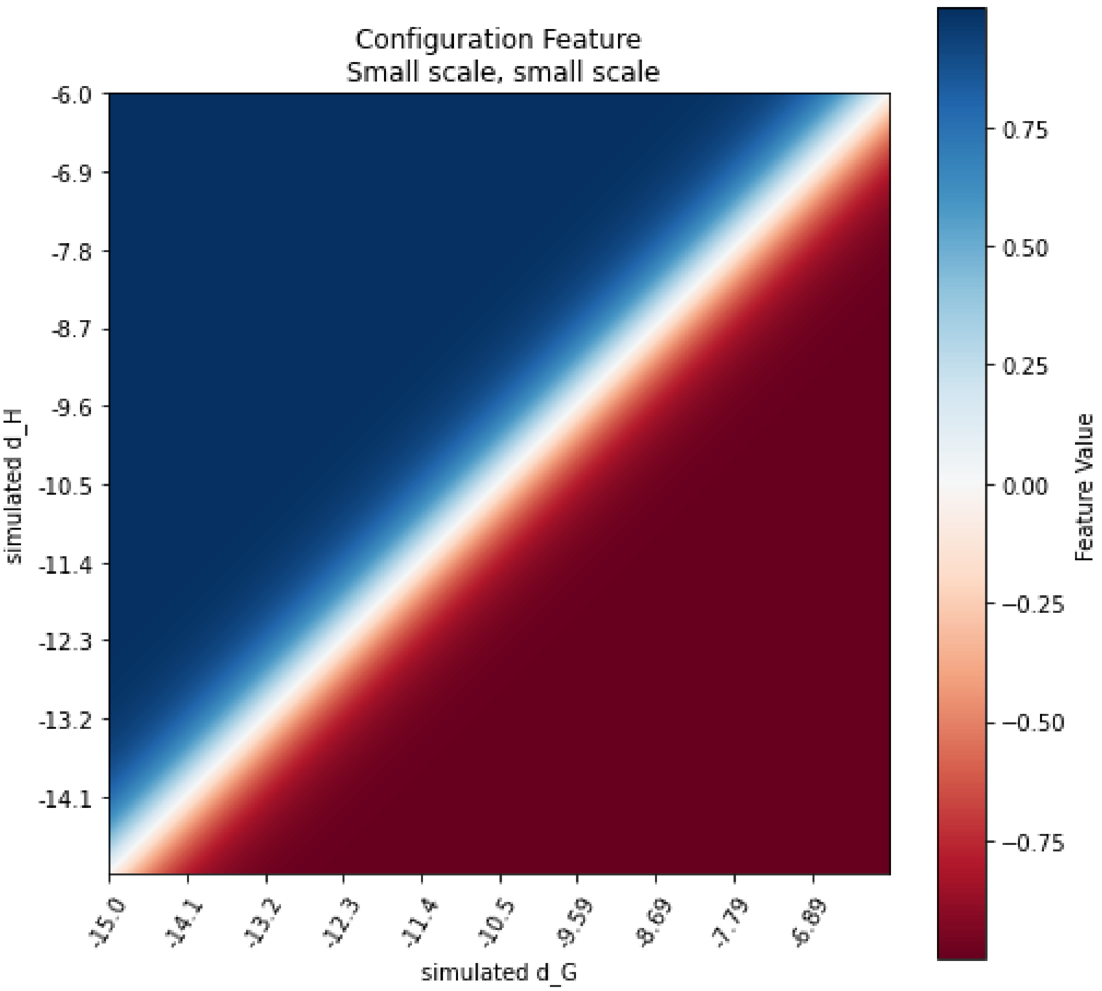
2 Precomputing features

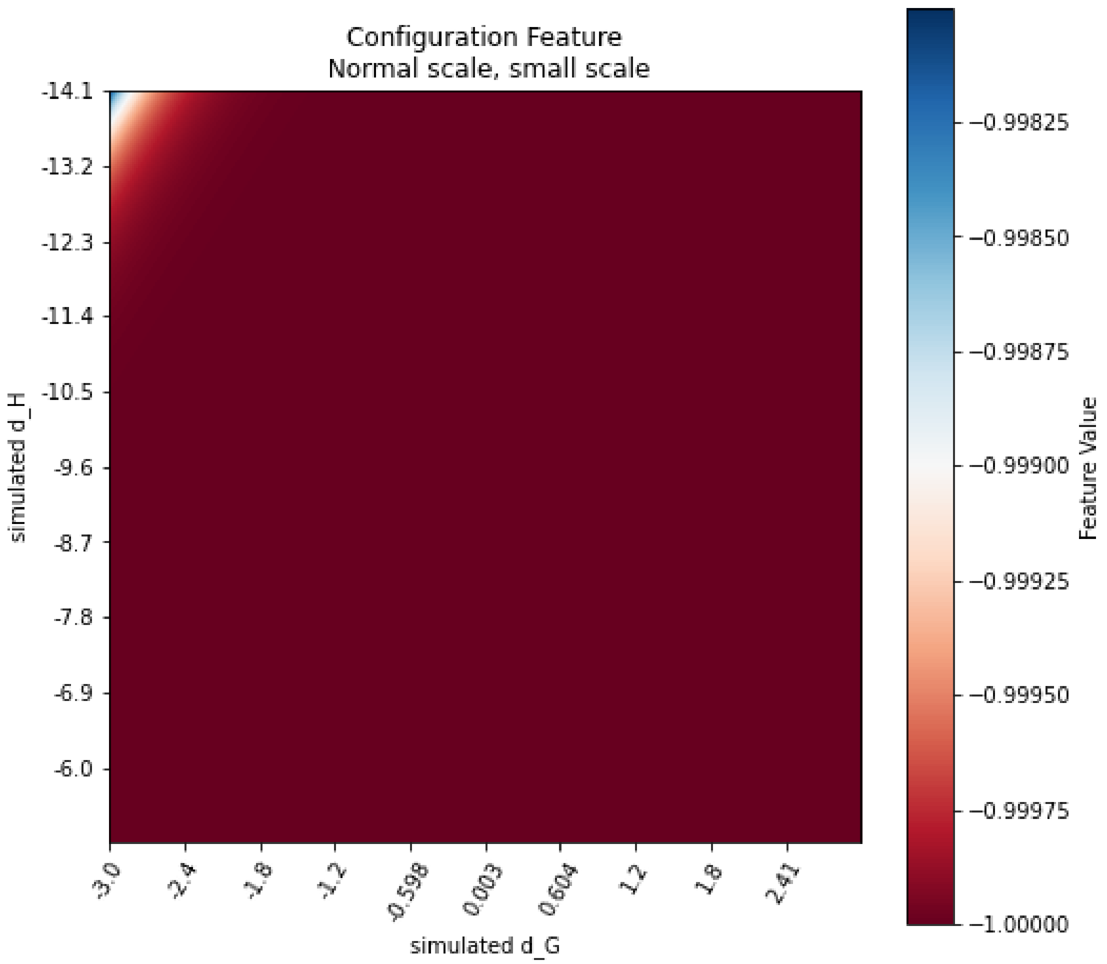
2.1 Transformations, melts, and pivots

3 Simulated feature estimates

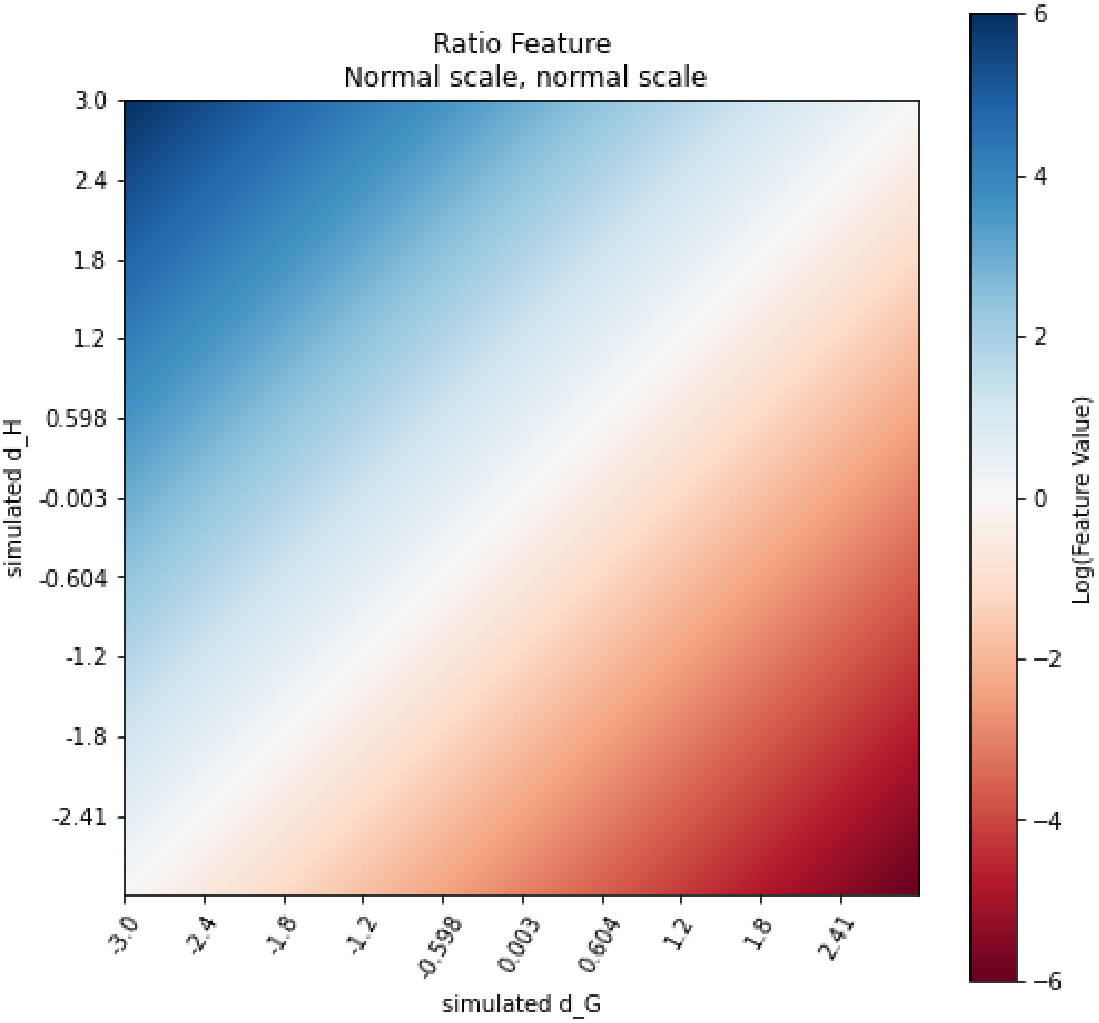
```
[13]: Text(0.5, 1.0, 'Configuration Feature\n Normal scale, small scale')
```

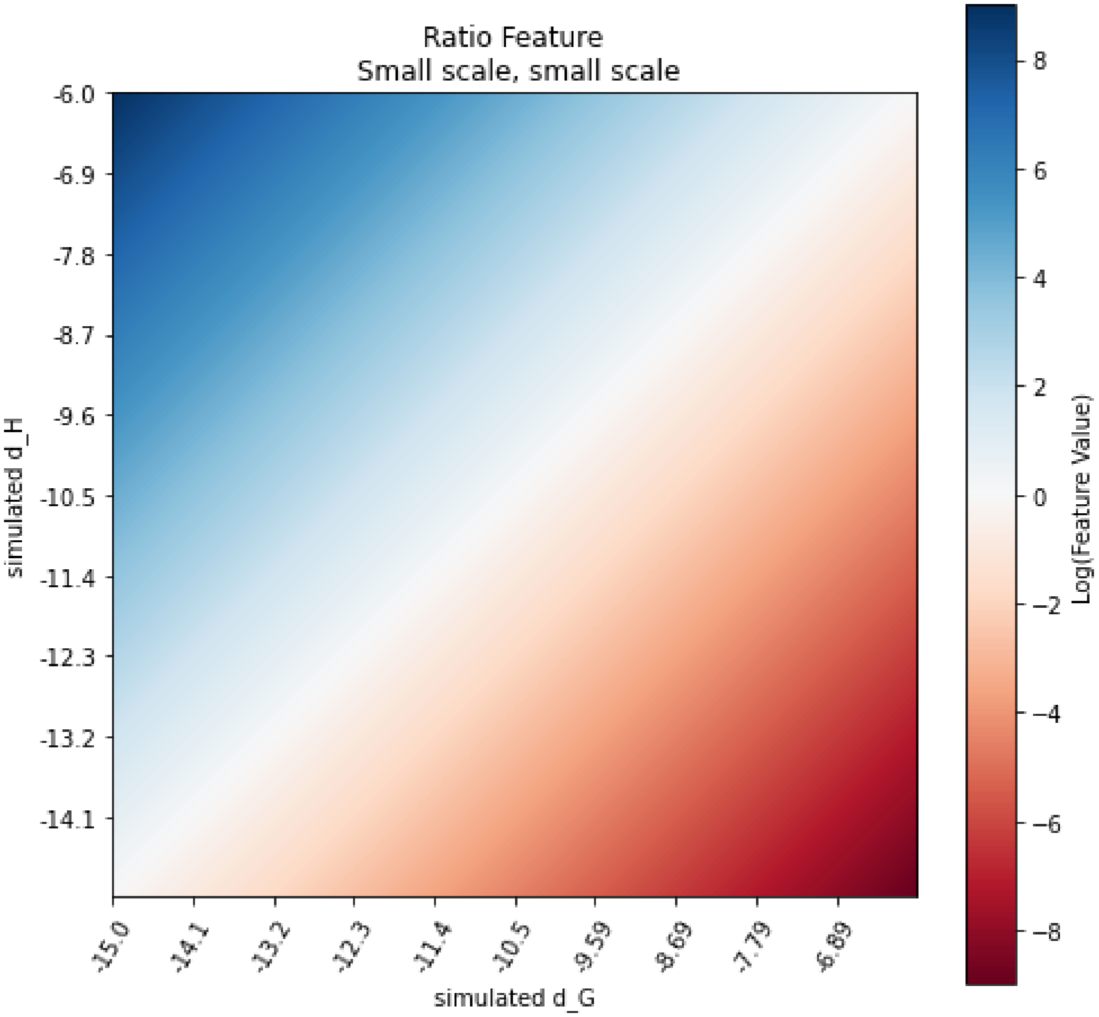

Configuration Feature
Normal scale, normal scale
3.0
2.4
1.8
1.2
0.598
-0.003
-0.604
-1.2
-1.8
-2.41
simulated d_H
-3.0
-2.4
-1.8
-1.2
-0.598
0.003
0.604
1.2
1.8
2.41
simulated d_G
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
Feature Value

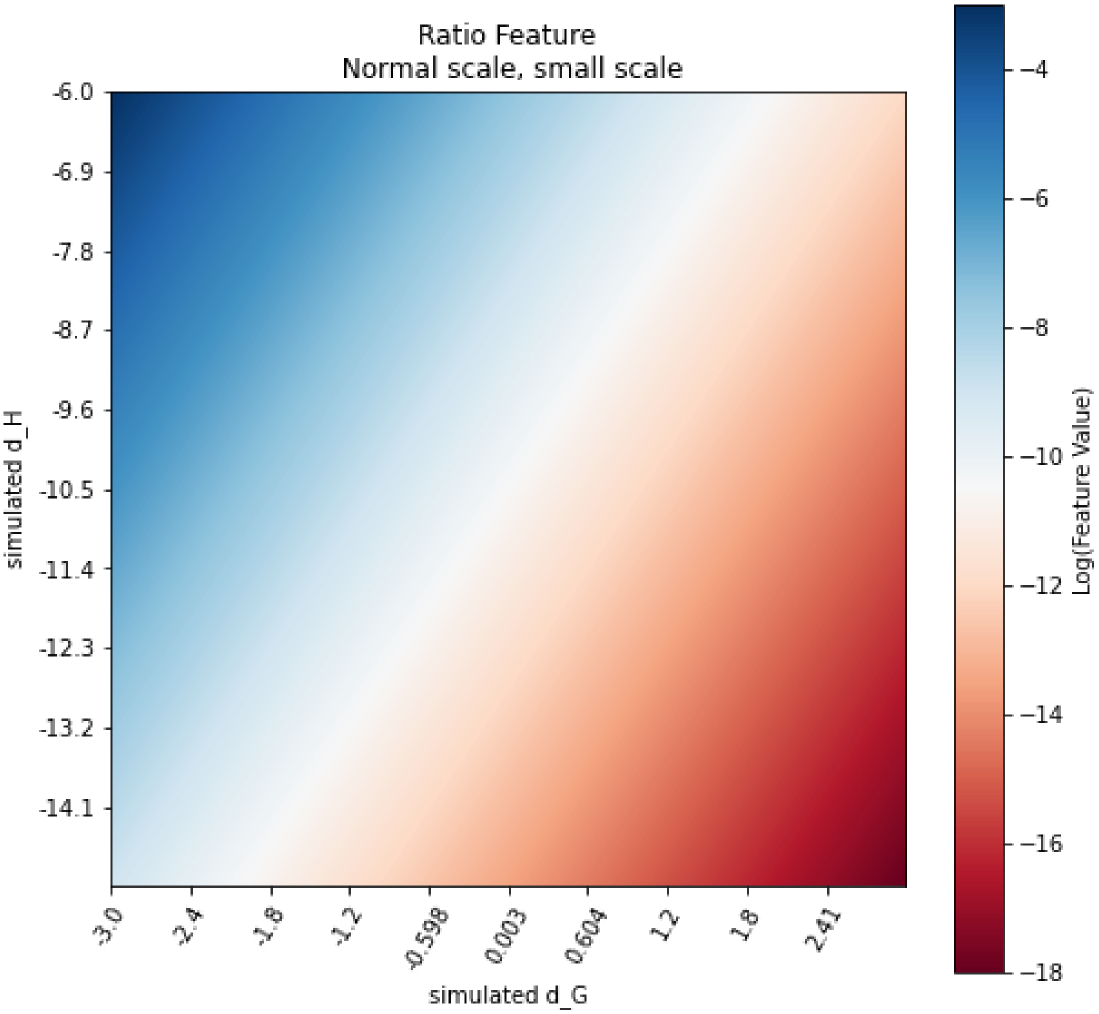

Configuration Feature
Small scale, small scale
-6.0
-6.9
-7.8
-8.7
-9.6
-10.5
-11.4
-12.3
-13.2
-14.1
simulated d_H
-15.0
-14.1
-13.2
-12.3
-11.4
-10.5
-9.59
-8.69
-7.79
-6.89
simulated d_G
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
Feature Value



```
[14]: Text(0.5, 1.0, 'Ratio Feature\n Normal scale, small scale')
```


Ratio Feature
Normal scale, normal scale
3.0
2.4
1.8
1.2
0.598
-0.003
-0.604
-1.2
-1.8
-2.41
simulated d_H
-3.0
-2.4
-1.8
-1.2
-0.598
0.003
0.604
1.2
1.8
2.41
simulated d_G
6
4
2
0
−2
−4
−6
Log(Feature Value)


Ratio Feature
Small scale, small scale
simulated d_H
-6.0
-6.9
-7.8
-8.7
-9.6
-10.5
-11.4
-12.3
-13.2
-14.1
simulated d_G
-15.0
-14.1
-13.2
-12.3
-11.4
-10.5
-9.59
-8.69
-7.79
-6.89
Log(Feature Value)
8
6
4
2
0
−2
−4
−6
−8



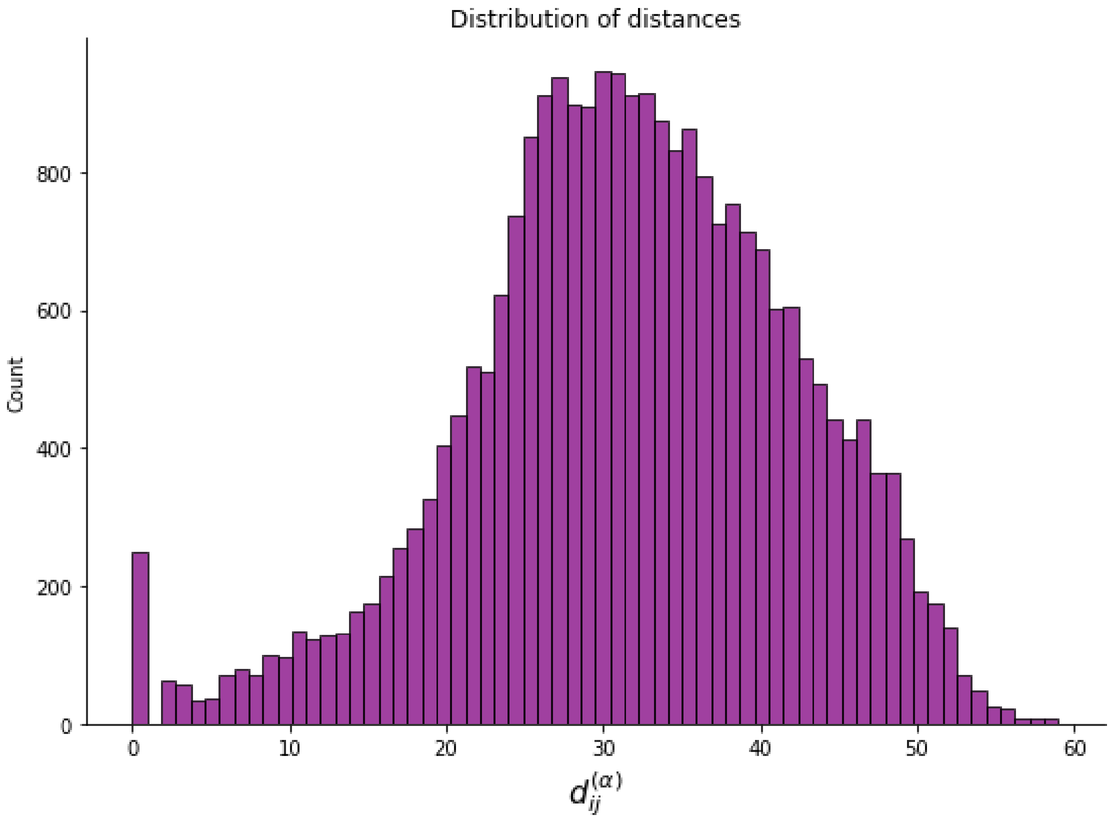
4 Feature visualizations

4.1 Basic distributions

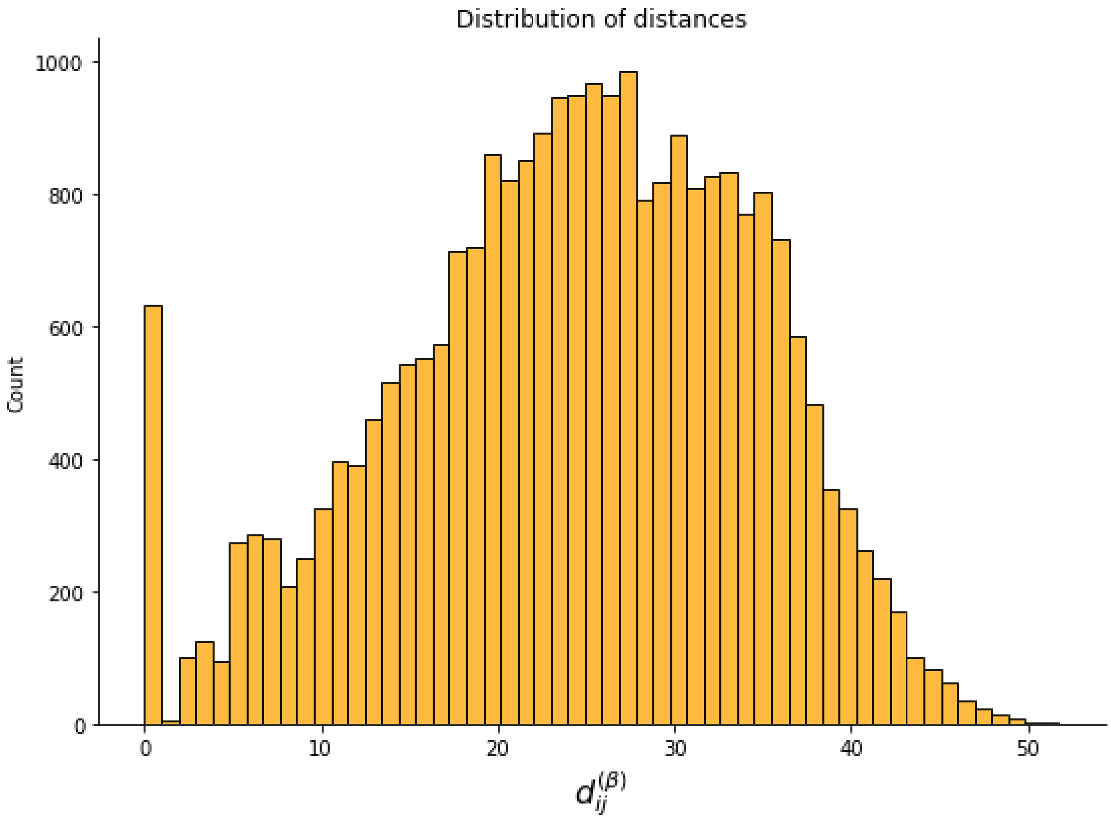
4.1.1 Distance

Univariate

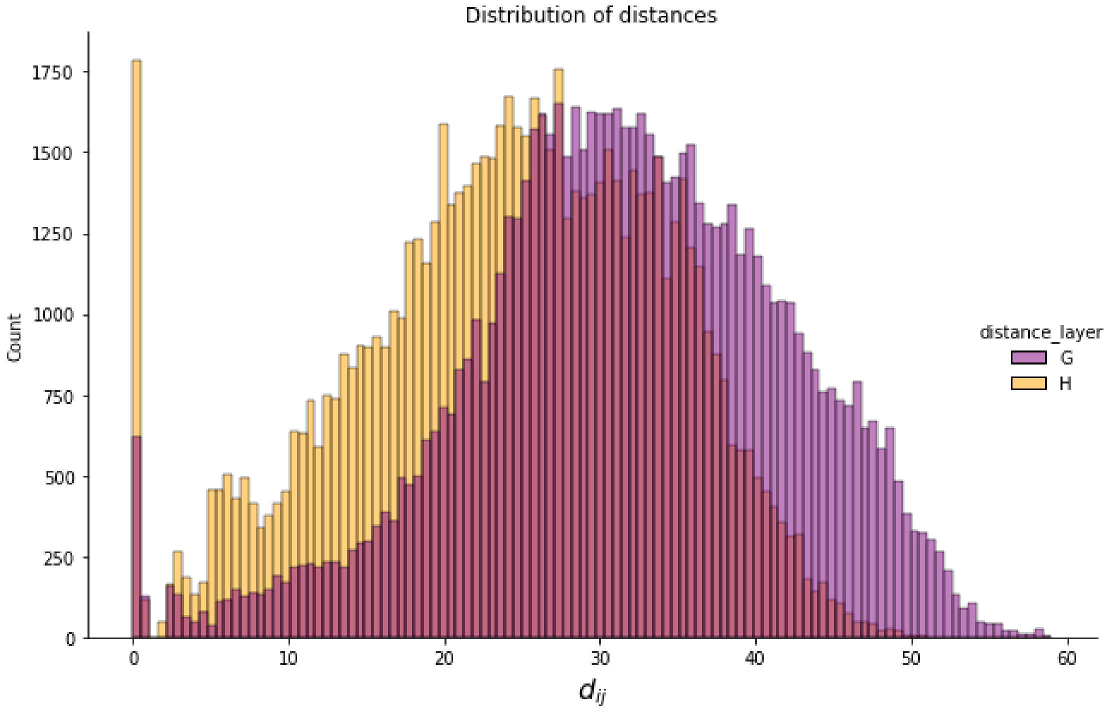
```
<Figure size 432x288 with 0 Axes>
```


Distribution of distances
Count
0
200
400
600
800
0
10
20
30
40
50
60
$d_{ij}^{(\alpha)}$

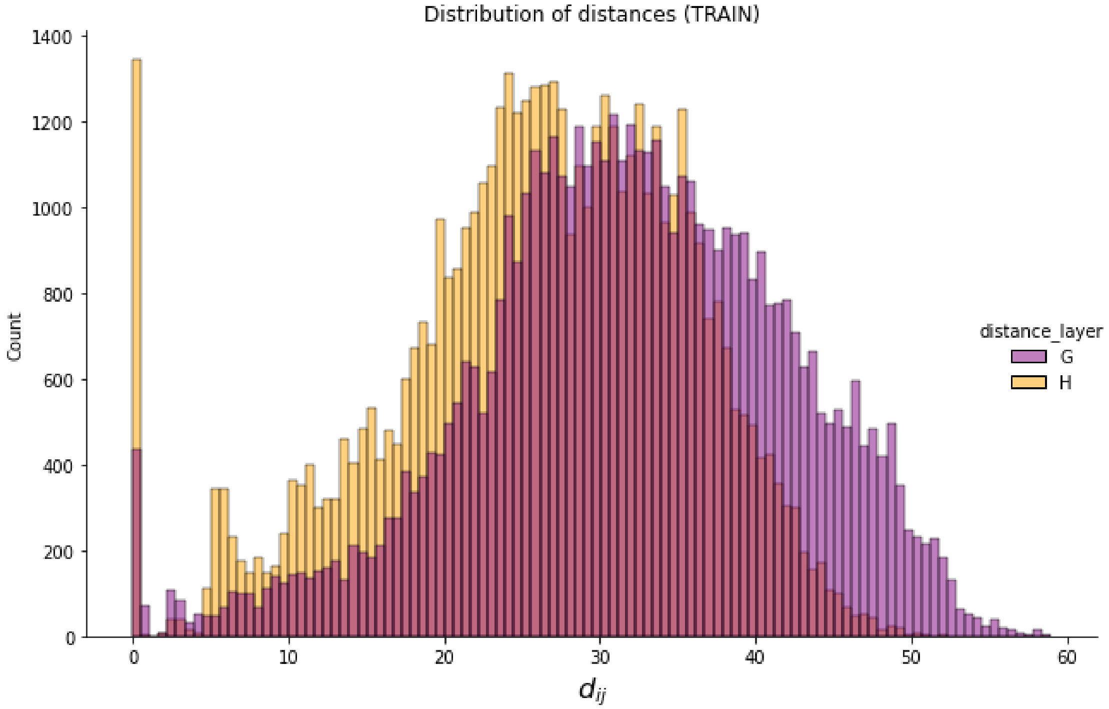
```
<Figure size 432x288 with 0 Axes>
```


Distribution of distances
Count
1000
800
600
400
200
0
0
10
20
30
40
50
$d_{ij}^{(\beta)}$

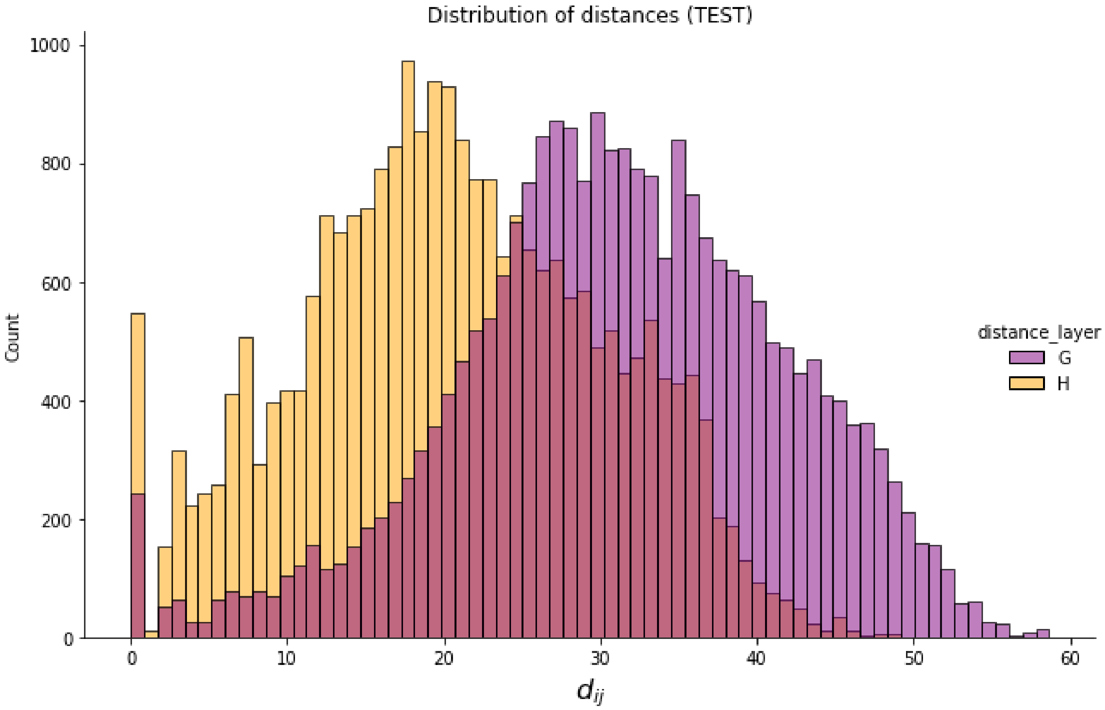
```
<Figure size 432x288 with 0 Axes>
```


Distribution of distances
Count
1750
1500
1250
1000
750
500
250
0
0
10
20
30
40
50
60
d_{ij}
distance_layer
G
H

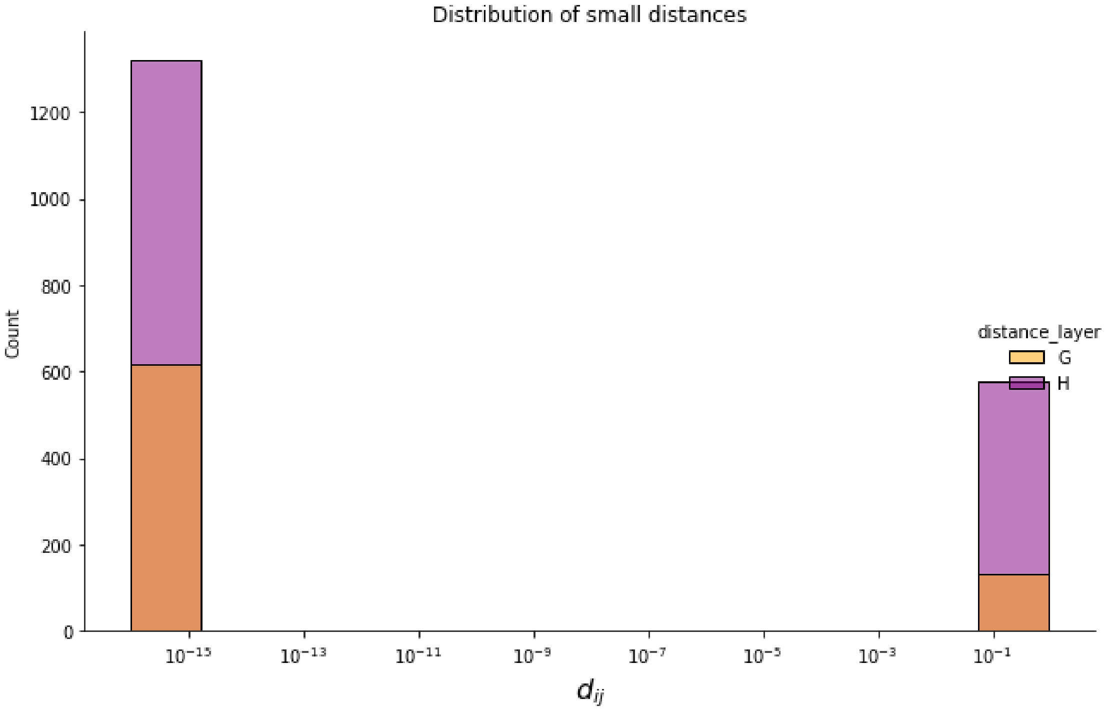
```
<Figure size 432x288 with 0 Axes>
```


Distribution of distances (TRAIN)
Count
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50
60
d_{ij}
distance_layer
G
H

```
<Figure size 432x288 with 0 Axes>
```


Distribution of distances (TEST)
Count
1000
800
600
400
200
0
0
10
20
30
40
50
60
d_{ij}
distance_layer
G
H

```
<Figure size 432x288 with 0 Axes>
```


Distribution of small distances
Count
1200
1000
800
600
400
200
0
10^{-15}
10^{-13}
10^{-11}
10^{-9}
10^{-7}
10^{-5}
10^{-3}
10^{-1}
d_{ij}
distance_layer
G
H

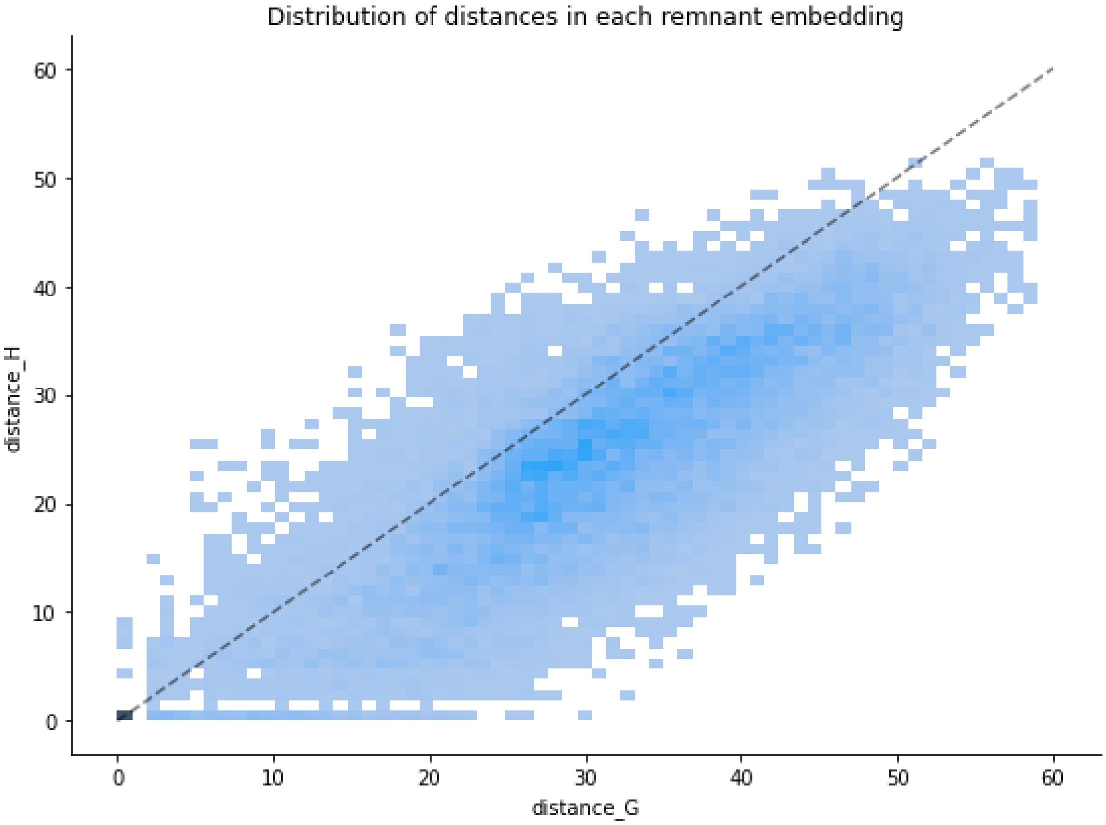
The distribution of distances in each remnant is approximately normal with non-negligible outliers near 0. We will explore later if these are associated with isolated edges or “island” components.

As can be seen from the overlapping histograms, the center of the distributions does appear to be different. We conduct a 2-sample T-test below but mostly for fun, there are like 23,000 degrees of freedom or something obnoxious like that

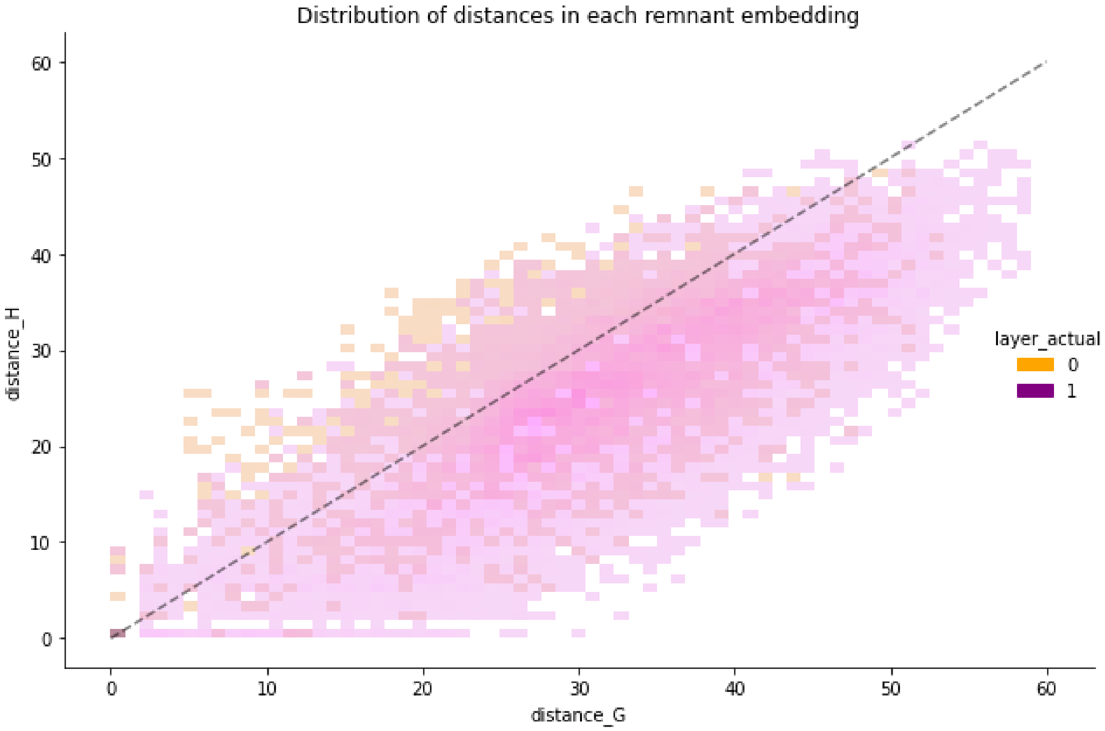
```
d_G variance = 101.4820, d_H variance = 92.4859
Ttest_indResult(statistic=140.89193423942422, pvalue=0.0)
Test p-value = 0.0 ===> With alpha = 0.05, we CAN reject the null hypothesis
that the two distributions have equal mean.
```

Bivariate

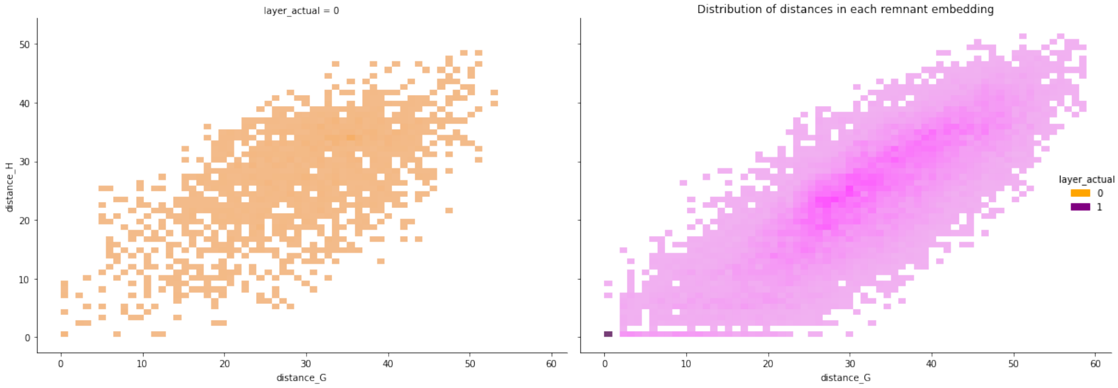
```
<Figure size 576x432 with 0 Axes>
```



```
<Figure size 576x432 with 0 Axes>
```



```
<Figure size 576x432 with 0 Axes>
```

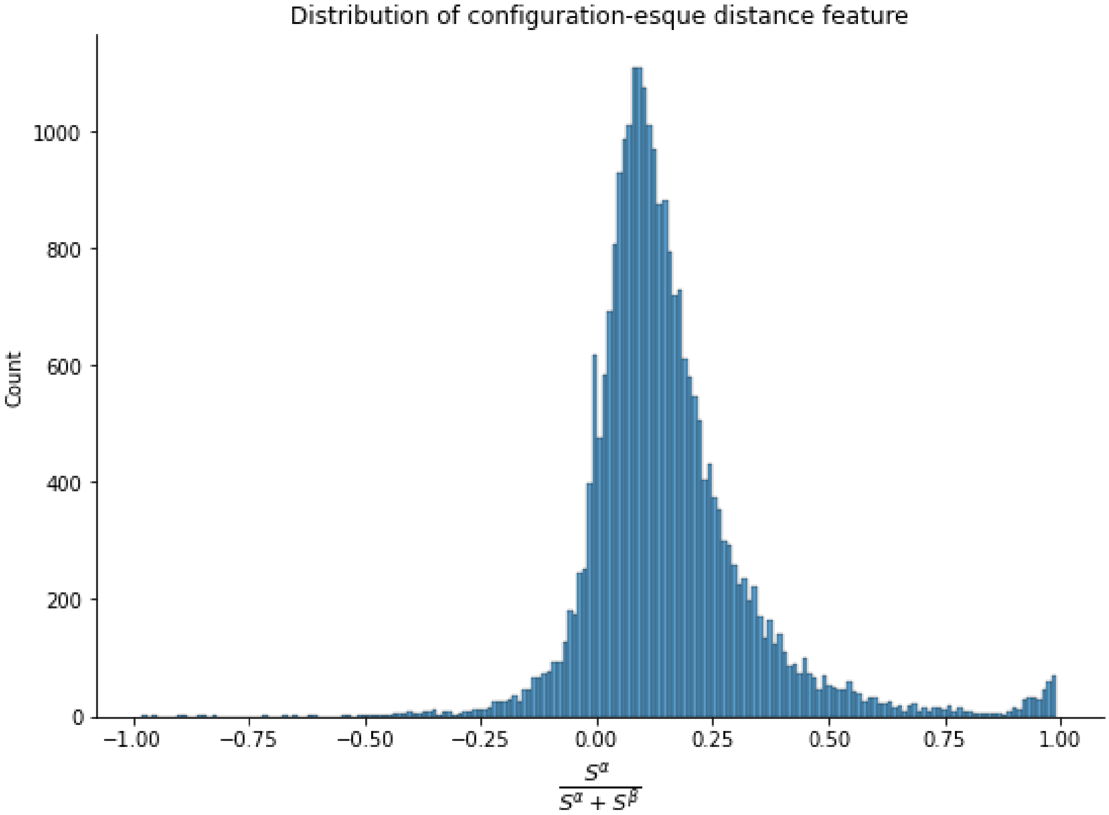


With the simplest embedding information (raw distance per layer) as above, the layers do not appear to be particularly separable. In fact, with these quantities the entirety of classification performance seems, by eye, to due to class imbalance. The absolute sizes of the classes is given below.

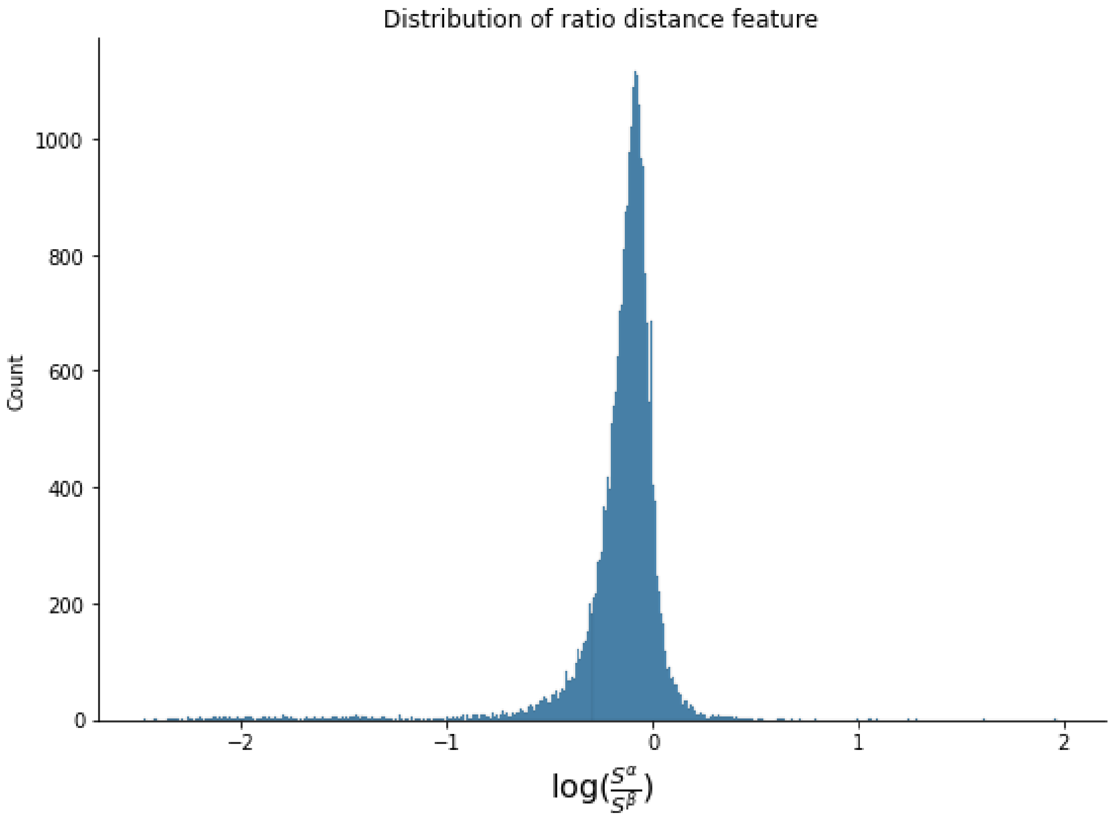
```
G contains 93% of all edges
```

4.1.2 Features

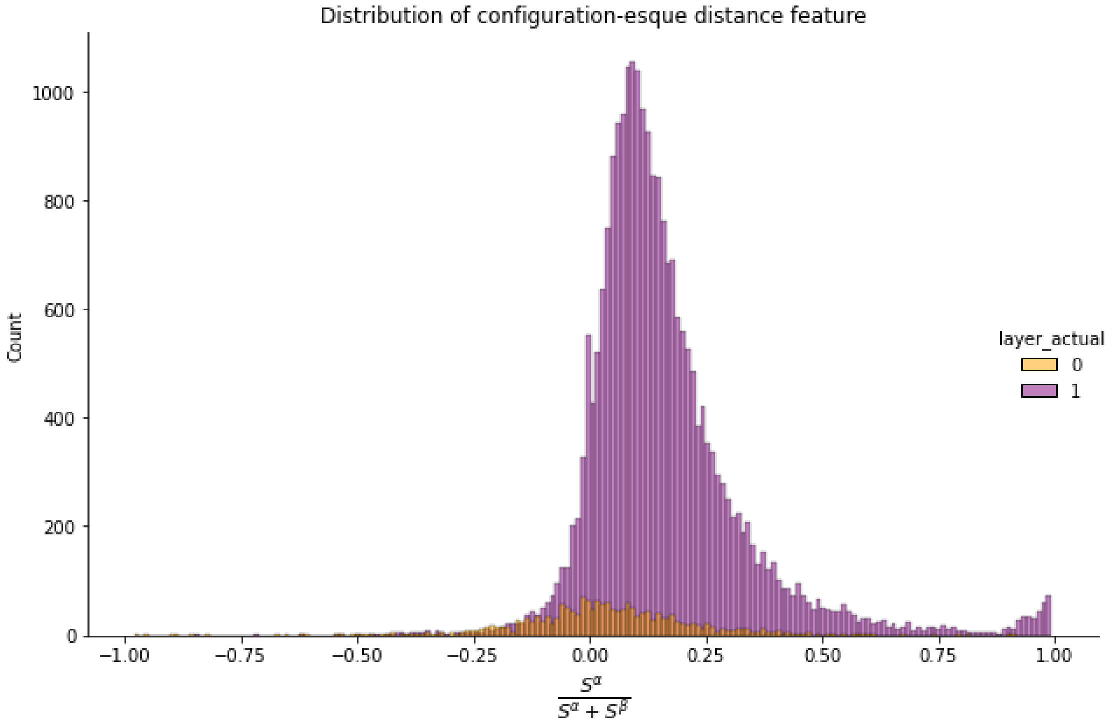
```
<Figure size 432x288 with 0 Axes>
```



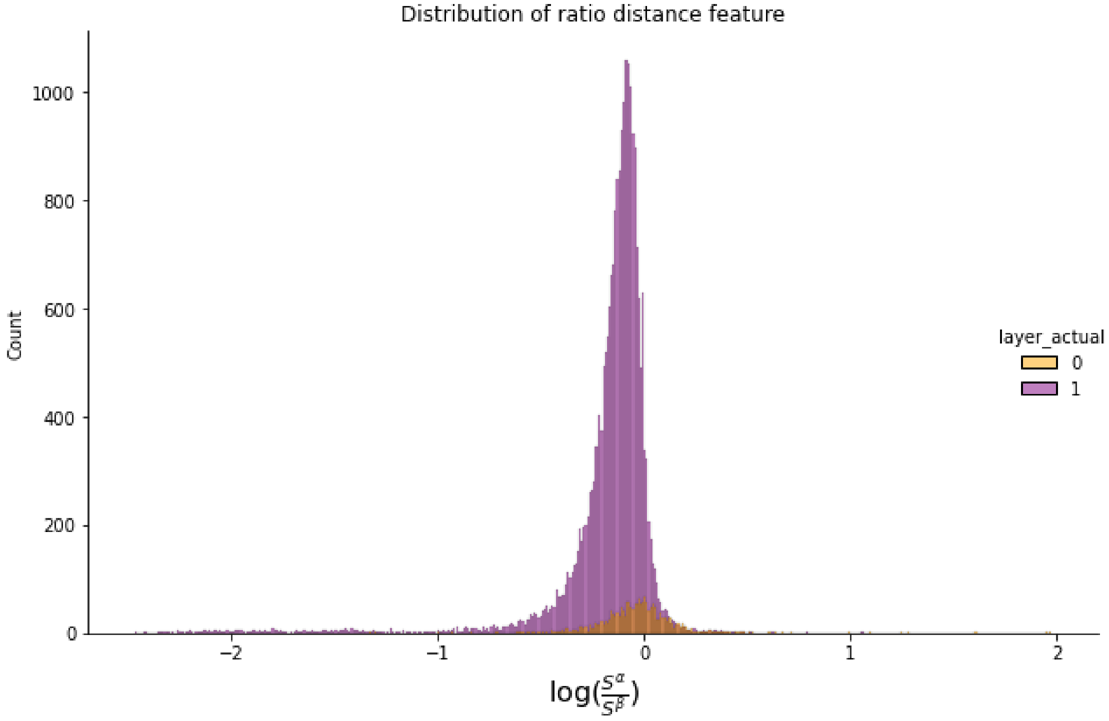
```
<Figure size 432x288 with 0 Axes>
```


Distribution of ratio distance feature
Count
1000
800
600
400
200
0
−2
−1
0
1
2
$\log(\frac{S^{\alpha}}{S^{\beta}})$

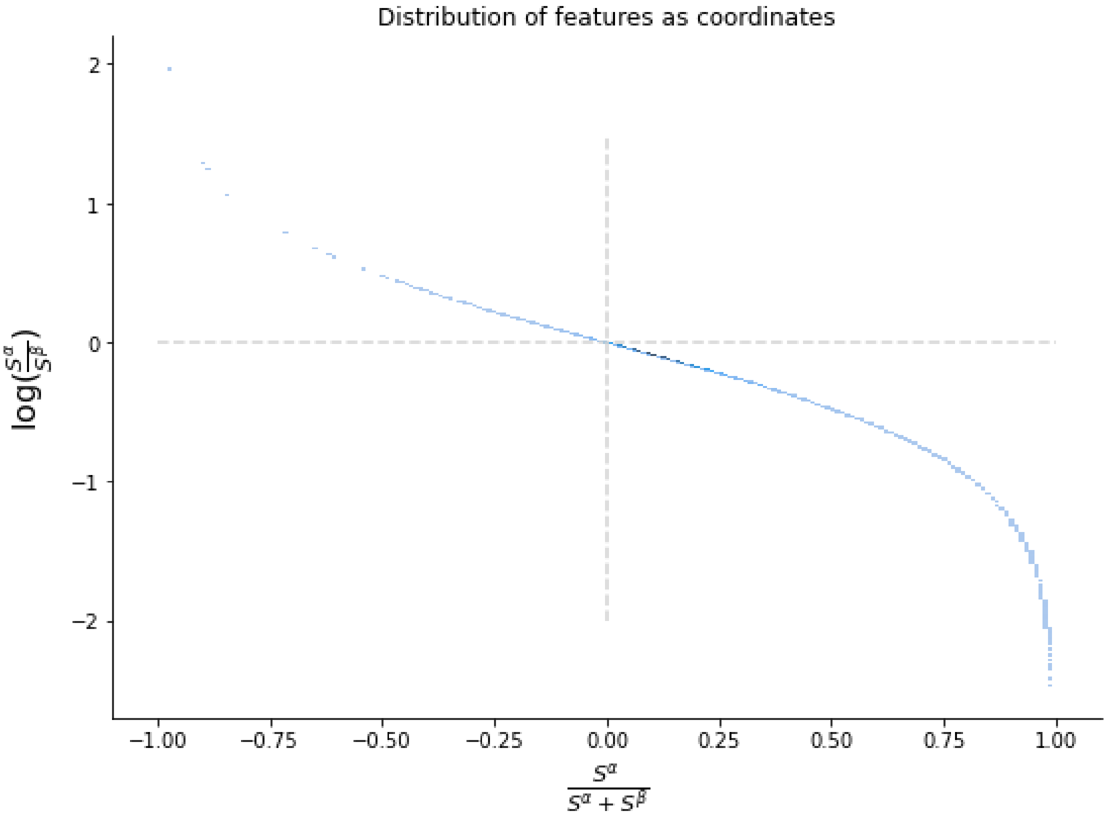
```
<Figure size 432x288 with 0 Axes>
```


Distribution of configuration-esque distance feature
Count
0
200
400
600
800
1000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
$\frac{S^\alpha}{S^\alpha + S^\beta}$
layer_actual
0
1

```
<Figure size 432x288 with 0 Axes>
```

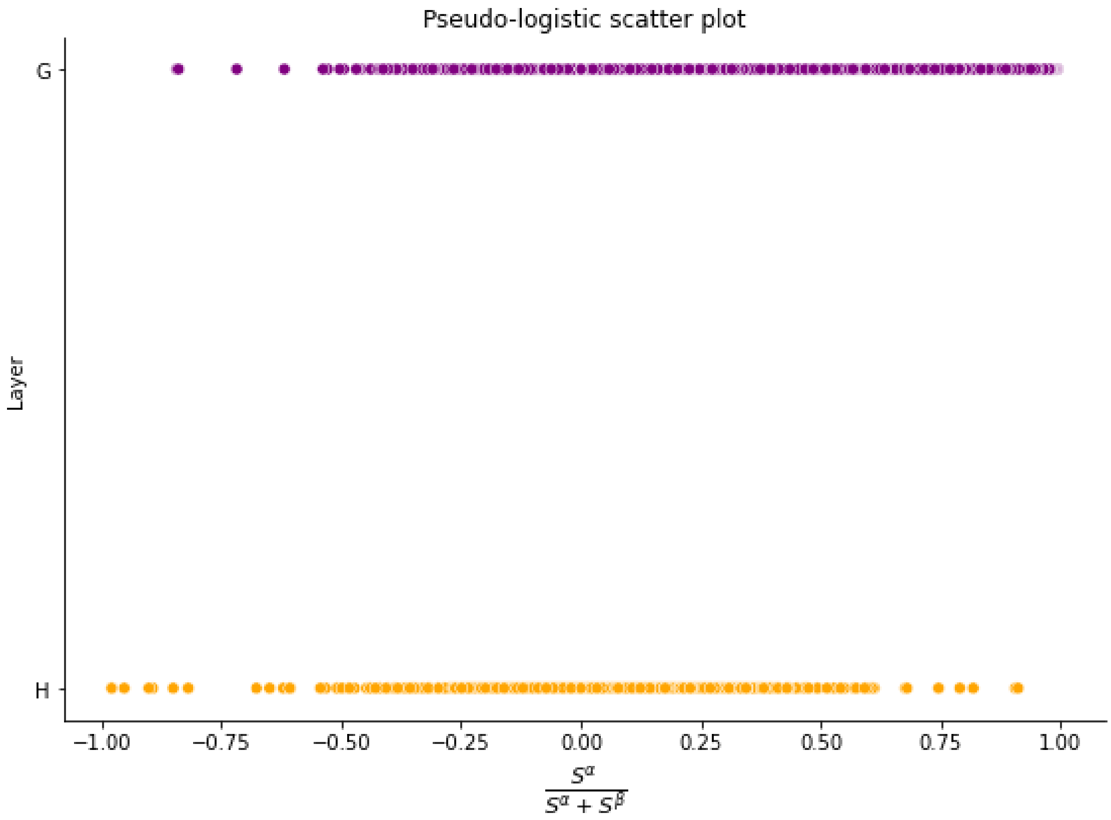

Distribution of ratio distance feature
Count
0
200
400
600
800
1000
−2
−1
0
1
2
$\log(\frac{S^\alpha}{S^\beta})$
layer_actual
0
1

```
<Figure size 576x432 with 0 Axes>
```

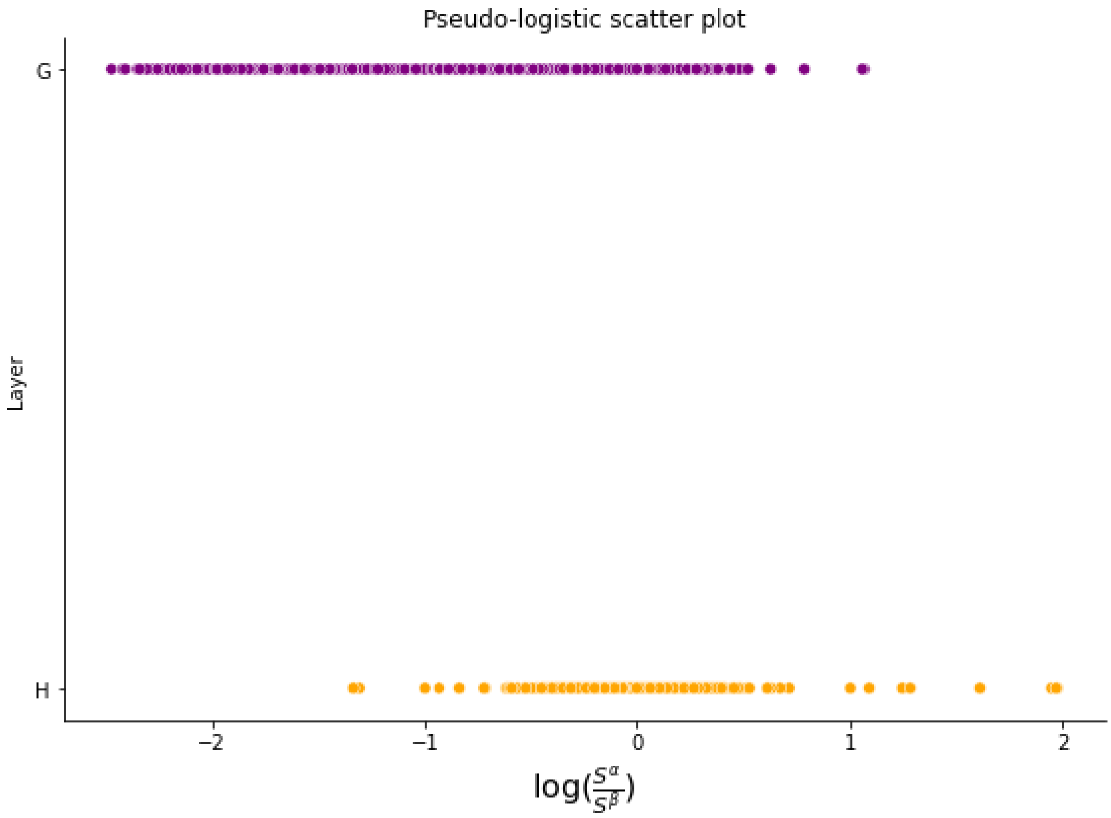


4.2 Class separability

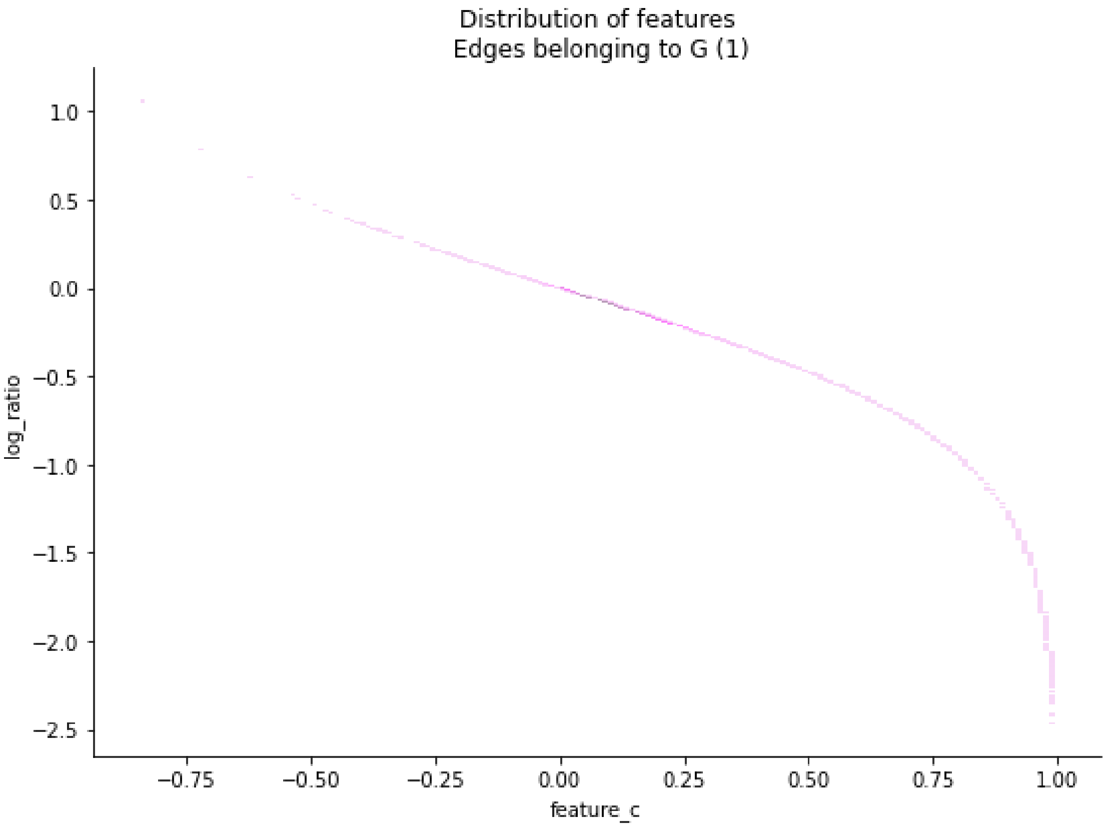
```
<Figure size 432x288 with 0 Axes>
```


Pseudo-logistic scatter plot
Layer
G
H
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
$\frac{S^{\alpha}}{S^{\alpha}+S^{\beta}}$

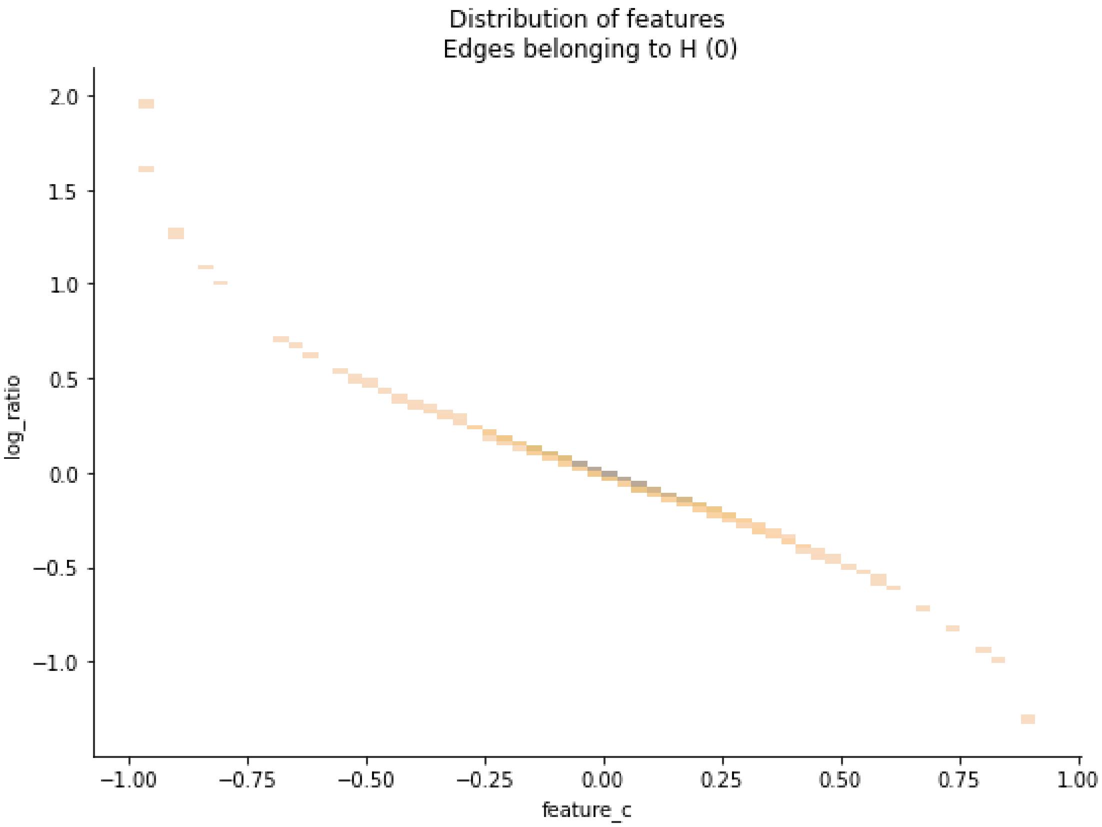
```
<Figure size 432x288 with 0 Axes>
```


Pseudo-logistic scatter plot
G
H
Layer
−2
−1
0
1
2
$\log(\frac{S^\alpha}{S^\beta})$

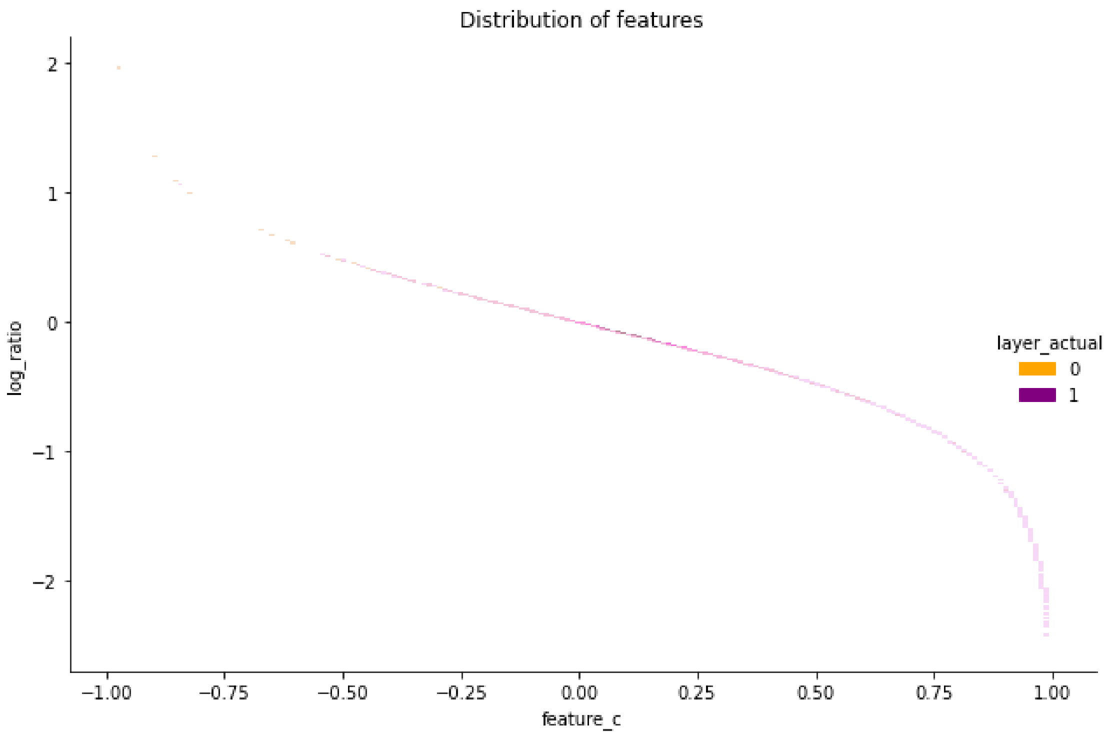
```
<Figure size 576x432 with 0 Axes>
```


Distribution of features
Edges belonging to G (1)
log_ratio
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
feature_c

```
<Figure size 576x432 with 0 Axes>
```


Distribution of features
Edges belonging to H (0)
log_ratio
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
feature_c

```
<Figure size 576x432 with 0 Axes>
```



4.3 Logistic regression models

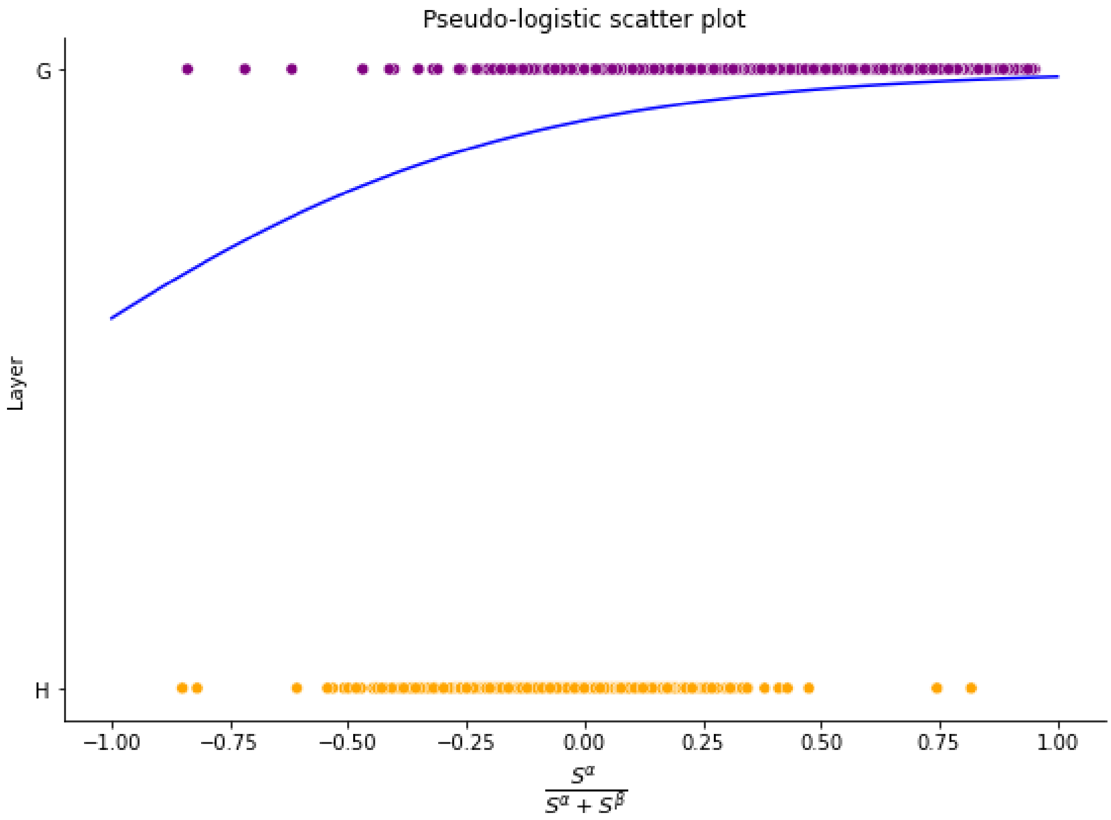
4.3.1 Configuration feature

```
[30]: LogisticRegression()
```

```
Simple score: 0.9279
```

Note that the simple score very closely matches the layer imbalance

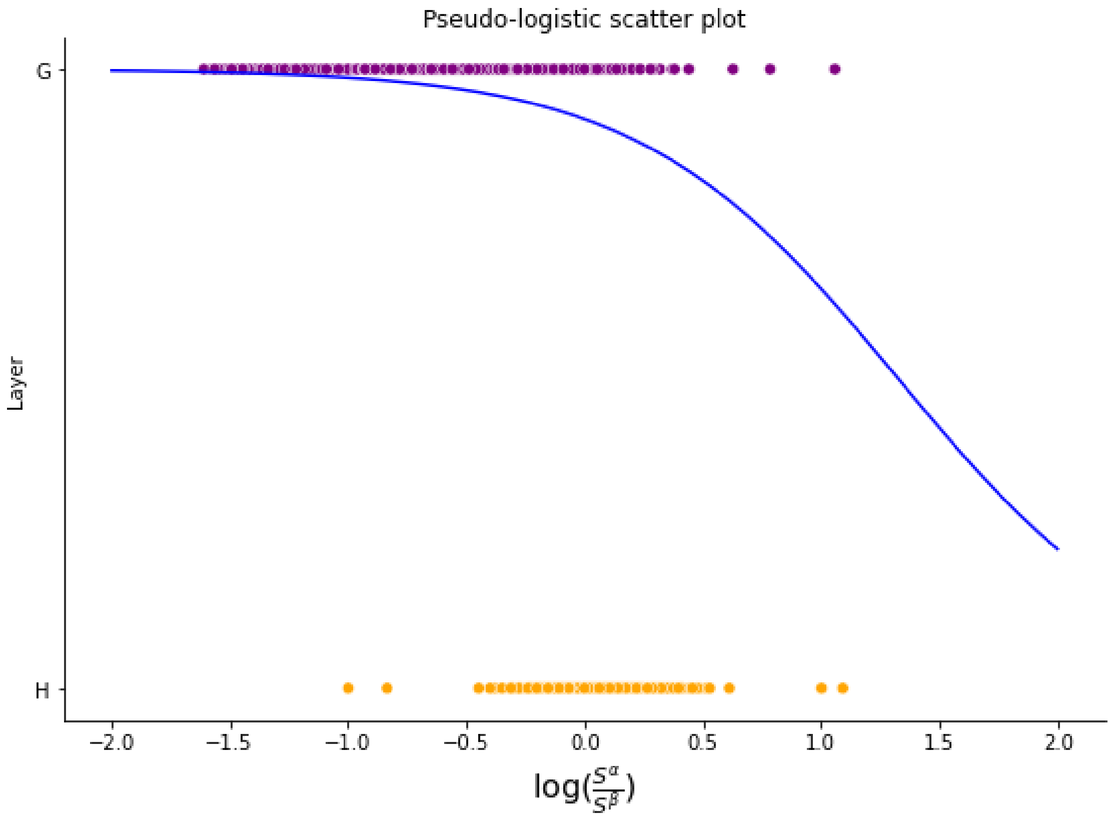
```
<Figure size 432x288 with 0 Axes>
```



4.3.2 Ratio feature

```
[36]: LogisticRegression()

<Figure size 432x288 with 0 Axes>
```

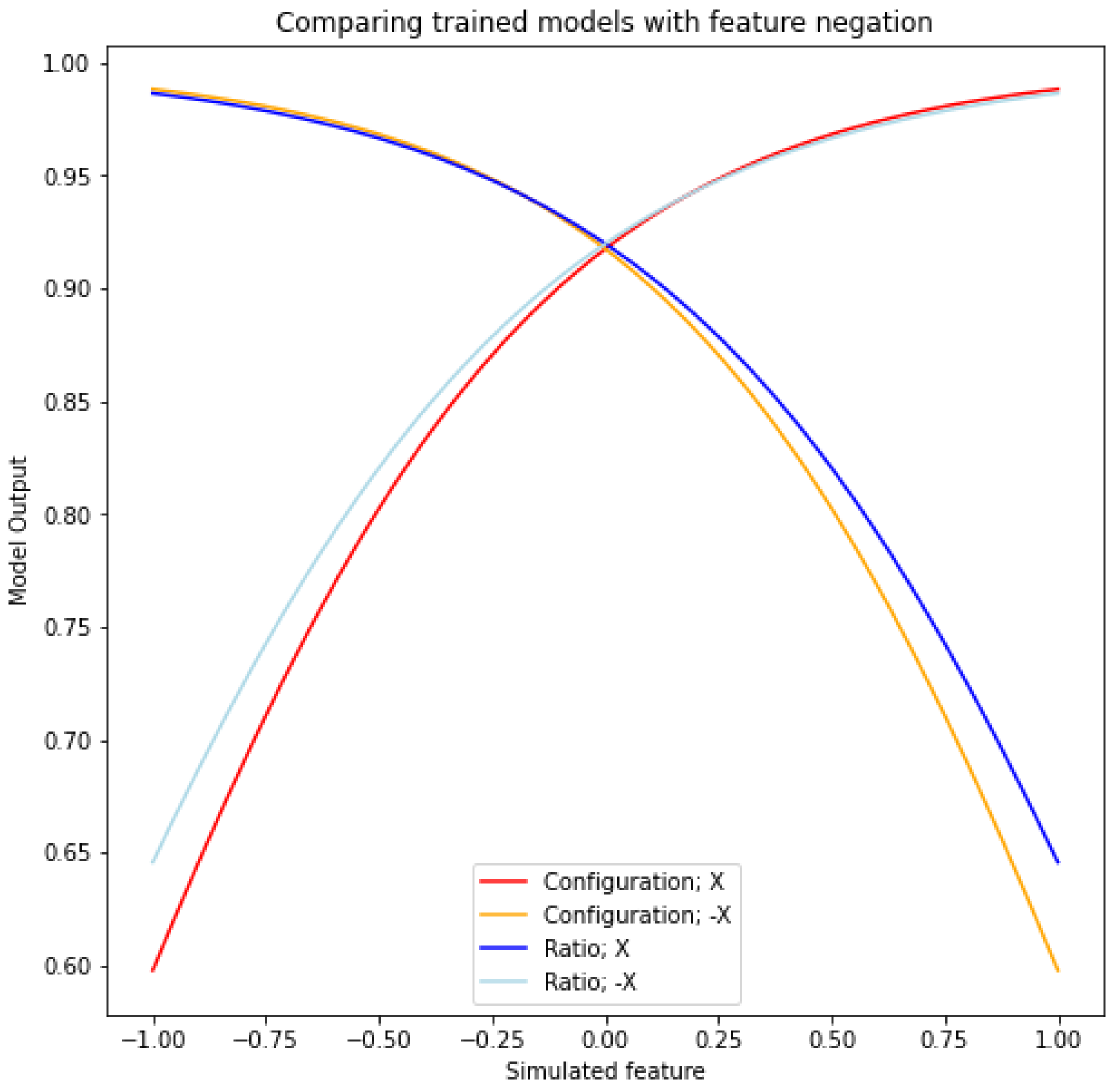


4.4 Model similarities

```
[40]: LogisticRegression()
```

```
Configuration model : Intercept = 2.4067; Coefficient = 2.0101
Ratio model : Intercept = 2.4378; Coefficient = -1.8375
```

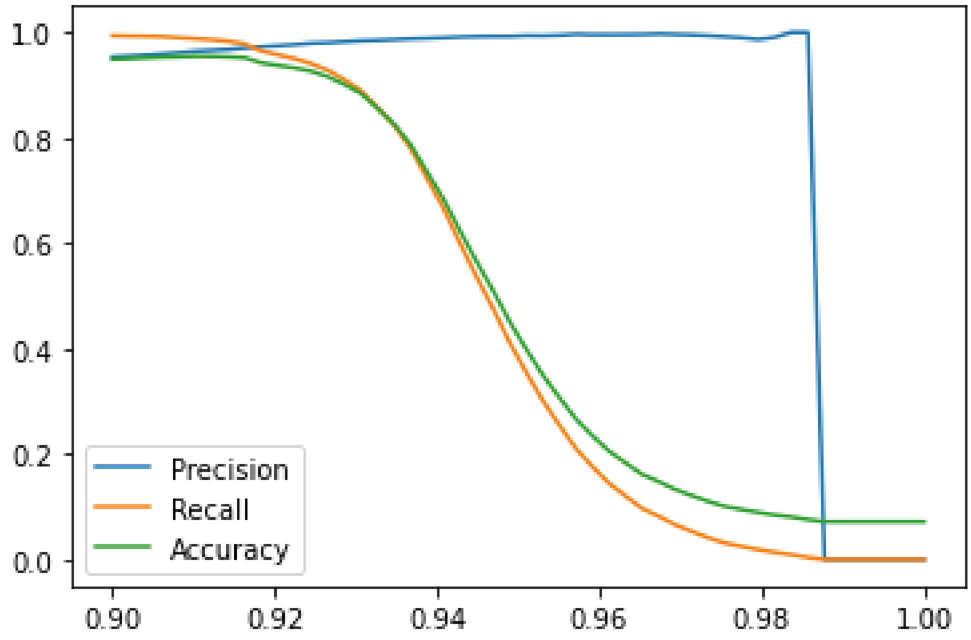
```
[43]: Text(0.5, 1.0, 'Comparing trained models with feature negation')
```



```
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
```

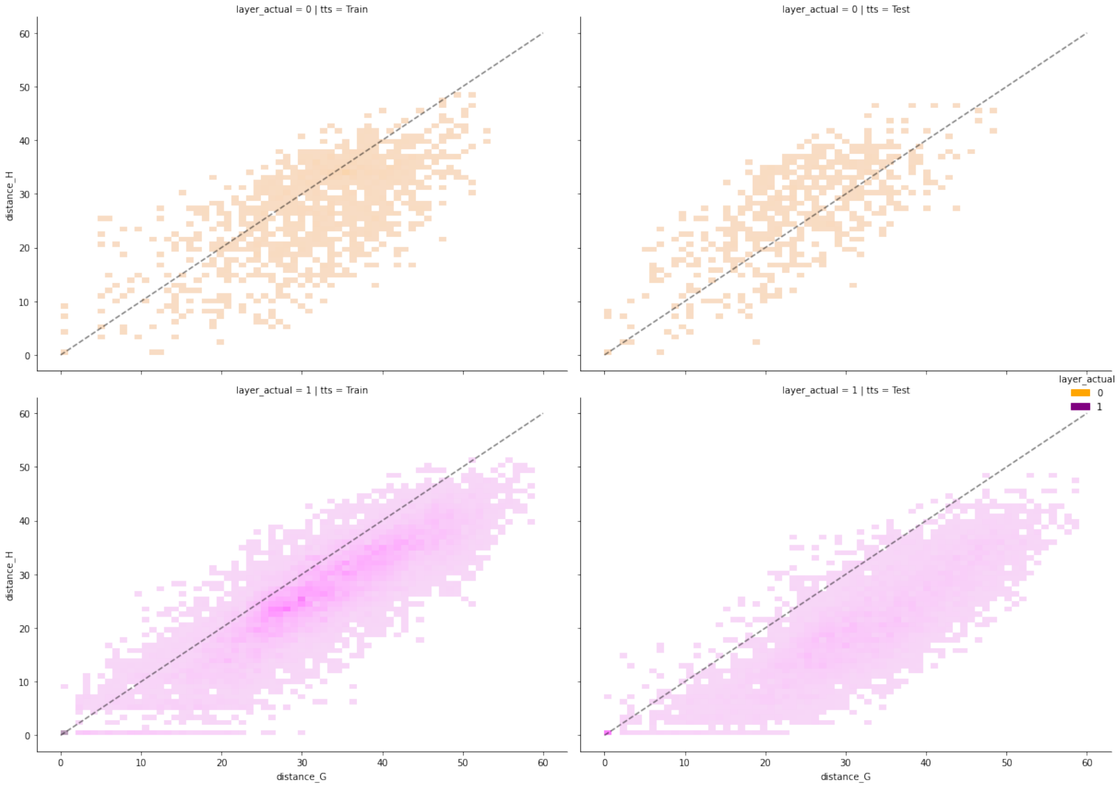
```
  _warn_prf(average, modifier, msg_start, len(result))
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/daniel/.conda/envs/EmbeddedNaive/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[46]: <matplotlib.legend.Legend at 0x7feab89f7fd0>
```

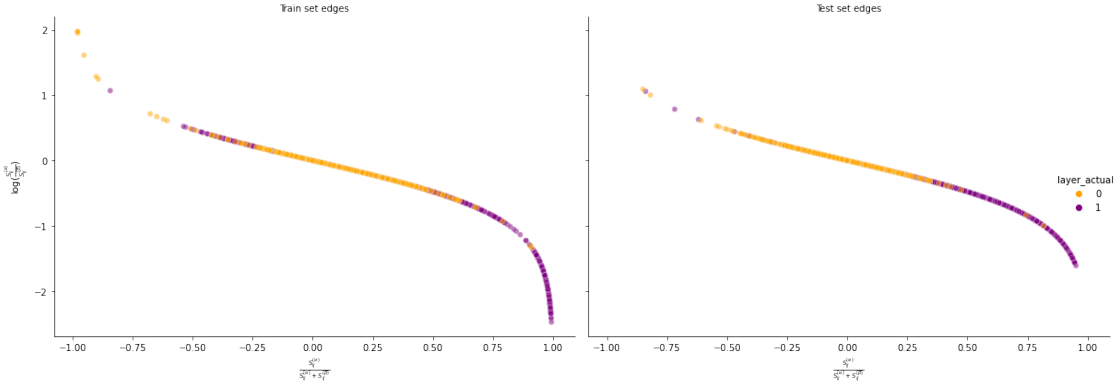


4.5 TTS Visualizations

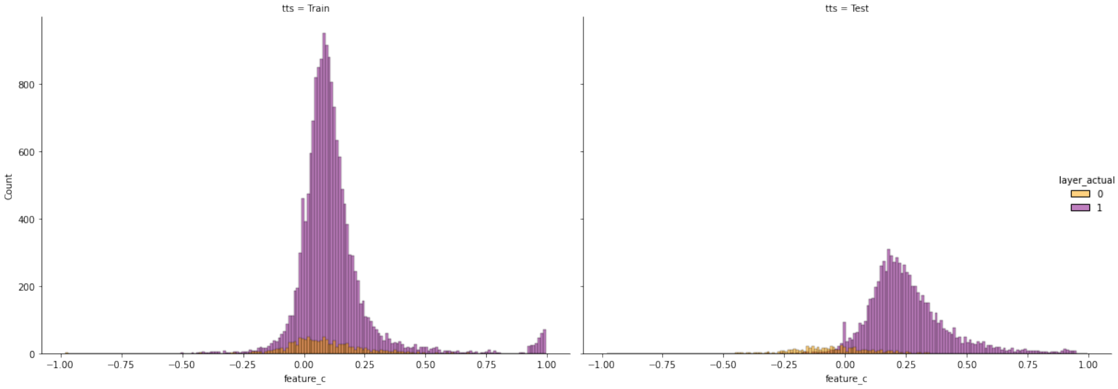
```
<Figure size 576x432 with 0 Axes>
```



```
<Figure size 576x432 with 0 Axes>
```



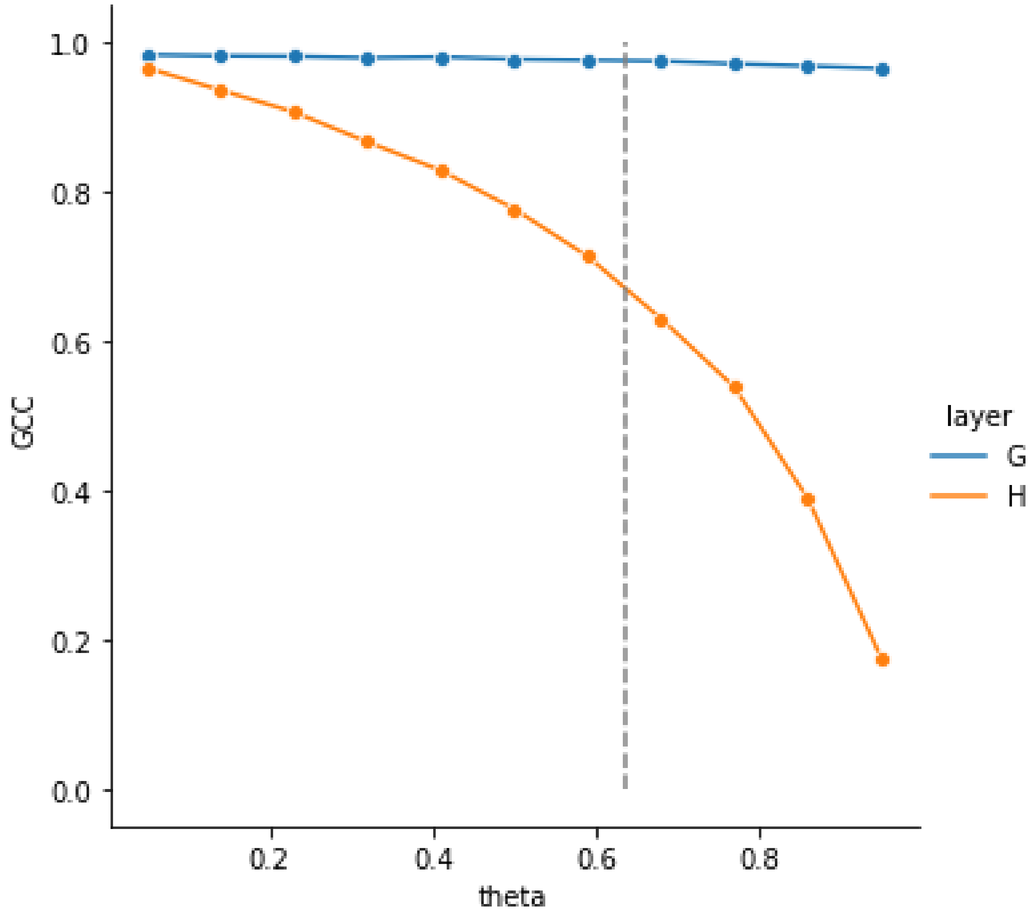
```
<Figure size 432x288 with 0 Axes>
```

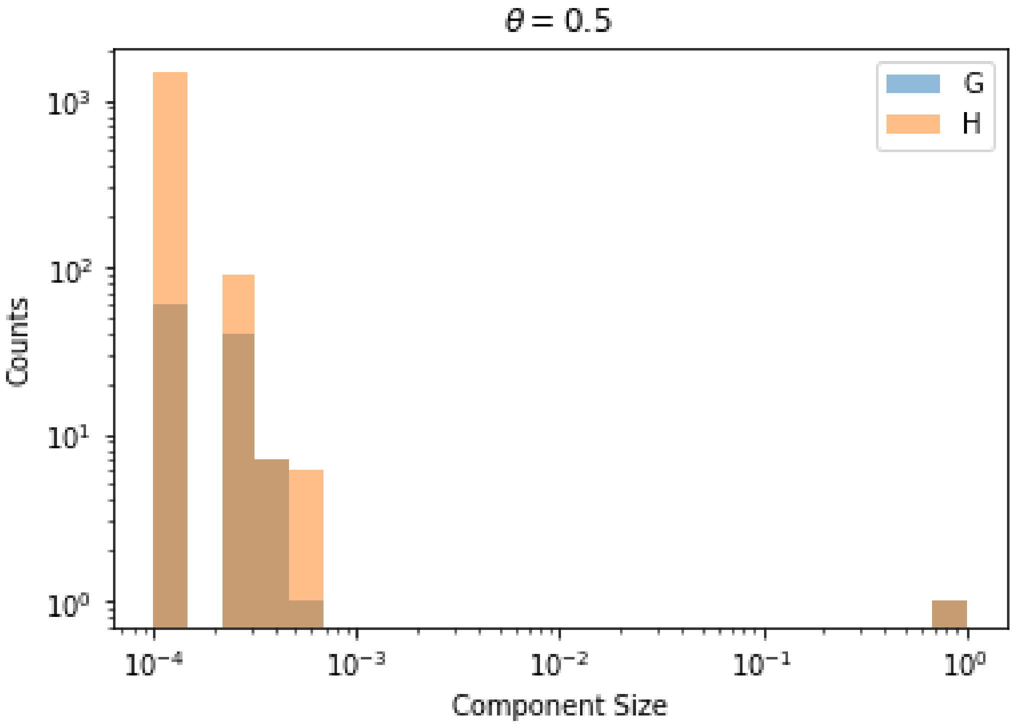


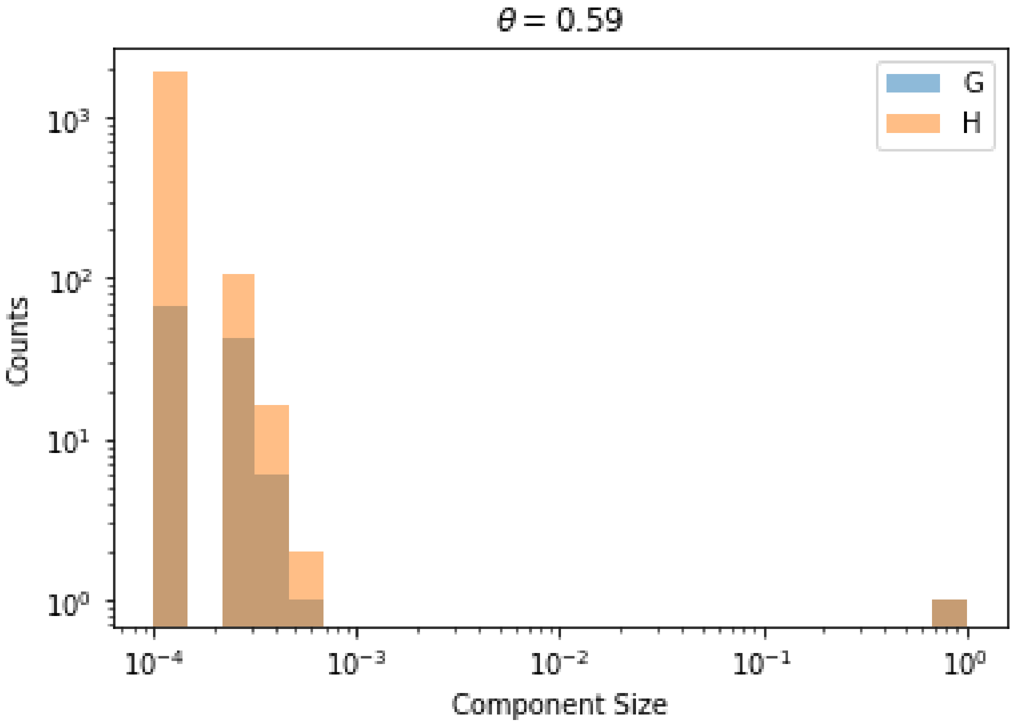
5 Network analysis

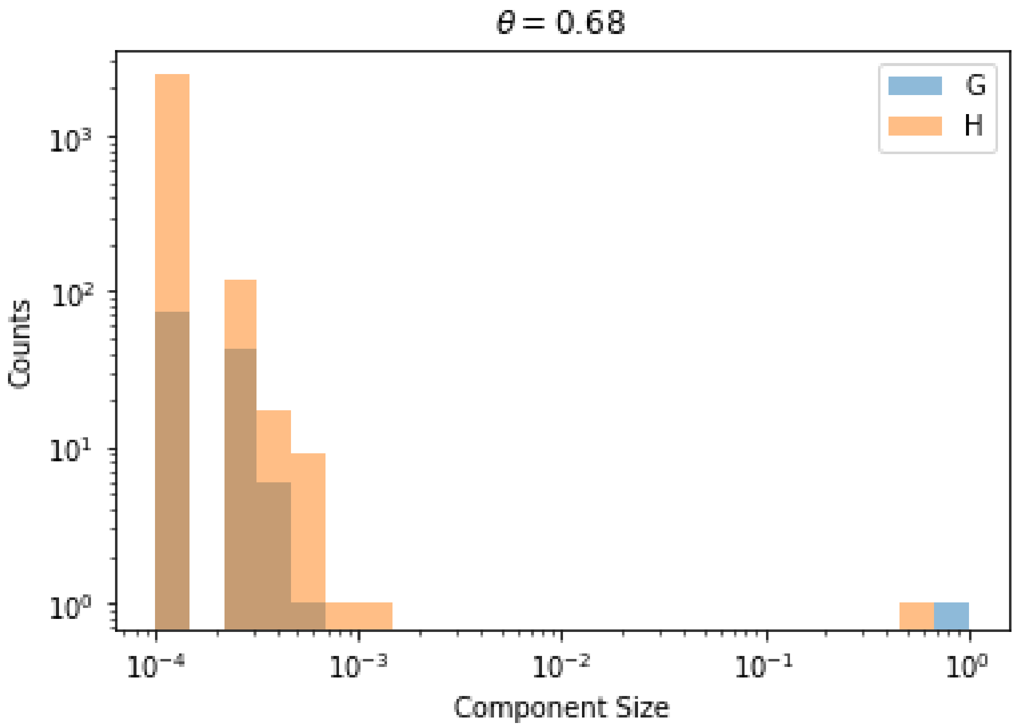
```
/home/daniel/.local/lib/python3.10/site-
packages/pandas/core/array_algos/replace.py:85: FutureWarning: elementwise
comparison failed; returning scalar instead, but in the future will perform
elementwise comparison
  op = lambda x: operator.eq(x, b)
```

```
[51]: <matplotlib.collections.LineCollection at 0x7feab8082f80>
```


1.0
0.8
0.6
0.4
0.2
0.0
GCC
0.2
0.4
0.6
0.8
theta
layer
G
H

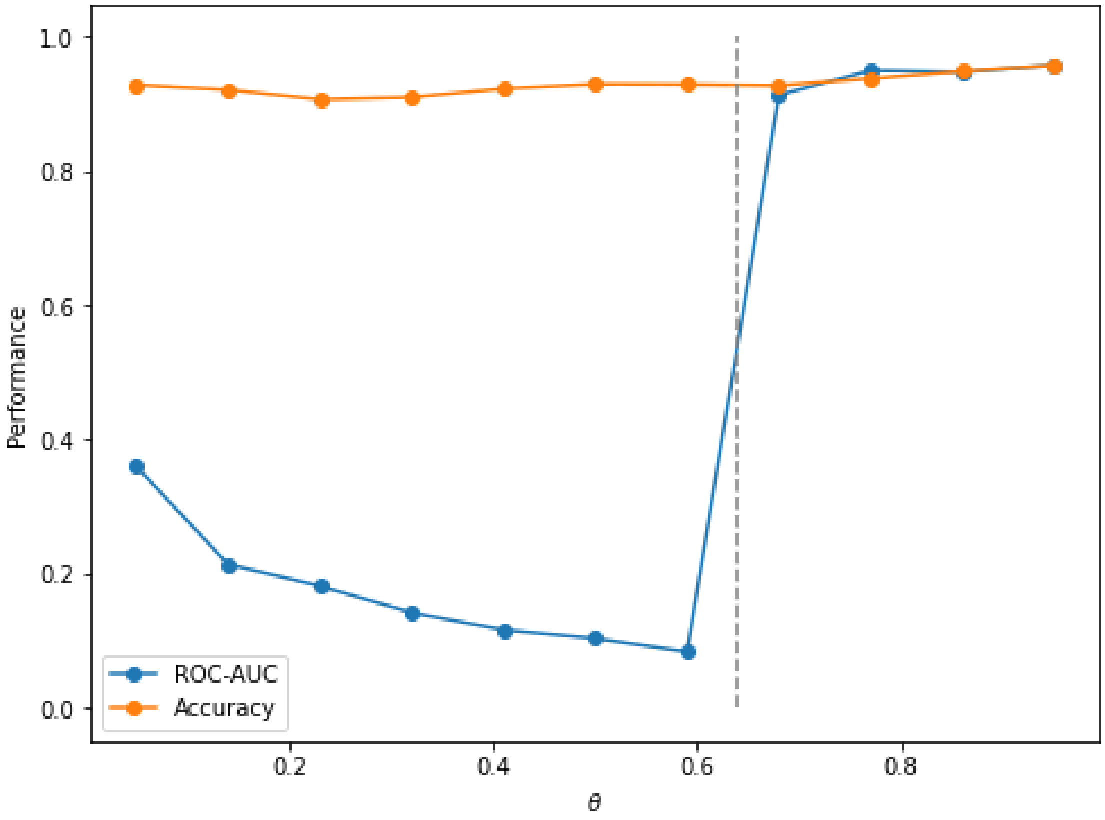

$\theta = 0.5$
G
H
Counts
10^0
10^1
10^2
10^3
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0
Component Size

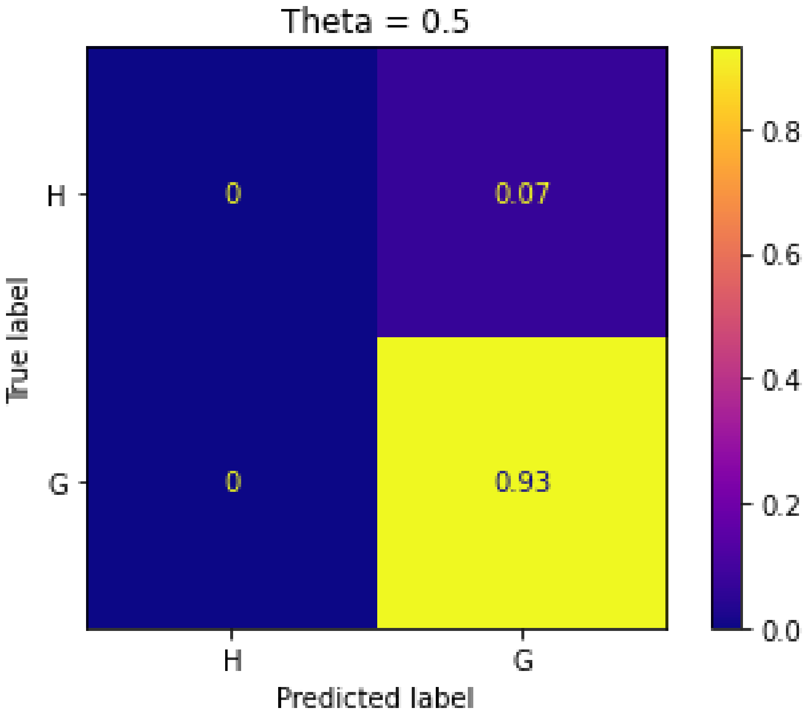

$\theta = 0.59$
G
H
Counts
10^0
10^1
10^2
10^3
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0
Component Size

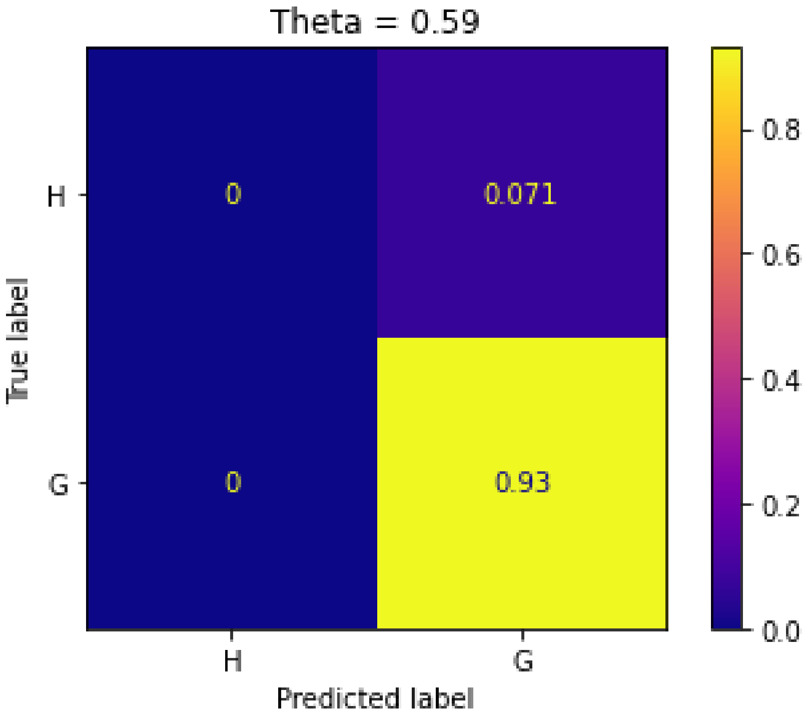


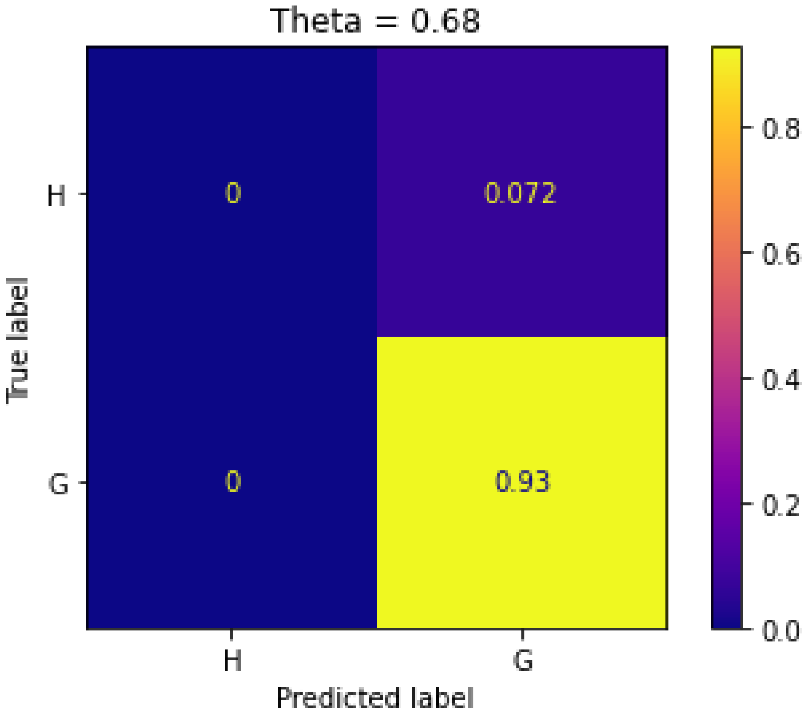
6 Performance versus PFI

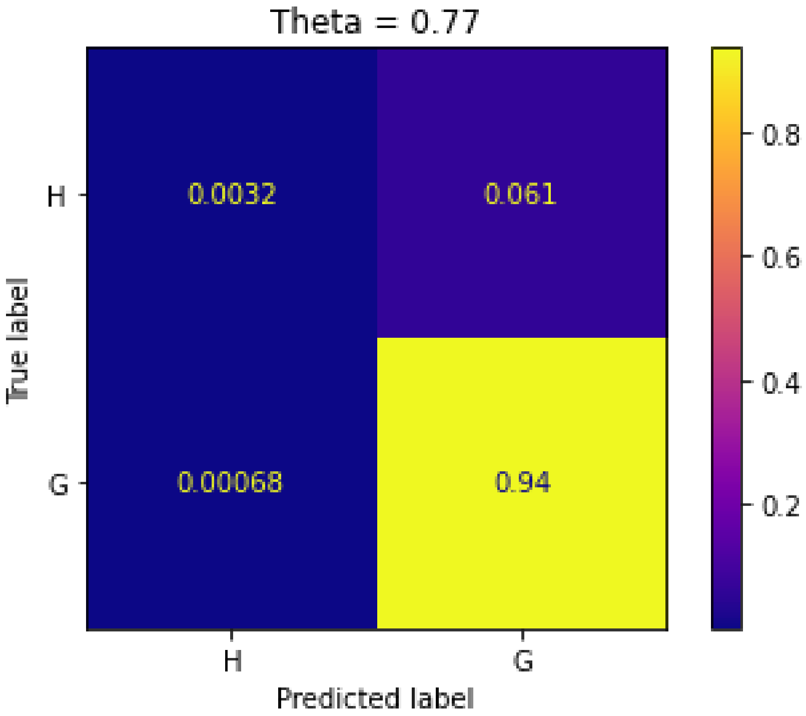
```
[63]: <matplotlib.collections.LineCollection at 0x7feab8877700>
```


1.0
0.8
0.6
0.4
0.2
0.0
Performance
0.2
0.4
0.6
0.8
θ
ROC-AUC
Accuracy


Theta = 0.5
H
G
True label
0
0.07
0
0.93
H
G
Predicted label
0.8
0.6
0.4
0.2
0.0


Theta = 0.59
H
G
True label
0
0.071
0
0.93
H
G
Predicted label
0.8
0.6
0.4
0.2
0.0


Theta = 0.68
H
G
True label
0
0.072
0
0.93
H
G
Predicted label
0.8
0.6
0.4
0.2
0.0


Theta = 0.77
True label
H
G
Predicted label
H
G
0.0032
0.061
0.00068
0.94
0.8
0.6
0.4
0.2