In [1]:
```julia
using Pkg
Pkg.activate("..")
Pkg.instantiate()
Pkg.update()
```

```
 Activating project at `~/logic-and-machine-learning`
   Updating registry at `~/.julia/registries/General.toml`
   Updating git-repo `https://github.com/aclai-lab/ManyExpertDecisionTree
s.jl`
   Updating git-repo `https://github.com/aclai-lab/SoleReasoners.jl#embed
ding`
  No Changes to `~/logic-and-machine-learning/Project.toml`
  No Changes to `~/logic-and-machine-learning/Manifest.toml`
```

In [2]:
```julia
using Random

Random.seed!(1235)
```

Out[2]:
```
TaskLocalRNG()
```

Interpretable land cover classification with modal decision trees (extra)

Interpretable land cover classification with modal decision trees

To run this notebook, you first need to download the following datasets and place them in the `/datasets/paviaU` folder:

- Pavia University
- Pavia University GT

In [3]:
```julia
include("../scripts/land-cover.jl")
data_dir = "../datasets/"

X_df, y = LandCoverDataset(
    "Pavia University";
    window_size          = 3,
    ninstances_per_class = 40,
    pad_window_size      = 5,
);
```

```
Load LandCoverDataset: Pavia University...
window_size         = (3, 3)
pad_window_size     = (5, 5)
ninstances_per_class  = 40
ninstances_per_class_strategy = updownsampling
flattened           = false
apply_filter        = false
seed                = 1

Image size: (610, 340, 103)
class_counts_d = [("Asphalt", 1 => 6631), ("Meadows", 2 => 18649), ("Grave
l", 3 => 2099), ("Trees", 4 => 3064), ("Painted metal sheets", 5 => 1345),
("Bare Soil", 6 => 5029), ("Bitumen", 7 => 1330), ("Self-Blocking Bricks",
8 => 3682), ("Shadows", 9 => 947)]
no_class_counts = 164624
n_classes = 9
ninstances = 40 * 9 = 360
effective_class_counts_d = [("Asphalt", 1 => 40), ("Meadows", 2 => 40),
("Gravel", 3 => 40), ("Trees", 4 => 40), ("Painted metal sheets", 5 => 4
0), ("Bare Soil", 6 => 40), ("Bitumen", 7 => 40), ("Self-Blocking Bricks",
8 => 40), ("Shadows", 9 => 40)]
countmap(labels) = Dict(5 => 40, 4 => 40, 6 => 40, 7 => 40, 2 => 40, 9 =>
40, 8 => 40, 3 => 40, 1 => 40)
```

In [4]:
```julia
countmap(y)
```

Out[4]:
```
Dict{String, Int64} with 9 entries:
  "Self-Blocking Bricks" => 40
  "Bitumen"              => 40
  "Gravel"               => 40
  "Bare Soil"            => 40
  "Painted metal sheets" => 40
  "Shadows"              => 40
  "Trees"                => 40
  "Asphalt"              => 40
  "Meadows"              => 40
```

In [5]:
```julia
length.(X_df)
```

Out[5]: 360×103 DataFrame *3 columns and 335 rows omitted*

Row	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64
1	9	9	9	9	9	9	9	9	9	9	9
2	9	9	9	9	9	9	9	9	9	9	9
3	9	9	9	9	9	9	9	9	9	9	9
4	9	9	9	9	9	9	9	9	9	9	9
5	9	9	9	9	9	9	9	9	9	9	9
6	9	9	9	9	9	9	9	9	9	9	9
7	9	9	9	9	9	9	9	9	9	9	9
8	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9
10	9	9	9	9	9	9	9	9	9	9	9
11	9	9	9	9	9	9	9	9	9	9	9
12	9	9	9	9	9	9	9	9	9	9	9
13	9	9	9	9	9	9	9	9	9	9	9
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
349	9	9	9	9	9	9	9	9	9	9	9
350	9	9	9	9	9	9	9	9	9	9	9
351	9	9	9	9	9	9	9	9	9	9	9
352	9	9	9	9	9	9	9	9	9	9	9
353	9	9	9	9	9	9	9	9	9	9	9
354	9	9	9	9	9	9	9	9	9	9	9
355	9	9	9	9	9	9	9	9	9	9	9
356	9	9	9	9	9	9	9	9	9	9	9
357	9	9	9	9	9	9	9	9	9	9	9
358	9	9	9	9	9	9	9	9	9	9	9
359	9	9	9	9	9	9	9	9	9	9	9
360	9	9	9	9	9	9	9	9	9	9	9

In [6]:
```
X_df = broadcast(values->Matrix{Float64}(values), X_df)
```

Out[6]: 360×103 DataFrame *3 columns and 335 rows omitted*

Row	V1	V2	V3	V4	V5	V6	V7	V8	V9
	Array…	Array…	Array…	Array…	Array…	Array…	Array…	Array…	Array
1	[981.0 534.0 290.0; 686.0 712.0 705.0; 611.0 669.0 762.0]	[750.0 559.0 481.0; 693.0 316.0 732.0; 579.0 578.0 539.0]	[453.0 400.0 520.0; 566.0 165.0 427.0; 480.0 548.0 523.0]	[356.0 408.0 500.0; 571.0 311.0 313.0; 286.0 514.0 540.0]	[399.0 420.0 517.0; 537.0 425.0 385.0; 341.0 412.0 431.0]	[460.0 376.0 465.0; 415.0 472.0 416.0; 429.0 305.0 374.0]	[432.0 401.0 352.0; 396.0 404.0 355.0; 416.0 294.0 419.0]	[377.0 460.0 376.0; 407.0 345.0 364.0; 424.0 403.0 404.0]	[390. 424.0 389.0; 415.0 383.0 402.0; 516.0 461.0 343.0]
2	[890.0 725.0 982.0; 914.0 1186.0 1198.0; 749.0 1037.0 1034.0]	[901.0 701.0 977.0; 1004.0 1148.0 1234.0; 902.0 848.0 1350.0]	[907.0 925.0 1063.0; 1097.0 1104.0 1129.0; 979.0 935.0 1280.0]	[1071.0 1102.0 1040.0; 1128.0 1120.0 1085.0; 972.0 981.0 1067.0]	[1152.0 1224.0 1093.0; 1052.0 1155.0 1169.0; 1023.0 1112.0 924.0]	[1162.0 1308.0 1137.0; 1049.0 1149.0 1279.0; 1073.0 1269.0 1058.0]	[1159.0 1263.0 1103.0; 1177.0 1112.0 1285.0; 1121.0 1233.0 1179.0]	[1049.0 1188.0 1047.0; 1164.0 1073.0 1235.0; 1015.0 1167.0 1146.0]	[1030. 1092. 1094. 1119. 1089. 1203. 936.0 1069. 1129.
3	[1199.0 1103.0 1083.0; 1030.0 162.0 881.0; 701.0 927.0 842.0]	[724.0 1122.0 614.0; 976.0 269.0 1042.0; 538.0 1236.0 938.0]	[708.0 1053.0 599.0; 817.0 727.0 836.0; 751.0 1234.0 963.0]	[914.0 1051.0 856.0; 892.0 1004.0 929.0; 841.0 1036.0 1018.0]	[985.0 1029.0 978.0; 915.0 955.0 1020.0; 902.0 875.0 984.0]	[935.0 965.0 979.0; 826.0 906.0 947.0; 973.0 819.0 914.0]	[955.0 848.0 962.0; 842.0 879.0 896.0; 961.0 806.0 908.0]	[998.0 895.0 893.0; 915.0 932.0 926.0; 926.0 780.0 833.0]	[1020. 943.0 857.0; 882.0 975.0 947.0; 888.0 777.0 753.0]
4	[1031.0 1114.0 679.0; 841.0 901.0 1020.0; 1170.0 715.0 660.0]	[633.0 999.0 572.0; 707.0 832.0 805.0; 905.0 735.0 595.0]	[543.0 867.0 590.0; 746.0 666.0 725.0; 920.0 861.0 759.0]	[713.0 872.0 567.0; 716.0 645.0 758.0; 831.0 903.0 823.0]	[727.0 916.0 589.0; 612.0 676.0 791.0; 726.0 852.0 719.0]	[671.0 856.0 625.0; 592.0 624.0 698.0; 663.0 806.0 652.0]	[665.0 768.0 653.0; 649.0 698.0 665.0; 657.0 728.0 693.0]	[655.0 692.0 704.0; 658.0 702.0 738.0; 670.0 715.0 650.0]	[600. 668.0 742.0; 617.0 691.0 725.0; 625.0 705.0 604.0]
5	[356.0 694.0 642.0; 680.0 941.0 775.0; 725.0 627.0 765.0]	[252.0 749.0 670.0; 782.0 721.0 724.0; 552.0 477.0 623.0]	[354.0 811.0 597.0; 655.0 341.0 786.0; 398.0 527.0 697.0]	[595.0 725.0 646.0; 610.0 408.0 750.0; 508.0 481.0 791.0]	[733.0 659.0 758.0; 742.0 658.0 588.0; 631.0 514.0 641.0]	[664.0 675.0 749.0; 768.0 696.0 578.0; 653.0 705.0 494.0]	[580.0 648.0 696.0; 696.0 647.0 619.0; 667.0 722.0 509.0]	[595.0 629.0 614.0; 706.0 614.0 595.0; 707.0 630.0 577.0]	[674. 615.0 595.0; 708.0 615.0 548.0; 738.0 622.0 570.0]
6	[763.0 798.0 370.0;	[705.0 648.0 396.0;	[569.0 504.0 92.0;	[441.0 311.0 97.0;	[397.0 258.0 387.0;	[323.0 332.0 464.0;	[322.0 376.0 429.0;	[358.0 334.0 364.0;	[357. 327.0 345.0;

Row	V1	V2	V3	V4	V5	V6	V7	V8	V9
	Array...	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array**
	700.0 578.0 666.0; 541.0 1022.0 400.0]	370.0 419.0 598.0; 600.0 803.0 411.0]	185.0 470.0 473.0; 612.0 426.0 586.0]	21.0 387.0 456.0; 392.0 261.0 517.0]	0.0 323.0 421.0; 198.0 238.0 408.0]	184.0 360.0 428.0; 254.0 264.0 358.0]	321.0 403.0 333.0; 403.0 341.0 367.0]	374.0 404.0 265.0; 360.0 348.0 342.0]	380.0 387.0 263.0; 283.0 331.0 335.0]
7	[798.0 1031.0 685.0; 981.0 534.0 290.0; 686.0 712.0 705.0]	[734.0 767.0 524.0; 750.0 559.0 481.0; 693.0 316.0 732.0]	[583.0 702.0 294.0; 453.0 400.0 520.0; 566.0 165.0 427.0]	[568.0 542.0 279.0; 356.0 408.0 500.0; 571.0 311.0 313.0]	[497.0 517.0 323.0; 399.0 420.0 517.0; 537.0 425.0 385.0]	[422.0 523.0 378.0; 460.0 376.0 465.0; 415.0 472.0 416.0]	[433.0 486.0 371.0; 432.0 401.0 352.0; 396.0 404.0 355.0]	[444.0 479.0 384.0; 377.0 460.0 376.0; 407.0 345.0 364.0]	[430. 432.0 353.0; 390.0 424.0 389.0; 415.0 383.0 402.0]
8	[791.0 755.0 1091.0; 1129.0 1009.0 396.0; 527.0 1117.0 1107.0]	[732.0 437.0 857.0; 870.0 742.0 461.0; 622.0 1133.0 770.0]	[586.0 552.0 613.0; 747.0 487.0 757.0; 601.0 1105.0 561.0]	[674.0 632.0 600.0; 837.0 557.0 799.0; 601.0 1069.0 630.0]	[717.0 601.0 753.0; 819.0 775.0 690.0; 786.0 901.0 661.0]	[691.0 573.0 806.0; 745.0 822.0 752.0; 821.0 759.0 631.0]	[630.0 627.0 777.0; 707.0 823.0 844.0; 819.0 709.0 769.0]	[660.0 736.0 728.0; 680.0 855.0 820.0; 785.0 721.0 860.0]	[665. 691.0 732.0; 662.0 805.0 749.0; 760.0 721.0 795.0]
9	[2174.0 1854.0 1971.0; 2664.0 3001.0 3041.0; 3430.0 3644.0 3362.0]	[2208.0 2096.0 2158.0; 3049.0 3262.0 3213.0; 3655.0 3736.0 3316.0]	[2076.0 2270.0 2207.0; 3243.0 3674.0 3269.0; 3950.0 3988.0 3687.0]	[2153.0 2527.0 2363.0; 3403.0 3961.0 3572.0; 4347.0 4283.0 4254.0]	[2210.0 2772.0 2509.0; 3555.0 4199.0 3971.0; 4546.0 4527.0 4765.0]	[2357.0 2904.0 2638.0; 3728.0 4493.0 4194.0; 4651.0 4891.0 5002.0]	[2493.0 2866.0 2762.0; 3837.0 4643.0 4326.0; 4913.0 5103.0 5073.0]	[2479.0 2859.0 2776.0; 4131.0 4684.0 4516.0; 5090.0 5206.0 5170.0]	[2559. 2994. 2747. 4373. 4865. 4718. 5254. 5422. 5231.
10	[527.0 727.0 592.0; 818.0 694.0 563.0; 782.0 819.0 833.0]	[526.0 548.0 403.0; 700.0 575.0 501.0; 544.0 770.0 751.0]	[471.0 568.0 398.0; 662.0 514.0 425.0; 426.0 627.0 583.0]	[484.0 566.0 519.0; 604.0 533.0 582.0; 539.0 660.0 606.0]	[532.0 490.0 494.0; 529.0 619.0 573.0; 648.0 691.0 540.0]	[578.0 515.0 482.0; 529.0 649.0 501.0; 639.0 642.0 449.0]	[567.0 565.0 497.0; 571.0 584.0 530.0; 580.0 626.0 433.0]	[505.0 519.0 542.0; 542.0 528.0 505.0; 507.0 603.0 441.0]	[487. 432.0 555.0; 506.0 558.0 501.0; 441.0 571.0 427.0]
11	[380.0 1141.0 439.0; 874.0 897.0 929.0; 1114.0	[410.0 776.0 402.0; 632.0 887.0 722.0; 999.0	[562.0 729.0 554.0; 677.0 915.0 719.0; 867.0	[614.0 661.0 716.0; 722.0 927.0 643.0; 872.0	[608.0 609.0 849.0; 686.0 823.0 596.0; 916.0	[540.0 591.0 838.0; 585.0 747.0 650.0; 856.0	[535.0 577.0 743.0; 483.0 734.0 724.0; 768.0	[596.0 651.0 708.0; 513.0 719.0 755.0; 692.0	[588. 700.0 739.0; 547.0 737.0 692.0; 668.0

Row	V1	V2	V3	V4	V5	V6	V7	V8	V9
	Array…	**Array…**	**Array…**	**Array…**	**Array…**	**Array…**	**Array…**	**Array…**	**Array**
	679.0 1006.0]	572.0 856.0]	590.0 632.0]	567.0 671.0]	589.0 602.0]	625.0 593.0]	653.0 703.0]	704.0 801.0]	742.0 764.0]
12	[788.0 665.0 660.0; 716.0 271.0 320.0; 517.0 815.0 430.0]	[512.0 527.0 559.0; 634.0 45.0 122.0; 360.0 506.0 351.0]	[307.0 401.0 245.0; 532.0 163.0 190.0; 108.0 328.0 247.0]	[309.0 460.0 158.0; 494.0 371.0 315.0; 99.0 333.0 292.0]	[355.0 452.0 437.0; 429.0 443.0 478.0; 222.0 362.0 421.0]	[384.0 378.0 491.0; 323.0 343.0 453.0; 375.0 372.0 534.0]	[467.0 408.0 438.0; 299.0 280.0 405.0; 394.0 420.0 534.0]	[461.0 382.0 399.0; 324.0 284.0 317.0; 288.0 389.0 433.0]	[384.0 347.0 396.0; 332.0 260.0 244.0; 288.0 330.0 384.0]
13	[828.0 989.0 761.0; 761.0 506.0 669.0; 489.0 873.0 549.0]	[861.0 1064.0 678.0; 746.0 590.0 698.0; 160.0 606.0 550.0]	[851.0 917.0 551.0; 649.0 557.0 688.0; 213.0 458.0 461.0]	[641.0 735.0 665.0; 577.0 617.0 708.0; 354.0 667.0 541.0]	[584.0 690.0 632.0; 622.0 639.0 686.0; 493.0 616.0 668.0]	[588.0 652.0 563.0; 509.0 609.0 597.0; 545.0 406.0 723.0]	[598.0 552.0 628.0; 400.0 601.0 586.0; 546.0 398.0 651.0]	[600.0 569.0 615.0; 403.0 603.0 552.0; 541.0 482.0 563.0]	[545.0 624.0 557.0; 449.0 613.0 518.0; 506.0 494.0 556.0]
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
349	[108.0 646.0 540.0; 334.0 269.0 561.0; 367.0 929.0 773.0]	[470.0 206.0 134.0; 427.0 58.0 462.0; 462.0 612.0 386.0]	[696.0 106.0 173.0; 426.0 207.0 251.0; 336.0 265.0 157.0]	[560.0 172.0 354.0; 311.0 365.0 304.0; 359.0 277.0 117.0]	[334.0 248.0 309.0; 158.0 390.0 388.0; 219.0 253.0 81.0]	[254.0 186.0 257.0; 122.0 251.0 375.0; 160.0 345.0 136.0]	[286.0 192.0 233.0; 153.0 209.0 308.0; 192.0 383.0 199.0]	[350.0 298.0 185.0; 165.0 204.0 218.0; 180.0 305.0 182.0]	[388.0 305.0 242.0; 181.0 170.0 163.0; 142.0 249.0 189.0]
350	[1217.0 610.0 876.0; 607.0 713.0 351.0; 332.0 724.0 796.0]	[890.0 454.0 479.0; 359.0 494.0 516.0; 365.0 634.0 585.0]	[446.0 443.0 292.0; 266.0 295.0 497.0; 466.0 437.0 370.0]	[251.0 374.0 360.0; 337.0 193.0 393.0; 438.0 408.0 411.0]	[251.0 233.0 355.0; 380.0 267.0 242.0; 357.0 366.0 356.0]	[254.0 293.0 351.0; 425.0 361.0 149.0; 336.0 322.0 323.0]	[290.0 292.0 394.0; 386.0 375.0 164.0; 367.0 306.0 322.0]	[332.0 225.0 305.0; 352.0 281.0 178.0; 351.0 306.0 246.0]	[302.0 236.0 156.0; 370.0 203.0 203.0; 352.0 218.0 143.0]
351	[461.0 467.0 821.0; 911.0 374.0 645.0; 592.0 674.0 654.0]	[397.0 231.0 485.0; 616.0 334.0 277.0; 478.0 738.0 537.0]	[535.0 127.0 365.0; 545.0 407.0 227.0; 236.0 450.0 495.0]	[449.0 177.0 436.0; 576.0 520.0 194.0; 336.0 327.0 321.0]	[359.0 304.0 439.0; 435.0 542.0 228.0; 428.0 433.0 250.0]	[223.0 379.0 441.0; 394.0 444.0 361.0; 422.0 461.0 276.0]	[238.0 424.0 386.0; 368.0 424.0 414.0; 387.0 377.0 258.0]	[331.0 381.0 280.0; 340.0 365.0 313.0; 327.0 304.0 204.0]	[338.0 288.0 232.0; 305.0 283.0 230.0; 264.0 221.0 184.0]

Row	V1	V2	V3	V4	V5	V6	V7	V8	V9
	Array...	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array**
352	[1294.0 1306.0 443.0; 711.0 918.0 421.0; 961.0 0.0 367.0]	[1560.0 1004.0 546.0; 654.0 815.0 178.0; 808.0 0.0 353.0]	[1445.0 837.0 596.0; 789.0 571.0 180.0; 536.0 120.0 204.0]	[1277.0 942.0 552.0; 798.0 561.0 168.0; 415.0 302.0 196.0]	[1240.0 1029.0 445.0; 683.0 535.0 287.0; 387.0 233.0 260.0]	[1224.0 1008.0 418.0; 595.0 465.0 319.0; 327.0 200.0 245.0]	[1217.0 1013.0 522.0; 516.0 413.0 265.0; 256.0 185.0 263.0]	[1240.0 990.0 501.0; 525.0 354.0 194.0; 237.0 242.0 233.0]	[1288. 892.0 418.0; 546.0 361.0 177.0; 261.0 309.0 153.0]
353	[376.0 569.0 730.0; 747.0 633.0 850.0; 617.0 797.0 698.0]	[334.0 476.0 556.0; 511.0 302.0 683.0; 425.0 529.0 553.0]	[421.0 517.0 182.0; 218.0 325.0 574.0; 272.0 495.0 454.0]	[246.0 506.0 206.0; 282.0 394.0 500.0; 297.0 512.0 418.0]	[173.0 453.0 409.0; 328.0 365.0 454.0; 346.0 390.0 398.0]	[207.0 414.0 460.0; 327.0 335.0 409.0; 321.0 244.0 368.0]	[273.0 335.0 293.0; 278.0 370.0 332.0; 318.0 263.0 340.0]	[255.0 221.0 236.0; 199.0 330.0 258.0; 296.0 293.0 260.0]	[261.0 250.0 235.0; 201.0 240.0 222.0; 232.0 310.0 192.0]
354	[2418.0 1489.0 990.0; 1743.0 731.0 906.0; 1639.0 1025.0 969.0]	[2284.0 1505.0 1027.0; 1612.0 712.0 762.0; 1697.0 1099.0 950.0]	[2157.0 1437.0 915.0; 1542.0 759.0 570.0; 1447.0 1097.0 720.0]	[2168.0 1402.0 852.0; 1587.0 830.0 477.0; 1490.0 1017.0 633.0]	[2252.0 1406.0 776.0; 1547.0 856.0 499.0; 1623.0 850.0 605.0]	[2357.0 1476.0 704.0; 1407.0 836.0 600.0; 1668.0 819.0 577.0]	[2370.0 1421.0 761.0; 1363.0 825.0 614.0; 1669.0 856.0 531.0]	[2332.0 1243.0 788.0; 1426.0 728.0 518.0; 1646.0 857.0 414.0]	[2374. 1256.0 753.0; 1564.0 611.0 323.0; 1633.0 890.0 385.0]
355	[517.0 566.0 155.0; 645.0 375.0 534.0; 666.0 896.0 745.0]	[555.0 158.0 336.0; 440.0 235.0 587.0; 307.0 553.0 472.0]	[585.0 31.0 177.0; 386.0 269.0 468.0; 258.0 332.0 384.0]	[573.0 194.0 136.0; 359.0 408.0 308.0; 166.0 120.0 283.0]	[390.0 371.0 223.0; 210.0 392.0 317.0; 153.0 162.0 256.0]	[384.0 380.0 260.0; 141.0 364.0 263.0; 242.0 261.0 236.0]	[392.0 338.0 242.0; 244.0 252.0 250.0; 274.0 263.0 222.0]	[446.0 287.0 204.0; 253.0 215.0 313.0; 304.0 262.0 235.0]	[441.0 242.0 223.0; 230.0 209.0 296.0; 276.0 228.0 223.0]
356	[329.0 353.0 587.0; 613.0 575.0 418.0; 520.0 755.0 846.0]	[124.0 331.0 373.0; 854.0 227.0 241.0; 429.0 542.0 592.0]	[116.0 272.0 213.0; 629.0 127.0 285.0; 527.0 286.0 497.0]	[241.0 245.0 238.0; 300.0 197.0 360.0; 444.0 264.0 292.0]	[360.0 175.0 247.0; 202.0 246.0 245.0; 226.0 366.0 187.0]	[346.0 178.0 153.0; 187.0 216.0 230.0; 181.0 363.0 256.0]	[280.0 237.0 175.0; 116.0 241.0 207.0; 284.0 331.0 296.0]	[199.0 218.0 211.0; 98.0 189.0 209.0; 292.0 268.0 263.0]	[147.0 157.0 205.0; 109.0 138.0 244.0; 254.0 308.0 242.0]
357	[523.0 624.0 623.0; 484.0	[446.0 139.0 495.0; 194.0	[330.0 0.0 395.0; 0.0	[248.0 96.0 264.0; 25.0	[313.0 178.0 274.0; 177.0	[340.0 137.0 418.0; 355.0	[348.0 137.0 361.0; 407.0	[328.0 258.0 267.0; 320.0	[239.0 280.0 222.0; 295.0

Row	V1	V2	V3	V4	V5	V6	V7	V8	V9
	Array...	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array...**	**Array**
	505.0 514.0; 599.0 996.0 999.0]	459.0 615.0; 529.0 718.0 789.0]	267.0 452.0; 394.0 538.0 571.0]	243.0 332.0; 237.0 484.0 365.0]	247.0 253.0; 261.0 415.0 265.0]	248.0 150.0; 285.0 317.0 311.0]	151.0 179.0; 243.0 307.0 314.0]	101.0 210.0; 241.0 329.0 348.0]	231.0 205.0; 219.0 334.0 316.0]
358	[747.0 633.0 850.0; 617.0 797.0 698.0; 682.0 570.0 946.0]	[511.0 302.0 683.0; 425.0 529.0 553.0; 680.0 478.0 567.0]	[218.0 325.0 574.0; 272.0 495.0 454.0; 591.0 331.0 473.0]	[282.0 394.0 500.0; 297.0 512.0 418.0; 541.0 381.0 422.0]	[328.0 365.0 454.0; 346.0 390.0 398.0; 425.0 484.0 349.0]	[327.0 335.0 409.0; 321.0 244.0 368.0; 357.0 393.0 289.0]	[278.0 370.0 332.0; 318.0 263.0 340.0; 344.0 318.0 253.0]	[199.0 330.0 258.0; 296.0 293.0 260.0; 291.0 335.0 171.0]	[201.0 240.0 222.0; 232.0 310.0 192.0; 270.0 298.0 167.0]
359	[1590.0 1048.0 879.0; 1547.0 913.0 903.0; 2493.0 1402.0 716.0]	[1666.0 934.0 736.0; 1640.0 966.0 602.0; 2396.0 1315.0 775.0]	[1699.0 875.0 642.0; 1633.0 831.0 442.0; 2449.0 1343.0 805.0]	[1649.0 572.0 626.0; 1610.0 651.0 452.0; 2756.0 1568.0 663.0]	[1640.0 358.0 608.0; 1771.0 524.0 431.0; 3039.0 1742.0 527.0]	[1737.0 466.0 533.0; 1978.0 519.0 428.0; 3173.0 1653.0 518.0]	[1726.0 617.0 467.0; 1976.0 513.0 420.0; 3217.0 1595.0 479.0]	[1658.0 638.0 413.0; 1938.0 518.0 365.0; 3327.0 1578.0 514.0]	[1637. 579.0 426.0; 1941.(507.0 379.0; 3451.(1505.(553.0]
360	[929.0 640.0 545.0; 842.0 764.0 692.0; 565.0 538.0 593.0]	[596.0 553.0 332.0; 609.0 522.0 590.0; 178.0 526.0 529.0]	[440.0 395.0 312.0; 394.0 444.0 491.0; 0.0 394.0 442.0]	[367.0 416.0 353.0; 408.0 330.0 328.0; 148.0 251.0 315.0]	[309.0 424.0 374.0; 384.0 161.0 316.0; 293.0 231.0 321.0]	[262.0 342.0 355.0; 314.0 199.0 400.0; 339.0 238.0 314.0]	[295.0 367.0 388.0; 241.0 282.0 331.0; 255.0 205.0 291.0]	[313.0 381.0 360.0; 198.0 298.0 192.0; 205.0 200.0 315.0]	[288.0 305.0 263.0; 167.0 297.0 148.0; 225.0 166.0 272.0]

In [7]:
```julia
using DataFrames

# Let's unwind the spatial axes
X_df_static = Matrix(X_df)
cols = []
for i_var in 1:size(X_df_static, 2)
    var_unroll = cat(X_df_static[:,i_var]...; dims = 3)
    append!(cols, eachrow(reshape(var_unroll, (9, nrow(X_df)))))
end
X_df_static = DataFrame(
    cols,
    ["$n[$i][$j]" for n in names(X_df) for i in 1:3 for j in 1:3]
)
```

```
Out[7]:
```

360×927 DataFrame *827 columns and 335 rows omitted*

Row	V1[1][1]	V1[1][2]	V1[1][3]	V1[2][1]	V1[2][2]	V1[2][3]	V1[3][1]	V1[3][2]	V1[3][3]
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Floa
1	981.0	686.0	611.0	534.0	712.0	669.0	290.0	705.0	76
2	890.0	914.0	749.0	725.0	1186.0	1037.0	982.0	1198.0	103
3	1199.0	1030.0	701.0	1103.0	162.0	927.0	1083.0	881.0	84
4	1031.0	841.0	1170.0	1114.0	901.0	715.0	679.0	1020.0	66
5	356.0	680.0	725.0	694.0	941.0	627.0	642.0	775.0	76
6	763.0	700.0	541.0	798.0	578.0	1022.0	370.0	666.0	40
7	798.0	981.0	686.0	1031.0	534.0	712.0	685.0	290.0	70
8	791.0	1129.0	527.0	755.0	1009.0	1117.0	1091.0	396.0	110
9	2174.0	2664.0	3430.0	1854.0	3001.0	3644.0	1971.0	3041.0	336
10	527.0	818.0	782.0	727.0	694.0	819.0	592.0	563.0	83
11	380.0	874.0	1114.0	1141.0	897.0	679.0	439.0	929.0	100
12	788.0	716.0	517.0	665.0	271.0	815.0	660.0	320.0	43
13	828.0	761.0	489.0	989.0	506.0	873.0	761.0	669.0	54
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
349	108.0	334.0	367.0	646.0	269.0	929.0	540.0	561.0	77
350	1217.0	607.0	332.0	610.0	713.0	724.0	876.0	351.0	79
351	461.0	911.0	592.0	467.0	374.0	674.0	821.0	645.0	65
352	1294.0	711.0	961.0	1306.0	918.0	0.0	443.0	421.0	36
353	376.0	747.0	617.0	569.0	633.0	797.0	730.0	850.0	69
354	2418.0	1743.0	1639.0	1489.0	731.0	1025.0	990.0	906.0	96
355	517.0	645.0	666.0	566.0	375.0	896.0	155.0	534.0	74
356	329.0	613.0	520.0	353.0	575.0	755.0	587.0	418.0	84
357	523.0	484.0	599.0	624.0	505.0	996.0	623.0	514.0	99
358	747.0	617.0	682.0	633.0	797.0	570.0	850.0	698.0	94
359	1590.0	1547.0	2493.0	1048.0	913.0	1402.0	879.0	903.0	71
360	929.0	842.0	565.0	640.0	764.0	538.0	545.0	692.0	59

```
In [8]: using MultiData

X_multimodal = MultiModalDataset([X_df, X_df_static])
```

```
Out[8]: ● MultiDataset{DataFrame}
           └─ dimensionalities: (2, 0)
        - Modality 1 / 2
           └─ dimensionality: 2
        360×103 SubDataFrame
         Row │ V1                                   V2
        V ⋯
             │ Array…                               Array…
        A ⋯
        ─────┼──────────────────────────────────────────────────────────────────────
        ────────
           1 │ [981.0 534.0 290.0; 686.0 712.0 …  [750.0 559.0 481.0; 693.0 316.
        0 …  [ ⋯
           2 │ [890.0 725.0 982.0; 914.0 1186.0…  [901.0 701.0 977.0; 1004.0 114
        8.…  [
           3 │ [1199.0 1103.0 1083.0; 1030.0 16…  [724.0 1122.0 614.0; 976.0 26
        9.0…  [
           4 │ [1031.0 1114.0 679.0; 841.0 901.…  [633.0 999.0 572.0; 707.0 832.
        0 …  [
           5 │ [356.0 694.0 642.0; 680.0 941.0 …  [252.0 749.0 670.0; 782.0 721.
        0 …  [ ⋯
           6 │ [763.0 798.0 370.0; 700.0 578.0 …  [705.0 648.0 396.0; 370.0 419.
        0 …  [
           7 │ [798.0 1031.0 685.0; 981.0 534.0…  [734.0 767.0 524.0; 750.0 559.
        0 …  [
           8 │ [791.0 755.0 1091.0; 1129.0 1009…  [732.0 437.0 857.0; 870.0 742.
        0 …  [
           9 │ [2174.0 1854.0 1971.0; 2664.0 30…  [2208.0 2096.0 2158.0; 3049.0
        32…  [ ⋯
          10 │ [527.0 727.0 592.0; 818.0 694.0 …  [526.0 548.0 403.0; 700.0 575.
        0 …  [
          11 │ [380.0 1141.0 439.0; 874.0 897.0…  [410.0 776.0 402.0; 632.0 887.
        0 …  [
          ⋮  │                  ⋮                                  ⋮
        ⋱
         351 │ [461.0 467.0 821.0; 911.0 374.0 …  [397.0 231.0 485.0; 616.0 334.
        0 …  [
         352 │ [1294.0 1306.0 443.0; 711.0 918.…  [1560.0 1004.0 546.0; 654.0 81
        5.…  [ ⋯
         353 │ [376.0 569.0 730.0; 747.0 633.0 …  [334.0 476.0 556.0; 511.0 302.
        0 …  [
         354 │ [2418.0 1489.0 990.0; 1743.0 731…  [2284.0 1505.0 1027.0; 1612.0
        71…  [
         355 │ [517.0 566.0 155.0; 645.0 375.0 …  [555.0 158.0 336.0; 440.0 235.
        0 …  [
         356 │ [329.0 353.0 587.0; 613.0 575.0 …  [124.0 331.0 373.0; 854.0 227.
        0 …  [ ⋯
         357 │ [523.0 624.0 623.0; 484.0 505.0 …  [446.0 139.0 495.0; 194.0 459.
        0 …  [
         358 │ [747.0 633.0 850.0; 617.0 797.0 …  [511.0 302.0 683.0; 425.0 529.
        0 …  [
         359 │ [1590.0 1048.0 879.0; 1547.0 913…  [1666.0 934.0 736.0; 1640.0 96
        6.…  [
         360 │ [929.0 640.0 545.0; 842.0 764.0 …  [596.0 553.0 332.0; 609.0 522.
        0 …  [ ⋯
                                                          101 columns and 339 rows
        omitted
        - Modality 2 / 2
           └─ dimensionality: 0
        360×927 SubDataFrame
```

```
 Row │ V1[1][1]  V1[1][2]  V1[1][3]  V1[2][1]  V1[2][2]  V1[2][3]  V1[3]
[1]  V …
     │ Float64   Float64   Float64   Float64   Float64   Float64   Float
64   F …
─────┼──────────────────────────────────────────────────────────────────
────────
   1 │    981.0     686.0     611.0     534.0     712.0     669.0     29
0.0    …
   2 │    890.0     914.0     749.0     725.0    1186.0    1037.0     98
2.0
   3 │   1199.0    1030.0     701.0    1103.0     162.0     927.0    108
3.0
   4 │   1031.0     841.0    1170.0    1114.0     901.0     715.0     67
9.0
   5 │    356.0     680.0     725.0     694.0     941.0     627.0     64
2.0    …
   6 │    763.0     700.0     541.0     798.0     578.0    1022.0     37
0.0
   7 │    798.0     981.0     686.0    1031.0     534.0     712.0     68
5.0
   8 │    791.0    1129.0     527.0     755.0    1009.0    1117.0    109
1.0
   9 │   2174.0    2664.0    3430.0    1854.0    3001.0    3644.0    197
1.0    …
  10 │    527.0     818.0     782.0     727.0     694.0     819.0     59
2.0
  11 │    380.0     874.0    1114.0    1141.0     897.0     679.0     43
9.0
  ⋮  │     ⋮         ⋮         ⋮         ⋮         ⋮         ⋮         ⋮
⋱
 351 │    461.0     911.0     592.0     467.0     374.0     674.0     82
1.0
 352 │   1294.0     711.0     961.0    1306.0     918.0       0.0     44
3.0    …
 353 │    376.0     747.0     617.0     569.0     633.0     797.0     73
0.0
 354 │   2418.0    1743.0    1639.0    1489.0     731.0    1025.0     99
0.0
 355 │    517.0     645.0     666.0     566.0     375.0     896.0     15
5.0
 356 │    329.0     613.0     520.0     353.0     575.0     755.0     58
7.0    …
 357 │    523.0     484.0     599.0     624.0     505.0     996.0     62
3.0
 358 │    747.0     617.0     682.0     633.0     797.0     570.0     85
0.0
 359 │   1590.0    1547.0    2493.0    1048.0     913.0    1402.0     87
9.0
 360 │    929.0     842.0     565.0     640.0     764.0     538.0     54
5.0    …
                                          920 columns and 339 rows
omitted
```

In [9]:

```julia
using ModalDecisionTrees

model = ModalDecisionTree(; relations = :RCC8)
```

Out[9]:

```
ModalDecisionTree(nothing, 4, 0.002, Inf, nothing, :RCC8, nothing, nothi
ng, Float64, nothing, SoleData.var"#downsize#541"(), true, false, false,
TaskLocalRNG(), nothing, nothing, identity, false, nothing, :split)
```

In [10]:
```julia
using MLJ

modalmach = machine(model, X_multimodal, y; scitype_check_level=0)
```

Out[10]:
```
untrained Machine; caches model-specific representations of data
  model: ModalDecisionTree(max_depth = nothing, …)
  args:
    1:  Source @934 ⏎ Table{Union{AbstractVector{AbstractVector{Abstract
Vector{Count}}}, AbstractVector{Table{Union{AbstractVector{AbstractMatri
x{Continuous}}, AbstractVector{Continuous}}}}}}
    2:  Source @366 ⏎ AbstractVector{Textual}
```

In [11]:
```julia
fit!(modalmach)
```

```
[ Info: Precomputing logiset...
[ Info: Training machine(ModalDecisionTree(max_depth = nothing, …), …).
```

Out[11]:
```
trained Machine; caches model-specific representations of data
  model: ModalDecisionTree(max_depth = nothing, …)
  args:
    1:  Source @934 ⏎ Table{Union{AbstractVector{AbstractVector{Abstract
Vector{Count}}}, AbstractVector{Table{Union{AbstractVector{AbstractMatri
x{Continuous}}, AbstractVector{Continuous}}}}}}
    2:  Source @366 ⏎ AbstractVector{Textual}
```

In [12]:
```julia
fitted_params(modalmach).tree
```

Out[12]:

```
Decision Tree{String}(
    worldtypes:    DataType[Interval2D{Int64}, OneWorld]
    initconditions: ModalDecisionTrees.InitialCondition[ModalDecisionTre
es.StartWithoutWorld(), ModalDecisionTrees.StartWithoutWorld()]
    ###########################################################
    sub-tree leaves: 19
    sub-tree nodes: 37
    sub-tree height: 7
    sub-tree modal height:  2
    ###########################################################
    tree:
{2} RestrictedDecision(⟨=⟩V897 ≥ 1652.0)                           Meadows : 4
0/360 (conf = 0.1111)
✔ {2} RestrictedDecision(⟨=⟩V45 ≥ 1046.0)                          Meadows : 4
0/235 (conf = 0.1702)
│✔ {1} RestrictedDecision(⟨G⟩min[V2] ≥ 2609.0)                     Painted met
al sheets : 40/119 (conf = 0.3361)
││✔ Painted metal sheets : 40/40 (conf = 1.0000)
││✘ {2} RestrictedDecision(⟨=⟩V115 ≥ 1443.0)                       Self-Blocki
ng Bricks : 40/79 (conf = 0.5063)
││ ✔ {1} RestrictedDecision(⟨G⟩min[V25] ≥ 2063.0)                 Self-Blocki
ng Bricks : 30/34 (conf = 0.8824)
││ │✔ Bare Soil : 3/4 (conf = 0.7500)
││ │✘ Self-Blocking Bricks : 30/30 (conf = 1.0000)
││ ✘ {2} RestrictedDecision(⟨=⟩V28 ≥ 1185.0)                      Gravel : 3
4/45 (conf = 0.7556)
││  ✔ Gravel : 28/29 (conf = 0.9655)
││  ✘ {2} RestrictedDecision(⟨=⟩V116 ≥ 1330.0)                    Self-Blocki
ng Bricks : 9/16 (conf = 0.5625)
││   ✔ Self-Blocking Bricks : 7/7 (conf = 1.0000)
││   ✘ {1} RestrictedDecision(⟨G⟩min[V13] ≥ 1386.0)               Gravel :
6/9 (conf = 0.6667)
││    ✔ Gravel : 5/5 (conf = 1.0000)
││    ✘ Self-Blocking Bricks : 2/4 (conf = 0.5000)
│✘ {1} RestrictedDecision(⟨G⟩min[V68] < 702.0)                     Meadows : 4
0/116 (conf = 0.3448)
│ ✔ Trees : 39/40 (conf = 0.9750)
│ ✘ {2} RestrictedDecision(⟨=⟩V891 ≥ 2573.0)                       Meadows : 3
9/76 (conf = 0.5132)
│  ✔ Meadows : 18/20 (conf = 0.9000)
│  ✘ {2} RestrictedDecision(⟨=⟩V742 ≥ 2298.0)                      Bare Soil :
35/56 (conf = 0.6250)
│   ✔ {1} RestrictedDecision(⟨G⟩min[V7] ≥ 640.0)                  Bare Soil :
21/23 (conf = 0.9130)
│   │✔ Bare Soil : 19/19 (conf = 1.0000)
│   │✘ Meadows : 2/4 (conf = 0.5000)
│   ✘ {1} RestrictedDecision(⟨G⟩min[V100] < 2098.0)               Meadows : 1
9/33 (conf = 0.5758)
│    ✔ {1} RestrictedDecision(⟨=⟩min[V83] ≥ 1947.0)               Bare Soil :
14/17 (conf = 0.8235)
│    │✔ Bare Soil : 12/12 (conf = 1.0000)
│    │✘ Meadows : 3/5 (conf = 0.6000)
│    ✘ Meadows : 16/16 (conf = 1.0000)
✘ {1} RestrictedDecision(⟨G⟩min[V40] < 306.0)                      Asphalt : 4
0/125 (conf = 0.3200)
 ✔ Shadows : 40/40 (conf = 1.0000)
 ✘ {2} RestrictedDecision(⟨=⟩V125 ≥ 1179.0)                        Asphalt : 4
0/85 (conf = 0.4706)
  ✔ {2} RestrictedDecision(⟨=⟩V139 ≥ 1247.0)                       Bitumen : 4
0/46 (conf = 0.8696)
```

```
   │✔ Bitumen : 38/39 (conf = 0.9744)
   │✘ Asphalt : 5/7 (conf = 0.7143)
  ✘ {1} RestrictedDecision(⟨G⟩min[V30] < 1143.0)            Asphalt : 3
4/39 (conf = 0.8718)
   ✔ Asphalt : 34/34 (conf = 1.0000)
   ✘ Gravel : 5/5 (conf = 1.0000)

)
```

In [13]:

```
🌱 = report(modalmach).model
```

Out[13]:

```
▣ {2}((V897 ≥ 1652.0))
├✔ {2}((V45 ≥ 1046.0))
│ ├✔ {1}(⟨G⟩(min[V2] ≥ 2609.0))
│ │ ├✔ Painted metal sheets
│ │ └✘ {2}((V115 ≥ 1443.0))
│ │   ├✔ {1}(⟨G⟩(min[V25] ≥ 2063.0))
│ │   │ ├✔ Bare Soil
│ │   │ └✘ Self-Blocking Bricks
│ │   └✘ {2}((V28 ≥ 1185.0))
│ │     ├✔ Gravel
│ │     └✘ {2}((V116 ≥ 1330.0))
│ │       ├✔ Self-Blocking Bricks
│ │       └✘ {1}(⟨G⟩(min[V13] ≥ 1386.0))
│ │         ├✔ Gravel
│ │         └✘ Self-Blocking Bricks
│ └✘ {1}(⟨G⟩(min[V68] < 702.0))
│   ├✔ Trees
│   └✘ {2}((V891 ≥ 2573.0))
│     ├✔ Meadows
│     └✘ {2}((V742 ≥ 2298.0))
│       ├✔ {1}(⟨G⟩(min[V7] ≥ 640.0))
│       │ ├✔ Bare Soil
│       │ └✘ Meadows
│       └✘ {1}(⟨G⟩(min[V100] < 2098.0))
│         ├✔ {1}(⟨G⟩((min[V100] < 2098.0) ∧ (min[V83] ≥ 1947.0)))
│         │ ├✔ Bare Soil
│         │ └✘ Meadows
│         └✘ Meadows
└✘ {1}(⟨G⟩(min[V40] < 306.0))
  ├✔ Shadows
  └✘ {2}((V125 ≥ 1179.0))
    ├✔ {2}((V139 ≥ 1247.0))
    │ ├✔ Bitumen
    │ └✘ Asphalt
    └✘ {1}(⟨G⟩(min[V30] < 1143.0))
      ├✔ Asphalt
      └✘ Gravel
```

In [14]:

```
using SoleModels

🌲 = listrules(🌱)
```

Out[14]:
```
19-element Vector{ClassificationRule{String}}:
 ▣ {1}(⟨G⟩(min[V2] ≥ 2609.0)) ∧ {2}((V45 ≥ 1046.0))  ↣  Painted metal sh
eets

 ▣ {1}⟨G⟩(min[V25] ≥ 2063.0) ∧ [G](min[V2] < 2609.0) ∧ {2}((V115 ≥ 144
3.0))  ↣  Bare Soil

 ▣ {1}[G](min[V2] < 2609.0) ∧ [G](min[V25] < 2063.0) ∧ {2}(V45 ≥ 1046.0)
∧ (V115 ≥ 1443.0)  ↣  Self-Blocking Bricks

 ▣ {1}([G](min[V2] < 2609.0)) ∧ {2}(V28 ≥ 1185.0) ∧ (V115 < 1443.0)  ↣
Gravel

 ▣ {1}([G](min[V2] < 2609.0)) ∧ {2}(V116 ≥ 1330.0) ∧ (V115 < 1443.0) ∧
(V28 < 1185.0)  ↣  Self-Blocking Bricks

 ▣ {1}⟨G⟩(min[V13] ≥ 1386.0) ∧ [G](min[V2] < 2609.0) ∧ {2}(V116 < 133
0.0) ∧ (V115 < 1443.0) ∧ (V28 < 1185.0)  ↣  Gravel

 ▣ {1}[G](min[V2] < 2609.0) ∧ [G](min[V13] < 1386.0) ∧ {2}(V45 ≥ 1046.0)
∧ (V115 < 1443.0) ∧ (V28 < 1185.0) ∧ (V116 < 1330.0)  ↣  Self-Blocking B
ricks

 ▣ {1}(⟨G⟩(min[V68] < 702.0)) ∧ {2}((V45 < 1046.0))  ↣  Trees

 ▣ {1}([G](min[V68] ≥ 702.0)) ∧ {2}(V891 ≥ 2573.0) ∧ (V45 < 1046.0)  ↣
Meadows

 ▣ {1}⟨G⟩(min[V7] ≥ 640.0) ∧ [G](min[V68] ≥ 702.0) ∧ {2}(V742 ≥ 2298.0)
∧ (V45 < 1046.0) ∧ (V891 < 2573.0)  ↣  Bare Soil

 ▣ {1}[G](min[V68] ≥ 702.0) ∧ [G](min[V7] < 640.0) ∧ {2}(V897 ≥ 1652.0)
∧ (V45 < 1046.0) ∧ (V891 < 2573.0) ∧ (V742 ≥ 2298.0)  ↣  Meadows

 ▣ {1}⟨G⟩((min[V100] < 2098.0) ∧ (min[V83] ≥ 1947.0)) ∧ [G](min[V68] ≥ 7
02.0) ∧ {2}(V742 < 2298.0) ∧ (V45 < 1046.0) ∧ (V891 < 2573.0)  ↣  Bare S
oil

 ▣ {1}⟨G⟩(min[V100] < 2098.0) ∧ [G](min[V68] ≥ 702.0) ∧ [G]((min[V100] <
2098.0) → (min[V83] < 1947.0)) ∧ {2}(V897 ≥ 1652.0) ∧ (V45 < 1046.0) ∧
(V891 < 2573.0) ∧ (V742 < 2298.0)  ↣  Meadows

 ▣ {1}[G](min[V68] ≥ 702.0) ∧ [G](min[V100] ≥ 2098.0) ∧ {2}(V897 ≥ 165
2.0) ∧ (V45 < 1046.0) ∧ (V891 < 2573.0) ∧ (V742 < 2298.0)  ↣  Meadows

 ▣ {1}(⟨G⟩(min[V40] < 306.0)) ∧ {2}((V897 < 1652.0))  ↣  Shadows

 ▣ {1}([G](min[V40] ≥ 306.0)) ∧ {2}(V139 ≥ 1247.0) ∧ (V897 < 1652.0)  ↣
Bitumen

 ▣ {1}([G](min[V40] ≥ 306.0)) ∧ {2}(V125 ≥ 1179.0) ∧ (V897 < 1652.0) ∧
(V139 < 1247.0)  ↣  Asphalt

 ▣ {1}⟨G⟩(min[V30] < 1143.0) ∧ [G](min[V40] ≥ 306.0) ∧ {2}(V125 < 117
9.0) ∧ (V897 < 1652.0)  ↣  Asphalt

 ▣ {1}[G](min[V40] ≥ 306.0) ∧ [G](min[V30] ≥ 1143.0) ∧ {2}(V897 < 165
2.0) ∧ (V125 < 1179.0)  ↣  Gravel
```

In [15]:
```julia
# Every symbolic model (including ruleslist) can have has additional info
# attached
println(🌲[1])

ruleinfo = SoleModels.info(🌲[1])
println(keys(ruleinfo))
```

```
▣ {1}(⟨G⟩(min[V2] ≥ 2609.0)) ∧ {2}((V45 ≥ 1046.0))  ↣  Painted metal sheet
s

(:supporting_labels, :supporting_predictions, :shortform)
```

In [16]:
```julia
ruleinfo[:supporting_predictions] |> length
```

Out[16]:
```
360
```

In [17]:
```julia
sort(readmetrics.(🌲), by=x->x[:coverage], rev = true)
```

Out[17]:
```
19-element Vector{@NamedTuple{ninstances::Int64, ncovered::Int64, covera
ge::Float64, confidence::Float64, lift::Float64, natoms::Int64}}:
 (ninstances = 360, ncovered = 40, coverage = 0.1111111111111111, confid
ence = 1.0, lift = 9.0, natoms = 2)
 (ninstances = 360, ncovered = 40, coverage = 0.1111111111111111, confid
ence = 0.975, lift = 8.775, natoms = 2)
 (ninstances = 360, ncovered = 40, coverage = 0.1111111111111111, confid
ence = 1.0, lift = 9.0, natoms = 2)
 (ninstances = 360, ncovered = 39, coverage = 0.10833333333333334, confi
dence = 0.9743589743589743, lift = 8.76923076923077, natoms = 3)
 (ninstances = 360, ncovered = 34, coverage = 0.09444444444444444, confi
dence = 1.0, lift = 9.0, natoms = 4)
 (ninstances = 360, ncovered = 30, coverage = 0.08333333333333333, confi
dence = 1.0, lift = 9.0, natoms = 4)
 (ninstances = 360, ncovered = 29, coverage = 0.08055555555555556, confi
dence = 0.9655172413793104, lift = 8.689655172413794, natoms = 3)
 (ninstances = 360, ncovered = 20, coverage = 0.05555555555555555, confi
dence = 0.9, lift = 8.100000000000001, natoms = 3)
 (ninstances = 360, ncovered = 19, coverage = 0.05277777777777778, confi
dence = 1.0, lift = 9.0, natoms = 5)
 (ninstances = 360, ncovered = 16, coverage = 0.044444444444444446, conf
idence = 1.0, lift = 9.0, natoms = 6)
 (ninstances = 360, ncovered = 12, coverage = 0.03333333333333333, confi
dence = 1.0, lift = 9.0, natoms = 6)
 (ninstances = 360, ncovered = 7, coverage = 0.019444444444444445, confi
dence = 1.0, lift = 9.0, natoms = 4)
 (ninstances = 360, ncovered = 7, coverage = 0.019444444444444445, confi
dence = 0.7142857142857143, lift = 6.428571428571429, natoms = 4)
 (ninstances = 360, ncovered = 5, coverage = 0.013888888888888888, confi
dence = 1.0, lift = 9.0, natoms = 5)
 (ninstances = 360, ncovered = 5, coverage = 0.013888888888888888, confi
dence = 0.6, lift = 5.4, natoms = 8)
 (ninstances = 360, ncovered = 5, coverage = 0.013888888888888888, confi
dence = 1.0, lift = 9.0, natoms = 4)
 (ninstances = 360, ncovered = 4, coverage = 0.011111111111111112, confi
dence = 0.75, lift = 6.75, natoms = 3)
 (ninstances = 360, ncovered = 4, coverage = 0.011111111111111112, confi
dence = 0.5, lift = 4.5, natoms = 6)
 (ninstances = 360, ncovered = 4, coverage = 0.011111111111111112, confi
dence = 0.5, lift = 4.5, natoms = 6)
```

In [18]:
```julia
metricstable(🌲)
```

Antecedent	Consequent	ninstances	ncovered	coverage	confidence	lift	natoms
{1}(⟨G⟩min[V2] ≥ 2609.0) ∧ {2}(V45 ≥ 1046.0)	Painted metal sheets	360	40	0.111111	1.0	9.0	2
{1}⟨G⟩min[V25] ≥ 2063.0 ∧ [G]min[V2] < 2609.0 ∧ {2}(V115 ≥ 1443.0)	Bare Soil	360	4	0.0111111	0.75	6.75	3
{1}[G]min[V2] < 2609.0 ∧ [G]min[V25] < 2063.0 ∧ {2}V45 ≥ 1046.0 ∧ V115 ≥ 1443.0	Self-Blocking Bricks	360	30	0.0833333	1.0	9.0	4
{1}([G]min[V2] < 2609.0) ∧ {2}V28 ≥ 1185.0 ∧ V115 < 1443.0	Gravel	360	29	0.0805556	0.965517	8.68966	3
{1}([G]min[V2] < 2609.0) ∧ {2}V116 ≥ 1330.0 ∧ V115 < 1443.0 ∧ V28 < 1185.0	Self-Blocking Bricks	360	7	0.0194444	1.0	9.0	4
{1}⟨G⟩min[V13] ≥ 1386.0 ∧ [G]min[V2] < 2609.0 ∧ {2}V116 < 1330.0 ∧ V115 < 1443.0 ∧ V28 < 1185.0	Gravel	360	5	0.0138889	1.0	9.0	5
{1}[G]min[V2] < 2609.0 ∧ [G]min[V13] < 1386.0 ∧ {2}V45 ≥ 1046.0 ∧ V115 < 1443.0 ∧ V28 < 1185.0 ∧ V116 < 1330.0	Self-Blocking Bricks	360	4	0.0111111	0.5	4.5	6
{1}(⟨G⟩min[V68] < 702.0) ∧ {2}(V45 < 1046.0)	Trees	360	40	0.111111	0.975	8.775	2
{1}([G]min[V68] ≥ 702.0) ∧ {2}V891 ≥ 2573.0 ∧ V45 < 1046.0	Meadows	360	20	0.0555556	0.9	8.1	3
{1}⟨G⟩min[V7] ≥ 640.0 ∧ [G]min[V68] ≥ 702.0 ∧ {2}V742 ≥ 2298.0 ∧ V45 < 1046.0 ∧ V891 < 2573.0	Bare Soil	360	19	0.0527778	1.0	9.0	5
{1}[G]min[V68] ≥ 702.0 ∧ [G]min[V7] < 640.0 ∧ {2}V897 ≥ 1652.0 ∧ V45 < 1046.0 ∧ V891 < 2573.0 ∧ V742 ≥ 2298.0	Meadows	360	4	0.0111111	0.5	4.5	6
{1}⟨G⟩(min[V100] < 2098.0 ∧ min[V83] ≥ 1947.0) ∧ [G]min[V68] ≥ 702.0 ∧ {2}V742 < 2298.0 ∧ V45 < 1046.0 ∧ V891 < 2573.0	Bare Soil	360	12	0.0333333	1.0	9.0	6
{1}⟨G⟩min[V100] < 2098.0 ∧ [G]min[V68] ≥ 702.0 ∧ [G](min[V100] < 2098.0 → min[V83] < 1947.0) ∧ {2}V897 ≥ 1652.0 ∧ V45 < 1046.0 ∧ V891 < 2573.0 ∧ V742 < 2298.0	Meadows	360	5	0.0138889			

```
0.6 │     5.4 │      8 │
│                                                {1}[G]min[V68] ≥ 702.0 ∧
[G]min[V100] ≥ 2098.0 ∧ {2}V897 ≥ 1652.0 ∧ V45 < 1046.0 ∧ V891 < 2573.0 ∧
V742 < 2298.0 │              Meadows │        360 │       16 │ 0.0444444 │
1.0 │     9.0 │      6 │
│
{1}(⟨G⟩min[V40] < 306.0) ∧ {2}(V897 < 1652.0) │              Shadows │
360 │       40 │  0.111111 │        1.0 │     9.0 │      2 │
│
{1}([G]min[V40] ≥ 306.0) ∧ {2}V139 ≥ 1247.0 ∧ V897 < 1652.0 │
Bitumen │        360 │       39 │  0.108333 │   0.974359 │ 8.76923 │
3 │
│
{1}([G]min[V40] ≥ 306.0) ∧ {2}V125 ≥ 1179.0 ∧ V897 < 1652.0 ∧ V139 < 1247.
0 │              Asphalt │        360 │        7 │ 0.0194444 │   0.714286
│ 6.42857 │      4 │
│
{1}⟨G⟩min[V30] < 1143.0 ∧ [G]min[V40] ≥ 306.0 ∧ {2}V125 < 1179.0 ∧ V897 <
1652.0 │              Asphalt │        360 │       34 │ 0.0944444 │
1.0 │     9.0 │      4 │
│
{1}[G]min[V40] ≥ 306.0 ∧ [G]min[V30] ≥ 1143.0 ∧ {2}V897 < 1652.0 ∧ V125 <
1179.0 │              Gravel │        360 │        5 │ 0.0138889 │
1.0 │     9.0 │      4 │
└──────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────────────────────────────────
─────────────┴───────────────────────┴────────────┴───────────┴────────────┴─
───────────┴──────────┴─────────┘
```

Extra: let's retrain our model, but in cross-validation! (it will take some time...)

In [19]:
```julia
# If you have more time, train in cross-validation!
e = evaluate!(
    machine(model, X_multimodal, y; scitype_check_level=0);
    resampling=StratifiedCV(rng = Random.Xoshiro(1), shuffle=true, nfolds
    measures=[accuracy],
    verbosity=0,
    check_measure=false
)
```

```
[ Info: Precomputing logiset...
```

Out[19]:
```
PerformanceEvaluation object with these fields:
  model, measure, operation,
  measurement, per_fold, per_observation,
  fitted_params_per_fold, report_per_fold,
  train_test_rows, resampling, repeats
Extract:
```

measure	operation	measurement
Accuracy()	predict_mode	0.753

per_fold	1.96*SE
[0.767, 0.739]	0.0385

In [20]:
```julia
# Test accuracies per fold
```

```julia
e.per_fold
```

Out[20]:
```
1-element Vector{Vector{Float64}}:
 [0.7666666666666667, 0.7388888888888889]
```

In [21]:
```julia
dtrees = map((((train_idxs, test_idxs), rep),)->begin
    predictions, tree_test = rep.sprinkle(
        slicedataset(X_multimodal, test_idxs),
        y[test_idxs];
        simplify = true
    )
    tree_test
end, zip(e.train_test_rows, e.report_per_fold))
```

```
Out[21]: 2-element Vector{DecisionTree{String}}:
 ▣ {2}((V62 ≥ 798.0))
├✔ {2}((V897 ≥ 1652.0))
│ ├✔ {1}(⟨G⟩(min[V1] ≥ 2514.0))
│ │ ├✔ Painted metal sheets
│ │ └✘ {1}(⟨G⟩(min[V76] < 2158.0))
│ │   ├✔ {2}((V115 ≥ 1443.0))
│ │   │ ├✔ Self-Blocking Bricks
│ │   │ └✘ {1}(⟨G⟩((min[V76] < 2158.0) ∧ (min[V97] ≥ 2079.0)))
│ │   │   ├✔ Self-Blocking Bricks
│ │   │   └✘ {2}((V64 ≥ 1148.0))
│ │   │     ├✔ Gravel
│ │   │     └✘ Self-Blocking Bricks
│ │   └✘ Bare Soil
│ └✘ {2}((V40 ≥ 1129.0))
│   ├✔ Bitumen
│   └✘ Asphalt
└✘ {1}(⟨G⟩(min[V69] < 854.0))
  ├✔ {1}(⟨G⟩((min[V69] < 854.0) ∧ (min[V69] < 253.0)))
  │ ├✔ Shadows
  │ └✘ Trees
  └✘ {1}(⟨G⟩(min[V25] ≥ 957.0))
    ├✔ {1}(⟨G⟩((min[V25] ≥ 957.0) ∧ (min[V3] < 699.0)))
    │ ├✔ Trees
    │ └✘ Bare Soil
    └✘ {2}((V762 ≥ 2080.0))
      ├✔ {1}(⟨G⟩(min[V28] ≥ 876.0))
      │ ├✔ {1}(⟨G⟩((min[V28] ≥ 876.0) ∧ (min[V18] ≥ 648.0)))
      │ │ ├✔ Meadows
      │ │ └✘ Bare Soil
      │ └✘ Meadows
      └✘ Bare Soil

 ▣ {1}(⟨G⟩(min[V100] < 1485.0))
├✔ {1}(⟨G⟩((min[V100] < 1485.0) ∧ ⟨PO⟩(max[V83] < 1390.0)))
│ ├✔ {1}(⟨G⟩((min[V100] < 1485.0) ∧ ⟨PO⟩((max[V83] < 1390.0) ∧ (max[V24]
< 296.0))))
│ │ ├✔ Shadows
│ │ └✘ Asphalt
│ └✘ Bitumen
└✘ {2}((V31 ≥ 1033.0))
  ├✔ {1}(⟨G⟩(min[V21] ≥ 2278.0))
  │ ├✔ Painted metal sheets
  │ └✘ {2}((V179 ≥ 1538.0))
  │   ├✔ Self-Blocking Bricks
  │   └✘ {1}(⟨G⟩(min[V13] ≥ 1351.0))
  │     ├✔ Gravel
  │     └✘ Self-Blocking Bricks
  └✘ {1}(⟨G⟩(min[V68] < 702.0))
    ├✔ Trees
    └✘ {2}((V906 ≥ 2540.0))
      ├✔ Meadows
      └✘ {2}((V59 ≥ 600.0))
        ├✔ {1}(⟨G⟩(min[V15] ≥ 876.0))
        │ ├✔ Bare Soil
        │ └✘ Meadows
        └✘ Bare Soil
```

```
In [22]: ruleslist = vcat(listrules.(dtrees)...)
```

```
Out[22]: 28-element Vector{ClassificationRule{String}}:
 ▣ {1}(⟨G⟩(min[V1] ≥ 2514.0)) ∧ {2}((V897 ≥ 1652.0))  ↣  Painted metal s
heets

 ▣ {1}⟨G⟩(min[V76] < 2158.0) ∧ [G](min[V1] < 2514.0) ∧ {2}((V115 ≥ 144
3.0))  ↣  Self-Blocking Bricks

 ▣ {1}⟨G⟩((min[V76] < 2158.0) ∧ (min[V97] ≥ 2079.0)) ∧ [G](min[V1] < 251
4.0) ∧ {2}((V115 < 1443.0))  ↣  Self-Blocking Bricks

 ▣ {1}[G]((min[V76] < 2158.0) → (min[V97] < 2079.0)) ∧ [G](min[V1] < 251
4.0) ∧ {2}(V64 ≥ 1148.0) ∧ (V115 < 1443.0)  ↣  Gravel

 ▣ {1}⟨G⟩(min[V76] < 2158.0) ∧ [G](min[V1] < 2514.0) ∧ [G]((min[V76] < 2
158.0) → (min[V97] < 2079.0)) ∧ {2}(V897 ≥ 1652.0) ∧ (V115 < 1443.0) ∧
(V64 < 1148.0)  ↣  Self-Blocking Bricks

 ▣ {1}[G](min[V1] < 2514.0) ∧ [G](min[V76] ≥ 2158.0) ∧ {2}(V62 ≥ 798.0)
∧ (V897 ≥ 1652.0)  ↣  Bare Soil

 ▣ {2}(V40 ≥ 1129.0) ∧ (V897 < 1652.0)  ↣  Bitumen

 ▣ {2}(V62 ≥ 798.0) ∧ (V897 < 1652.0) ∧ (V40 < 1129.0)  ↣  Asphalt

 ▣ {1}(⟨G⟩((min[V69] < 854.0) ∧ (min[V69] < 253.0))) ∧ {2}((V62 < 79
8.0))  ↣  Shadows

 ▣ {1}⟨G⟩(min[V69] < 854.0) ∧ [G]((min[V69] < 854.0) → (min[V69] ≥ 25
3.0)) ∧ {2}((V62 < 798.0))  ↣  Trees

 ▣ {1}⟨G⟩((min[V25] ≥ 957.0) ∧ (min[V3] < 699.0)) ∧ [G](min[V69] ≥ 85
4.0) ∧ {2}((V62 < 798.0))  ↣  Trees

 ▣ {1}⟨G⟩(min[V25] ≥ 957.0) ∧ [G](min[V69] ≥ 854.0) ∧ [G]((min[V25] ≥ 95
7.0) → (min[V3] ≥ 699.0)) ∧ {2}((V62 < 798.0))  ↣  Bare Soil

 ▣ {1}⟨G⟩((min[V28] ≥ 876.0) ∧ (min[V18] ≥ 648.0)) ∧ [G](min[V69] ≥ 85
4.0) ∧ [G](min[V25] < 957.0) ∧ {2}(V762 ≥ 2080.0) ∧ (V62 < 798.0)  ↣  Me
adows

 ⋮
 ▣ {1}(⟨G⟩((min[V100] < 1485.0) ∧ ⟨PO⟩((max[V83] < 1390.0) ∧ (max[V24] <
296.0))))  ↣  Shadows

 ▣ {1}⟨G⟩((min[V100] < 1485.0) ∧ ⟨PO⟩(max[V83] < 1390.0)) ∧ [G]((min[V10
0] < 1485.0) → [PO]((max[V83] < 1390.0) → (max[V24] ≥ 296.0)))  ↣  Aspha
lt

 ▣ {1}⟨G⟩(min[V100] < 1485.0) ∧ [G]((min[V100] < 1485.0) → [PO](max[V83]
≥ 1390.0))  ↣  Bitumen

 ▣ {1}⟨G⟩(min[V21] ≥ 2278.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}((V31 ≥ 103
3.0))  ↣  Painted metal sheets

 ▣ {1}[G](min[V21] < 2278.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}((V179 ≥ 153
8.0))  ↣  Self-Blocking Bricks

 ▣ {1}⟨G⟩(min[V13] ≥ 1351.0) ∧ [G](min[V100] ≥ 1485.0) ∧ [G](min[V21] <
2278.0) ∧ {2}((V179 < 1538.0))  ↣  Gravel
```

```
▣ {1}[G](min[V100] ≥ 1485.0) ∧ [G](min[V21] < 2278.0) ∧ [G](min[V13] <
1351.0) ∧ {2}(V31 ≥ 1033.0) ∧ (V179 < 1538.0)  ↣  Self-Blocking Bricks

▣ {1}⟨G⟩(min[V68] < 702.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}((V31 < 103
3.0))  ↣  Trees

▣ {1}[G](min[V68] ≥ 702.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}(V906 ≥ 254
0.0) ∧ (V31 < 1033.0)  ↣  Meadows

▣ {1}⟨G⟩(min[V15] ≥ 876.0) ∧ [G](min[V100] ≥ 1485.0) ∧ [G](min[V68] ≥ 7
02.0) ∧ {2}(V59 ≥ 600.0) ∧ (V31 < 1033.0) ∧ (V906 < 2540.0)  ↣  Bare Soi
l

▣ {1}[G](min[V100] ≥ 1485.0) ∧ [G](min[V68] ≥ 702.0) ∧ [G](min[V15] < 8
76.0) ∧ {2}(V31 < 1033.0) ∧ (V906 < 2540.0) ∧ (V59 ≥ 600.0)  ↣  Meadows

▣ {1}[G](min[V100] ≥ 1485.0) ∧ [G](min[V68] ≥ 702.0) ∧ {2}(V31 < 103
3.0) ∧ (V906 < 2540.0) ∧ (V59 < 600.0)  ↣  Bare Soil
```

In [23]:
```
# Every symbolic model (including ruleslist) can have has additional info
# attached
println(ruleslist[1])

ruleinfo = SoleModels.info(ruleslist[1])
println(keys(ruleinfo))
```

```
▣ {1}(⟨G⟩(min[V1] ≥ 2514.0)) ∧ {2}((V897 ≥ 1652.0))  ↣  Painted metal shee
ts

(:supporting_labels, :supporting_predictions, :shortform)
```

In [24]:
```
ruleinfo[:supporting_predictions] |> length
```

Out[24]:
```
180
```

In [25]:
```
sort(readmetrics.(ruleslist), by=x->x[:coverage], rev = true)
```

```
Out[25]: 28-element Vector{@NamedTuple{ninstances::Int64, ncovered::Int64, covera
         ge::Float64, confidence::Float64, lift::Float64, natoms::Int64}}:
          (ninstances = 180, ncovered = 27, coverage = 0.15, confidence = 0.59259
         25925925926, lift = 5.333333333333333, natoms = 5)
          (ninstances = 180, ncovered = 25, coverage = 0.1388888888888889, confid
         ence = 0.64, lift = 5.760000000000001, natoms = 3)
          (ninstances = 180, ncovered = 21, coverage = 0.11666666666666667, confi
         dence = 0.9523809523809523, lift = 8.571428571428571, natoms = 2)
          (ninstances = 180, ncovered = 21, coverage = 0.11666666666666667, confi
         dence = 0.9523809523809523, lift = 8.571428571428571, natoms = 3)
          (ninstances = 180, ncovered = 21, coverage = 0.11666666666666667, confi
         dence = 0.5714285714285714, lift = 5.142857142857143, natoms = 4)
          (ninstances = 180, ncovered = 20, coverage = 0.1111111111111111, confid
         ence = 0.85, lift = 7.65, natoms = 3)
          (ninstances = 180, ncovered = 20, coverage = 0.1111111111111111, confid
         ence = 1.0, lift = 9.0, natoms = 3)
          (ninstances = 180, ncovered = 20, coverage = 0.1111111111111111, confid
         ence = 0.95, lift = 8.55, natoms = 3)
          (ninstances = 180, ncovered = 19, coverage = 0.10555555555555556, confi
         dence = 0.9473684210526315, lift = 8.526315789473685, natoms = 4)
          (ninstances = 180, ncovered = 18, coverage = 0.1, confidence = 1.0, lif
         t = 9.0, natoms = 3)
          (ninstances = 180, ncovered = 18, coverage = 0.1, confidence = 0.611111
         1111111112, lift = 5.500000000000001, natoms = 5)
          (ninstances = 180, ncovered = 17, coverage = 0.09444444444444444, confi
         dence = 0.7058823529411765, lift = 6.352941176470589, natoms = 2)
          (ninstances = 180, ncovered = 17, coverage = 0.09444444444444444, confi
         dence = 0.7647058823529411, lift = 6.88235294117647, natoms = 3)
          ⋮
          (ninstances = 180, ncovered = 10, coverage = 0.05555555555555555, confi
         dence = 0.0, lift = 0.0, natoms = 4)
          (ninstances = 180, ncovered = 9, coverage = 0.05, confidence = 0.666666
         6666666666, lift = 6.0, natoms = 4)
          (ninstances = 180, ncovered = 7, coverage = 0.03888888888888889, confid
         ence = 0.7142857142857143, lift = 6.428571428571429, natoms = 6)
          (ninstances = 180, ncovered = 6, coverage = 0.03333333333333333, confid
         ence = 0.3333333333333333, lift = 3.0, natoms = 6)
          (ninstances = 180, ncovered = 5, coverage = 0.027777777777777776, confi
         dence = 1.0, lift = 9.0, natoms = 4)
          (ninstances = 180, ncovered = 4, coverage = 0.022222222222222223, confi
         dence = 0.25, lift = 2.25, natoms = 4)
          (ninstances = 180, ncovered = 4, coverage = 0.022222222222222223, confi
         dence = 0.25, lift = 2.25, natoms = 4)
          (ninstances = 180, ncovered = 4, coverage = 0.022222222222222223, confi
         dence = 1.0, lift = 9.0, natoms = 6)
          (ninstances = 180, ncovered = 2, coverage = 0.011111111111111112, confi
         dence = 0.5, lift = 4.5, natoms = 7)
          (ninstances = 180, ncovered = 2, coverage = 0.011111111111111112, confi
         dence = 0.5, lift = 4.5, natoms = 5)
          (ninstances = 180, ncovered = 1, coverage = 0.005555555555555556, confi
         dence = 1.0, lift = 9.0, natoms = 7)
          (ninstances = 180, ncovered = 1, coverage = 0.005555555555555556, confi
         dence = 0.0, lift = 0.0, natoms = 5)
```

```
In [26]: goodrules = sort(ruleslist, by=r->readmetrics(r)[:coverage], rev = true)
         printmodel.(goodrules; show_metrics = true, threshold_digits = 4);
```

▣ {1}⟨G⟩((min[V100] < 1485.0) ∧ ⟨PO⟩(max[V83] < 1390.0)) ∧ [G]((min[V100] < 1485.0) → [PO]((max[V83] < 1390.0) → (max[V24] ≥ 296.0))) ↣ Asphalt : (ninstances = 180, ncovered = 27, coverage = 0.15, confidence = 0.59, lift = 5.33, natoms = 5)
▣ {2}(V62 ≥ 798.0) ∧ (V897 < 1652.0) ∧ (V40 < 1129.0) ↣ Asphalt : (ninstances = 180, ncovered = 25, coverage = 0.14, confidence = 0.64, lift = 5.76, natoms = 3)
▣ {1}(⟨G⟩(min[V1] ≥ 2514.0)) ∧ {2}((V897 ≥ 1652.0)) ↣ Painted metal sheets : (ninstances = 180, ncovered = 21, coverage = 0.12, confidence = 0.95, lift = 8.57, natoms = 2)
▣ {1}⟨G⟩(min[V21] ≥ 2278.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}((V31 ≥ 1033.0)) ↣ Painted metal sheets : (ninstances = 180, ncovered = 21, coverage = 0.12, confidence = 0.95, lift = 8.57, natoms = 3)
▣ {1}⟨G⟩(min[V13] ≥ 1351.0) ∧ [G](min[V100] ≥ 1485.0) ∧ [G](min[V21] < 2278.0) ∧ {2}((V179 < 1538.0)) ↣ Gravel : (ninstances = 180, ncovered = 21, coverage = 0.12, confidence = 0.57, lift = 5.14, natoms = 4)
▣ {1}⟨G⟩(min[V76] < 2158.0) ∧ [G](min[V1] < 2514.0) ∧ {2}((V115 ≥ 1443.0)) ↣ Self-Blocking Bricks : (ninstances = 180, ncovered = 20, coverage = 0.11, confidence = 0.85, lift = 7.65, natoms = 3)
▣ {1}(⟨G⟩((min[V69] < 854.0) ∧ (min[V69] < 253.0))) ∧ {2}((V62 < 798.0)) ↣ Shadows : (ninstances = 180, ncovered = 20, coverage = 0.11, confidence = 1.0, lift = 9.0, natoms = 3)
▣ {1}⟨G⟩(min[V68] < 702.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}((V31 < 1033.0)) ↣ Trees : (ninstances = 180, ncovered = 20, coverage = 0.11, confidence = 0.95, lift = 8.55, natoms = 3)
▣ {1}⟨G⟩(min[V69] < 854.0) ∧ [G]((min[V69] < 854.0) → (min[V69] ≥ 253.0)) ∧ {2}((V62 < 798.0)) ↣ Trees : (ninstances = 180, ncovered = 19, coverage = 0.11, confidence = 0.95, lift = 8.53, natoms = 4)
▣ {1}(⟨G⟩((min[V100] < 1485.0) ∧ ⟨PO⟩((max[V83] < 1390.0) ∧ (max[V24] < 296.0)))) ↣ Shadows : (ninstances = 180, ncovered = 18, coverage = 0.1, confidence = 1.0, lift = 9.0, natoms = 3)
▣ {1}[G](min[V100] ≥ 1485.0) ∧ [G](min[V68] ≥ 702.0) ∧ {2}(V31 < 1033.0) ∧ (V906 < 2540.0) ∧ (V59 < 600.0) ↣ Bare Soil : (ninstances = 180, ncovered = 18, coverage = 0.1, confidence = 0.61, lift = 5.5, natoms = 5)
▣ {2}(V40 ≥ 1129.0) ∧ (V897 < 1652.0) ↣ Bitumen : (ninstances = 180, ncovered = 17, coverage = 0.09, confidence = 0.71, lift = 6.35, natoms = 2)
▣ {1}⟨G⟩(min[V100] < 1485.0) ∧ [G]((min[V100] < 1485.0) → [PO](max[V83] ≥ 1390.0)) ↣ Bitumen : (ninstances = 180, ncovered = 17, coverage = 0.09, confidence = 0.76, lift = 6.88, natoms = 3)
▣ {1}[G]((min[V76] < 2158.0) → (min[V97] < 2079.0)) ∧ [G](min[V1] < 2514.0) ∧ {2}(V64 ≥ 1148.0) ∧ (V115 < 1443.0) ↣ Gravel : (ninstances = 180, ncovered = 15, coverage = 0.08, confidence = 0.93, lift = 8.4, natoms = 5)
▣ {1}[G](min[V21] < 2278.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}((V179 ≥ 1538.0)) ↣ Self-Blocking Bricks : (ninstances = 180, ncovered = 15, coverage = 0.08, confidence = 0.73, lift = 6.6, natoms = 3)
▣ {1}[G](min[V69] ≥ 854.0) ∧ [G](min[V25] < 957.0) ∧ [G](min[V28] < 876.0) ∧ {2}(V62 < 798.0) ∧ (V762 ≥ 2080.0) ↣ Meadows : (ninstances = 180, ncovered = 11, coverage = 0.06, confidence = 0.64, lift = 5.73, natoms = 5)
▣ {1}⟨G⟩((min[V25] ≥ 957.0) ∧ (min[V3] < 699.0)) ∧ [G](min[V69] ≥ 854.0) ∧ {2}((V62 < 798.0)) ↣ Trees : (ninstances = 180, ncovered = 10, coverage = 0.06, confidence = 0.0, lift = 0.0, natoms = 4)
▣ {1}[G](min[V68] ≥ 702.0) ∧ [G](min[V100] ≥ 1485.0) ∧ {2}(V906 ≥ 2540.0) ∧ (V31 < 1033.0) ↣ Meadows : (ninstances = 180, ncovered = 9, coverage = 0.05, confidence = 0.67, lift = 6.0, natoms = 4)
▣ {1}[G](min[V100] ≥ 1485.0) ∧ [G](min[V68] ≥ 702.0) ∧ [G](min[V15] < 876.0) ∧ {2}(V31 < 1033.0) ∧ (V906 < 2540.0) ∧ (V59 ≥ 600.0) ↣ Meadows : (ninstances = 180, ncovered = 7, coverage = 0.04, confidence = 0.71, lift = 6.43, natoms = 6)
▣ {1}⟨G⟩(min[V15] ≥ 876.0) ∧ [G](min[V100] ≥ 1485.0) ∧ [G](min[V68] ≥ 702.0) ∧ {2}(V59 ≥ 600.0) ∧ (V31 < 1033.0) ∧ (V906 < 2540.0) ↣ Bare Soil :

```
(ninstances = 180, ncovered = 6, coverage = 0.03, confidence = 0.33, lift = 3.0, natoms = 6)
▣ {1}[G](min[V69] ≥ 854.0) ∧ [G](min[V25] < 957.0) ∧ {2}(V62 < 798.0) ∧ (V762 < 2080.0)  ↣  Bare Soil : (ninstances = 180, ncovered = 5, coverage = 0.03, confidence = 1.0, lift = 9.0, natoms = 4)
▣ {1}⟨G⟩((min[V76] < 2158.0) ∧ (min[V97] ≥ 2079.0)) ∧ [G](min[V1] < 2514.0) ∧ {2}((V115 < 1443.0))  ↣  Self-Blocking Bricks : (ninstances = 180, ncovered = 4, coverage = 0.02, confidence = 0.25, lift = 2.25, natoms = 4)
▣ {1}[G](min[V1] < 2514.0) ∧ [G](min[V76] ≥ 2158.0) ∧ {2}(V62 ≥ 798.0) ∧ (V897 ≥ 1652.0)  ↣  Bare Soil : (ninstances = 180, ncovered = 4, coverage = 0.02, confidence = 0.25, lift = 2.25, natoms = 4)
▣ {1}⟨G⟩((min[V28] ≥ 876.0) ∧ (min[V18] ≥ 648.0)) ∧ [G](min[V69] ≥ 854.0) ∧ [G](min[V25] < 957.0) ∧ {2}(V762 ≥ 2080.0) ∧ (V62 < 798.0)  ↣  Meadows : (ninstances = 180, ncovered = 4, coverage = 0.02, confidence = 1.0, lift = 9.0, natoms = 6)
▣ {1}⟨G⟩(min[V76] < 2158.0) ∧ [G](min[V1] < 2514.0) ∧ [G]((min[V76] < 2158.0) → (min[V97] < 2079.0)) ∧ {2}(V897 ≥ 1652.0) ∧ (V115 < 1443.0) ∧ (V64 < 1148.0)  ↣  Self-Blocking Bricks : (ninstances = 180, ncovered = 2, coverage = 0.01, confidence = 0.5, lift = 4.5, natoms = 7)
▣ {1}⟨G⟩(min[V25] ≥ 957.0) ∧ [G](min[V69] ≥ 854.0) ∧ [G]((min[V25] ≥ 957.0) → (min[V3] ≥ 699.0)) ∧ {2}((V62 < 798.0))  ↣  Bare Soil : (ninstances = 180, ncovered = 2, coverage = 0.01, confidence = 0.5, lift = 4.5, natoms = 5)
▣ {1}⟨G⟩(min[V28] ≥ 876.0) ∧ [G](min[V69] ≥ 854.0) ∧ [G](min[V25] < 957.0) ∧ [G]((min[V28] ≥ 876.0) → (min[V18] < 648.0)) ∧ {2}(V62 < 798.0) ∧ (V762 ≥ 2080.0)  ↣  Bare Soil : (ninstances = 180, ncovered = 1, coverage = 0.01, confidence = 1.0, lift = 9.0, natoms = 7)
▣ {1}[G](min[V100] ≥ 1485.0) ∧ [G](min[V21] < 2278.0) ∧ [G](min[V13] < 1351.0) ∧ {2}(V31 ≥ 1033.0) ∧ (V179 < 1538.0)  ↣  Self-Blocking Bricks : (ninstances = 180, ncovered = 1, coverage = 0.01, confidence = 0.0, lift = 0.0, natoms = 5)
```

Exercise: (if you have time) try with 10 folds!