Mapping Your AI Alarm

Who alerts us to the failures of AI algorithms? This notebook will show you how to create your own alarm for the misbehaving of ensembles of noisy AI judges - binary classifiers.
The alarm will be based on the theorem that showed how you could evaluate error independent trios of binary classifiers. We call it “the independent trio evaluator”. It only works correctly near error independence. This notebook is about treating that “bug” as a “feature”.
The notebook is organized as follows:

1. Alarms : we briefly talk about how all alarms have flaws. Proper safety engineering requires that they be evaluated in their application context.
2. Designing an alarm using the failures of the independent trio evaluator: Since algebraic evaluators can return out of bounds or imaginary numbers, we know they must be wrong in their assumptions. We can use this to design an AI alarm.
3. Mapping an AI alarm: We use the UCI Adult dataset to construct maps of the amount of error correlation that triggers the failure of the independent trio evaluator.

Another Mathematica notebook in the submission - “An Enabling Technology” - shows how you can use these maps to create different applications such as AutoML, error-correction, fairness testing on unlabeled data, etc.

Alarms will fail, good engineers know when

There is no perfect alarm. There cannot be one. No one will ever submit to competitions like Stanford HAI’s AI Audit Challenge the perfect tool to carry out its intended auditing task. Good safety engineering requires that we understand the failure properties of any proposed alarm.
We take it as a given that the only safe AI systems of the future will be based on the use of ensemble methods in one way or another. Our independent trio evaluator is one such ensemble technique that can contribute to that future safety. But it fails. This notebook is about finding out where that happens.
This is an exercise that cannot be carried out with probabilistic evaluators of noisy judges. By construction they will always return sensible answers - always real numbers. They never fail! We think that is a flaw for AI safety.

Algebraic numbers make AI alarms possible

Initial look at the code and confirming it works

Getting the dataset and preparing it for training and testing

Mathematica has powerful built-in functions to help us retrieve public datasets. Public datasets are an important part of the scientific/research/development eco-systems. They help promote transparency and reproducibility of research claims.The running example used throughout this submission is the UCI Adult Dataset. There are various reasons this dataset was chosen:

1. It is public.
2. It is used to train and test binary classifiers.
3. It is used in the AI fairness research literature since it contains two sensitive attributes of concern to society - gender and age.

Mathematica curates an extensive data repository. Perhaps we can find UCI Adult in it.

In[∘]:=
```
ResourceSearch["UCI"]
```

Out[∘]=

Name	ResourceType
Sample Data: UCI Letter	DataResource
Gliders in 2D Cellular Automata	DataResource
Sample Data: Satellite	DataResource

In[∘]:=
```
ResourceSearch["Adult"]
```

Out[∘]=

Name	ResourceT
Sample Data: Time to AIDS Induction	DataResou
Sample Data: Paracou Kimboto Trees	DataResou
Sample Data: Fisher's Cats	DataResou
NYC After–School Jobs and Internships	DataResou

The UPenn Dataset Repository is one place we can find the UCI Adult dataset and other ones like it. Let’s use that.

In[∘]:=
```
{tsvHeader, benchmarkData} = ImportPennMLBenchmarksDataset["adult"];
```

In[ ]:=
```
tsvHeader
```
Out[ ]=
```
{age, workclass, fnlwgt, education, education-num,
 marital-status, occupation, relationship, race, sex, capital-gain,
 capital-loss, hours-per-week, native-country, target}
```

Choosing features for the ensemble of classifiers.

If we could always have classifiers independent on their testing sample errors, then AI safety would be easy to solve by the main theorem in TheCoreConcept.nb. The practical reality is that error correlated classifiers are a reality and should be planned for in any evaluator that claims to successfully monitor them.
The engineering approach taken in this notebook is that we can increase their error independence by training them on different features of the data. Here is a specific partition of the UCI feature set. We will use it to test some initial code.

In[ ]:=
```
featurePartition = {{{2, "Nominal"}, {10, "Nominal"}, {14, "Nominal"}},
   {{4, "Nominal"}, {5, "Nominal"}, {7, "Nominal"}},
   {{6, "Nominal"}, {8, "Nominal"}, {9, "Nominal"}},
   {{1, "Numerical"}, {3, "Numerical"}, {11, "Numerical"}, {12, "Numerical"}}};
```

Preparing the benchmark data for training and tests

Our choices for training and testing sizes. We train on 10K items, test on 18K

In[ ]:=
```
trainTestSplit = Association[
   0 → (RandomSample[benchmarkData[0]] // {Take[#, 5000], Take[#, -6000]} &),
   1 → (RandomSample[benchmarkData[1]] // {Take[#, 5000], Take[#, -12 000]} &)];
```

Let’s check our data sizes came out correctly.

In[ ]:=
```
Map[Length, trainTestSplit, {2}]
```
Out[ ]=
```
<|0 → {5000, 6000}, 1 → {5000, 12 000}|>
```

Looks good. Let’s put it all together with our 1st purely algebraic manipulation.

A quick evaluation of four Nearest Neighbours classifiers.

Our choices for the NearestNeighbors are illustrative.

In[ ]:=
```
searchSet = benchmarkData;
classifierTypes =
  Table[{"NearestNeighbors", "NeighborsNumber" → RandomChoice[{2, 3, 4}],
    "DistributionSmoothing" → 0.1, "NearestMethod" → "KDtree"}, {4}];
classifiersFeatures = featurePartition;
trainTestSplit =
  Association[0 → (RandomSample[searchSet[0]] // {Take[#, 5000], Take[#, -6000]} &),
   1 → (RandomSample[searchSet[1]] // {Take[#, 6000], Take[#, -12 000]} &)];
trainingIndices = Transpose@{
    RandomSample[Range@5000] // Partition[#, 1250] &,
    RandomSample[Range@6000] // Partition[#, 1500] &};
classifiersData = Table[Map[#⟦First@Transpose@features⟧ &, trainTestSplit, {3}],
   {features, classifiersFeatures}];
classifiers = TrainClassifiersDisjoint[classifiersData, classifierTypes,
   trainingIndices, Map[(Last@Transpose@#) &, classifiersFeatures]];
voteCountsByLabel = LabelCounts[classifiers, classifiersData]
```

Out[ ]=
```
<|0 → <|{1, 0, 1, 0} → 1177, {1, 1, 1, 1} → 1016, {0, 1, 1, 1} → 189, {1, 0, 1, 1} → 1124,
   {0, 1, 1, 0} → 145, {1, 0, 0, 0} → 188, {1, 1, 1, 0} → 834, {1, 1, 0, 1} → 168,
   {1, 0, 0, 1} → 194, {1, 1, 0, 0} → 154, {0, 0, 1, 1} → 356, {0, 0, 1, 0} → 375,
   {0, 0, 0, 1} → 20, {0, 1, 0, 0} → 15, {0, 0, 0, 0} → 25, {0, 1, 0, 1} → 20|>,
 1 → <|{1, 1, 1, 1} → 5973, {0, 0, 1, 1} → 298, {1, 1, 1, 0} → 1216, {1, 0, 1, 1} → 1958,
   {1, 1, 0, 1} → 705, {0, 0, 0, 1} → 29, {0, 1, 1, 1} → 480, {1, 0, 1, 0} → 493,
   {1, 1, 0, 0} → 253, {0, 1, 1, 0} → 111, {1, 0, 0, 0} → 78, {1, 0, 0, 1} → 262,
   {0, 0, 1, 0} → 80, {0, 1, 0, 0} → 14, {0, 1, 0, 1} → 41, {0, 0, 0, 0} → 9|>|>
```

The voteCountsByLabel data structure is used throughout this notebook. It allows us to benchmark the evaluation on unlabeled data. To do so, we need to have their voting frequencies broken down by true label.

In[ ]:=
```
AlgebraicallyEvaluateClassifiers[classifiers, classifiersData]
```
Out[ ]=

$$\{\langle| P_{\alpha} \to 0.333333,\ P_{1,\alpha} \to 0.190833,\ P_{2,\alpha} \to 0.5765,\ P_{3,\alpha} \to 0.130667,$$
$$P_{4,\alpha} \to 0.4855,\ P_{1,\beta} \to 0.9115,\ P_{2,\beta} \to 0.73275,\ P_{3,\beta} \to 0.884083,\ P_{4,\beta} \to 0.812167,$$
$$\Gamma_{1,2,\alpha} \to 0.0193179,\ \Gamma_{1,3,\alpha} \to -0.0116022,\ \Gamma_{1,4,\alpha} \to 0.00068375,\ \Gamma_{2,3,\alpha} \to -0.00416267,$$
$$\Gamma_{2,4,\alpha} \to 0.0142759,\ \Gamma_{3,4,\alpha} \to 0.000228,\ \Gamma_{1,2,\beta} \to 0.011015,\ \Gamma_{1,3,\beta} \to -0.00250863,$$
$$\Gamma_{1,4,\beta} \to 0.00121008,\ \Gamma_{2,3,\beta} \to 0.000521271,\ \Gamma_{2,4,\beta} \to 0.00480154,$$
$$\Gamma_{3,4,\beta} \to 0.00772699,\ \Gamma_{1,2,3,\alpha} \to -0.0019165,\ \Gamma_{1,2,4,\alpha} \to 0.000756831,$$
$$\Gamma_{1,3,4,\alpha} \to 0.00006048,\ \Gamma_{2,3,4,\alpha} \to -0.00104825,\ \Gamma_{1,2,3,\beta} \to 0.00022748,$$
$$\Gamma_{1,2,4,\beta} \to -0.00015678,\ \Gamma_{1,3,4,\beta} \to 0.000363149,\ \Gamma_{2,3,4,\beta} \to 0.00128837 |\rangle,$$
$$\{\{\{P_{\alpha} \to 0.00133549,\ P_{1,\alpha} \to 55.5284,\ P_{1,\beta} \to 0.951482,\ P_{2,\alpha} \to 0.651268,\ P_{2,\beta} \to 0.630042,$$
$$P_{3,\alpha} \to 0.0505916,\ P_{3,\beta} \to 0.879073\},\ \{P_{\alpha} \to 0.998665,\ P_{1,\alpha} \to 0.0485184,$$
$$P_{1,\beta} \to -54.5284,\ P_{2,\alpha} \to 0.369958,\ P_{2,\beta} \to 0.348732,\ P_{3,\alpha} \to 0.120927,\ P_{3,\beta} \to 0.949408\}\},$$
$$\{\{P_{\alpha} \to 0.30836,\ P_{1,\alpha} \to 0.235844,\ P_{1,\beta} \to 0.927873,\ P_{2,\alpha} \to 0.782649,\ P_{2,\beta} \to 0.813493,$$
$$P_{4,\alpha} \to 0.441636,\ P_{4,\beta} \to 0.781862\},\ \{P_{\alpha} \to 0.69164,\ P_{1,\alpha} \to 0.0721274,\ P_{1,\beta} \to 0.764156,$$
$$P_{2,\alpha} \to 0.186507,\ P_{2,\beta} \to 0.217351,\ P_{4,\alpha} \to 0.218138,\ P_{4,\beta} \to 0.558364\}\},$$
$$\{\{P_{\alpha} \to 0.5 - 1.68639\,i,\ P_{1,\alpha} \to 0.0450327 + 0.0230014\,i,\ P_{1,\beta} \to 0.954967 + 0.0230014\,i,$$
$$P_{3,\alpha} \to 0.0591685 + 0.0182831\,i,\ P_{3,\beta} \to 0.940831 + 0.0182831\,i,$$
$$P_{4,\alpha} \to 0.37952 - 0.0274148\,i,\ P_{4,\beta} \to 0.62048 - 0.0274148\,i\},$$
$$\{P_{\alpha} \to 0.5 + 1.68639\,i,\ P_{1,\alpha} \to 0.0450327 - 0.0230014\,i,\ P_{1,\beta} \to 0.954967 - 0.0230014\,i,$$
$$P_{3,\alpha} \to 0.0591685 - 0.0182831\,i,\ P_{3,\beta} \to 0.940831 - 0.0182831\,i,$$
$$P_{4,\alpha} \to 0.37952 + 0.0274148\,i,\ P_{4,\beta} \to 0.62048 + 0.0274148\,i\}\},$$
$$\{\{P_{\alpha} \to -0.00129093,\ P_{2,\alpha} \to 0.064123,\ P_{2,\beta} \to 0.630061,\ P_{3,\alpha} \to 0.0539898,$$
$$P_{3,\beta} \to 0.879253,\ P_{4,\alpha} \to 72.2653,\ P_{4,\beta} \to 0.620146\},\ \{P_{\alpha} \to 1.00129,\ P_{2,\alpha} \to 0.369939,$$
$$P_{2,\beta} \to 0.935877,\ P_{3,\alpha} \to 0.120747,\ P_{3,\beta} \to 0.94601,\ P_{4,\alpha} \to 0.379854,\ P_{4,\beta} \to -71.2653\}\}\}\}$$

We immediately see that the algebraic evaluator is emitting non-sense answers. It has failed. This means that these classifiers are not independent in their error. How correlated are they?

In[ ]:=
```
GTClassifiers[voteCountsByLabel] // N
```
Out[ ]=

$$\langle| P_{\alpha} \to 0.333333,\ P_{1,\alpha} \to 0.190833,\ P_{2,\alpha} \to 0.5765,\ P_{3,\alpha} \to 0.130667,\ P_{4,\alpha} \to 0.4855,$$
$$P_{1,\beta} \to 0.9115,\ P_{2,\beta} \to 0.73275,\ P_{3,\beta} \to 0.884083,\ P_{4,\beta} \to 0.812167,\ \Gamma_{1,2,\alpha} \to 0.0193179,$$
$$\Gamma_{1,3,\alpha} \to -0.0116022,\ \Gamma_{1,4,\alpha} \to 0.00068375,\ \Gamma_{2,3,\alpha} \to -0.00416267,\ \Gamma_{2,4,\alpha} \to 0.0142759,$$
$$\Gamma_{3,4,\alpha} \to 0.000228,\ \Gamma_{1,2,\beta} \to 0.011015,\ \Gamma_{1,3,\beta} \to -0.00250863,\ \Gamma_{1,4,\beta} \to 0.00121008,$$
$$\Gamma_{2,3,\beta} \to 0.000521271,\ \Gamma_{2,4,\beta} \to 0.00480154,\ \Gamma_{3,4,\beta} \to 0.00772699,\ \Gamma_{1,2,3,\alpha} \to -0.0019165,$$
$$\Gamma_{1,2,4,\alpha} \to 0.000756831,\ \Gamma_{1,3,4,\alpha} \to 0.00006048,\ \Gamma_{2,3,4,\alpha} \to -0.00104825,\ \Gamma_{1,2,3,\beta} \to 0.00022748,$$
$$\Gamma_{1,2,4,\beta} \to -0.00015678,\ \Gamma_{1,3,4,\beta} \to 0.000363149,\ \Gamma_{2,3,4,\beta} \to 0.00128837 |\rangle$$

The Γ expressions are our notation for the sample error correlation of the classifiers. Here we see that these NNs classifiers were at most 1.9% pair correlated but the evaluation failed. Will more testing data fix this?

In[∘]:=
```
searchSet = benchmarkData;
classifierTypes =
  Table[{"NearestNeighbors", "NeighborsNumber" → RandomChoice[{2, 3, 4}],
    "DistributionSmoothing" → 0.1, "NearestMethod" → "KDtree"}, {4}];
classifiersFeatures = featurePartition;
trainTestSplit =
  Association[0 → (RandomSample[searchSet[0]] // {Take[#, 5000], Take[#, -8000]} &),
   1 → (RandomSample[searchSet[1]] // {Take[#, 6000], Take[#, -12 000]} &)];
trainingIndices = Transpose@{
    RandomSample[Range@5000] // Partition[#, 1250] &,
    RandomSample[Range@6000] // Partition[#, 1500] &};
classifiersData = Table[Map[#[[First@Transpose@features]] &, trainTestSplit, {3}],
   {features, classifiersFeatures}];
classifiers = TrainClassifiersDisjoint[classifiersData, classifierTypes,
   trainingIndices, Map[(Last@Transpose@#) &, classifiersFeatures]];
voteCountsByLabel = LabelCounts[classifiers, classifiersData]
```

Out[∘]=
```
<|0 → <|{1, 0, 1, 1} → 1147, {1, 1, 0, 1} → 319, {1, 0, 0, 0} → 171, {1, 1, 1, 1} → 1674,
   {0, 0, 1, 1} → 473, {1, 1, 0, 0} → 233, {1, 0, 1, 0} → 1023, {0, 1, 1, 1} → 535,
   {0, 1, 1, 0} → 444, {1, 1, 1, 0} → 1163, {0, 0, 1, 0} → 493, {0, 1, 0, 1} → 38,
   {1, 0, 0, 1} → 212, {0, 1, 0, 0} → 23, {0, 0, 0, 1} → 26, {0, 0, 0, 0} → 26|>,
 1 → <|{1, 1, 1, 1} → 6259, {0, 1, 1, 1} → 834, {1, 1, 1, 0} → 1059, {1, 1, 0, 1} → 813,
   {1, 1, 0, 0} → 242, {1, 0, 1, 1} → 1438, {0, 1, 0, 1} → 92, {1, 0, 0, 1} → 192,
   {0, 1, 1, 0} → 239, {0, 0, 0, 1} → 54, {0, 1, 0, 0} → 26, {1, 0, 0, 0} → 66,
   {0, 0, 1, 1} → 292, {1, 0, 1, 0} → 288, {0, 0, 1, 0} → 93, {0, 0, 0, 0} → 13|>|>
```

In[∘]:=
```
AlgebraicallyEvaluateClassifiers[classifiers, classifiersData]
```

Out[∘]=

$$\Big\{\Big\langle\Big| P_{\alpha} \to 0.4,\ P_{1,\alpha} \to 0.25725,\ P_{2,\alpha} \to 0.446375,\ P_{3,\alpha} \to 0.131,\ P_{4,\alpha} \to 0.447,\ P_{1,\beta} \to 0.863083,\\
P_{2,\beta} \to 0.797,\ P_{3,\beta} \to 0.875167,\ P_{4,\beta} \to 0.831167,\ \Gamma_{1,2,\alpha} \to 0.01242,\ \Gamma_{1,3,\alpha} \to -0.0195748,\\
\Gamma_{1,4,\alpha} \to 0.00825925,\ \Gamma_{2,3,\alpha} \to -0.00410013,\ \Gamma_{2,4,\alpha} \to 0.0145954,\ \Gamma_{3,4,\alpha} \to -0.001932,\\
\Gamma_{1,2,\beta} \to 0.00987258,\ \Gamma_{1,3,\beta} \to -0.0016751,\ \Gamma_{1,4,\beta} \to 0.00780057,\ \Gamma_{2,3,\beta} \to 0.00174217,\\
\Gamma_{2,4,\beta} \to 0.00406017,\ \Gamma_{3,4,\beta} \to 0.00784064,\ \Gamma_{1,2,3,\alpha} \to -0.000377314,\ \Gamma_{1,2,4,\alpha} \to 0.000552867,\\
\Gamma_{1,3,4,\alpha} \to -0.00077383,\ \Gamma_{2,3,4,\alpha} \to -0.000730223,\ \Gamma_{1,2,3,\beta} \to -0.000982791,\\
\Gamma_{1,2,4,\beta} \to -0.000334524,\ \Gamma_{1,3,4,\beta} \to 0.00140013,\ \Gamma_{2,3,4,\beta} \to 0.0000877299\Big|\Big\rangle,\\
\Big\{\Big\{\Big\{P_{\alpha} \to 0.0527819,\ P_{1,\alpha} \to 3.63023,\ P_{1,\beta} \to 1.00693,\ P_{2,\alpha} \to 0.393696,\ P_{2,\beta} \to 0.704852,\\
P_{3,\alpha} \to 0.0822065,\ P_{3,\beta} \to 0.870187\Big\},\ \Big\{P_{\alpha} \to 0.947218,\ P_{1,\alpha} \to -0.0069261,\\
P_{1,\beta} \to -2.63023,\ P_{2,\alpha} \to 0.295148,\ P_{2,\beta} \to 0.606304,\ P_{3,\alpha} \to 0.129813,\ P_{3,\beta} \to 0.917793\Big\}\Big\},\\
\Big\{\Big\{P_{\alpha} \to 0.379811,\ P_{1,\alpha} \to 0.323302,\ P_{1,\beta} \to 0.899617,\ P_{2,\alpha} \to 0.512006,\ P_{2,\beta} \to 0.82927,\\
P_{4,\alpha} \to 0.469283,\ P_{4,\beta} \to 0.835758\Big\},\ \Big\{P_{\alpha} \to 0.620189,\ P_{1,\alpha} \to 0.100383,\ P_{1,\beta} \to 0.676698,\\
P_{2,\alpha} \to 0.17073,\ P_{2,\beta} \to 0.487994,\ P_{4,\alpha} \to 0.164242,\ P_{4,\beta} \to 0.530717\Big\}\Big\},\\
\Big\{\Big\{P_{\alpha} \to -0.121105,\ P_{1,\alpha} \to -0.358591,\ P_{1,\beta} \to 0.873675,\ P_{3,\alpha} \to -0.0201125,\\
P_{3,\beta} \to 0.888624,\ P_{4,\alpha} \to 0.552852,\ P_{4,\beta} \to 0.690437\Big\},\ \Big\{P_{\alpha} \to 1.1211,\ P_{1,\alpha} \to 0.126325,\\
P_{1,\beta} \to 1.35859,\ P_{3,\alpha} \to 0.111376,\ P_{3,\beta} \to 1.02011,\ P_{4,\alpha} \to 0.309563,\ P_{4,\beta} \to 0.447148\Big\}\Big\},\\
\Big\{\Big\{P_{\alpha} \to -0.0162489,\ P_{2,\alpha} \to 0.0125573,\ P_{2,\beta} \to 0.704252,\ P_{3,\alpha} \to 0.0763269,\\
P_{3,\beta} \to 0.873515,\ P_{4,\alpha} \to 5.60918,\ P_{4,\beta} \to 0.634693\Big\},\ \Big\{P_{\alpha} \to 1.01625,\ P_{2,\alpha} \to 0.295748,\\
P_{2,\beta} \to 0.987443,\ P_{3,\alpha} \to 0.126485,\ P_{3,\beta} \to 0.923673,\ P_{4,\alpha} \to 0.365307,\ P_{4,\beta} \to -4.60918\Big\}\Big\}\Big\}\Big\}$$

Using separate algorithms for training

Increasing the test data made the evaluator get closer to returning sensible answers.
Would using different algorithms improve the algebraic estimates?

In[◦]:=
```
searchSet = benchmarkData;
classifierTypes = {"NearestNeighbors",
   "NeuralNetwork", "SupportVectorMachine", "LogisticRegression"};
classifiersFeatures = featurePartition;
trainTestSplit =
  Association[0 → (RandomSample[searchSet[0]] // {Take[#, 5000], Take[#, -8000]} &),
   1 → (RandomSample[searchSet[1]] // {Take[#, 6000], Take[#, -12 000]} &)];
trainingIndices = Transpose@{
    RandomSample[Range@5000] // Partition[#, 1250] &,
    RandomSample[Range@6000] // Partition[#, 1500] &};
classifiersData = Table[Map[#[[First@Transpose@features]] &, trainTestSplit, {3}],
   {features, classifiersFeatures}];
classifiers = TrainClassifiersDisjoint[classifiersData, classifierTypes,
   trainingIndices, Map[(Last@Transpose@#) &, classifiersFeatures]];
voteCountsByLabel = LabelCounts[classifiers, classifiersData]
```

Out[◦]=
```
<|0 → <|{0, 0, 1, 0} → 163, {1, 0, 1, 1} → 228, {0, 1, 0, 1} → 1482, {0, 0, 0, 0} → 1143,
    {1, 1, 1, 0} → 98, {1, 0, 1, 0} → 182, {0, 1, 0, 0} → 796, {1, 1, 0, 0} → 339,
    {1, 1, 0, 1} → 440, {1, 1, 1, 1} → 103, {0, 0, 0, 1} → 1683, {1, 0, 0, 1} → 534,
    {1, 0, 0, 0} → 418, {0, 1, 1, 0} → 93, {0, 0, 1, 1} → 190, {0, 1, 1, 1} → 108|>,
  1 → <|{1, 1, 1, 1} → 3357, {1, 1, 0, 1} → 778, {1, 0, 1, 1} → 978, {0, 1, 0, 1} → 1638,
    {0, 1, 1, 1} → 2077, {0, 0, 1, 1} → 528, {0, 1, 0, 0} → 447, {0, 0, 0, 0} → 105,
    {1, 0, 1, 0} → 187, {0, 0, 0, 1} → 426, {1, 1, 1, 0} → 581, {1, 1, 0, 0} → 338,
    {0, 0, 1, 0} → 72, {1, 0, 0, 1} → 207, {0, 1, 1, 0} → 192, {1, 0, 0, 0} → 89|>|>
```

In[∘]:=
```
AlgebraicallyEvaluateClassifiers[classifiers, classifiersData]
```

Out[∘]=

$$\Big\{\big\langle\big|\, P_{\alpha} \to 0.4,\ P_{1,\alpha} \to 0.70725,\ P_{2,\alpha} \to 0.567625,\ P_{3,\alpha} \to 0.854375,\ P_{4,\alpha} \to 0.404,\ P_{1,\beta} \to 0.542917,\ P_{2,\beta} \to 0.784,\ P_{3,\beta} \to 0.664333,\ P_{4,\beta} \to 0.832417,\ \Gamma_{1,2,\alpha} \to -0.00407778,\ \Gamma_{1,3,\alpha} \to 0.0337433,\ \Gamma_{1,4,\alpha} \to -0.011354,\ \Gamma_{2,3,\alpha} \to -0.0127146,\ \Gamma_{2,4,\alpha} \to 0.0089295,\ \Gamma_{3,4,\alpha} \to -0.0081675,\ \Gamma_{1,2,\beta} \to -0.00448,\ \Gamma_{1,3,\beta} \to 0.0645724,\ \Gamma_{1,4,\beta} \to -0.00859955,\ \Gamma_{2,3,\beta} \to -0.00358733,\ \Gamma_{2,4,\beta} \to 0.001552,\ \Gamma_{3,4,\beta} \to 0.0253312,\ \Gamma_{1,2,3,\alpha} \to 0.00358161,\ \Gamma_{1,2,4,\alpha} \to 0.00283993,\ \Gamma_{1,3,4,\alpha} \to 0.000100038,\ \Gamma_{2,3,4,\alpha} \to 0.00134294,\ \Gamma_{1,2,3,\beta} \to -0.000305497,\ \Gamma_{1,2,4,\beta} \to -0.00010338,\ \Gamma_{1,3,4,\beta} \to -0.000774948,\ \Gamma_{2,3,4,\beta} \to 0.00137511 \,\big|\big\rangle,$$

$$\Big\{\Big\{\big\{P_{\alpha} \to 0.475336,\ P_{1,\alpha} \to 0.762825,\ P_{1,\beta} \to 0.629187,\ P_{2,\alpha} \to 0.446768,\ P_{2,\beta} \to 0.724996,\ P_{3,\alpha} \to 0.990639,\ P_{3,\beta} \to 0.862267\big\},\ \big\{P_{\alpha} \to 0.524664,\ P_{1,\alpha} \to 0.370813,\ P_{1,\beta} \to 0.237175,\ P_{2,\alpha} \to 0.275004,\ P_{2,\beta} \to 0.553232,\ P_{3,\alpha} \to 0.137733,\ P_{3,\beta} \to 0.00936095\big\}\Big\},$$

$$\Big\{\big\{P_{\alpha} \to 0.166692,\ P_{1,\alpha} \to 0.681344,\ P_{1,\beta} \to 0.467693,\ P_{2,\alpha} \to 1.03259,\ P_{2,\beta} \to 0.778561,\ P_{4,\alpha} \to 0.443009,\ P_{4,\beta} \to 0.774028\big\},\ \big\{P_{\alpha} \to 0.833308,\ P_{1,\alpha} \to 0.532307,\ P_{1,\beta} \to 0.318656,\ P_{2,\alpha} \to 0.221439,\ P_{2,\beta} \to -0.0325856,\ P_{4,\alpha} \to 0.225972,\ P_{4,\beta} \to 0.556991\big\}\Big\},$$

$$\Big\{\big\{P_{\alpha} \to 0.369662,\ P_{1,\alpha} \to 0.432872,\ P_{1,\beta} \to 0.369967,\ P_{3,\alpha} \to -0.600936,\ P_{3,\beta} \to -0.214099,\ P_{4,\alpha} \to 0.200501,\ P_{4,\beta} \to 0.701696\big\},\ \big\{P_{\alpha} \to 0.630338,\ P_{1,\alpha} \to 0.630033,\ P_{1,\beta} \to 0.567128,\ P_{3,\alpha} \to 1.2141,\ P_{3,\beta} \to 1.60094,\ P_{4,\alpha} \to 0.298304,\ P_{4,\beta} \to 0.799499\big\}\Big\},$$

$$\Big\{\big\{P_{\alpha} \to 0.473531,\ P_{2,\alpha} \to 0.201683,\ P_{2,\beta} \to 0.503965,\ P_{3,\alpha} \to 0.28103,\ P_{3,\beta} \to 0.221087,\ P_{4,\alpha} \to 0.0867062,\ P_{4,\beta} \to 0.580047\big\},\ \big\{P_{\alpha} \to 0.526469,\ P_{2,\alpha} \to 0.496035,\ P_{2,\beta} \to 0.798317,\ P_{3,\alpha} \to 0.778913,\ P_{3,\beta} \to 0.71897,\ P_{4,\alpha} \to 0.419953,\ P_{4,\beta} \to 0.913294\big\}\Big\}\Big\}\Big\}$$

We now have two of the four trios returning sensible answers. This is for classifiers that can be up to 3% error correlated as seen it the top of the output. This is now an operating point at the borderline of failure for the algebraic evaluator based on them being error independent. The rest of this notebook is about automating this exploration to make a map of what correlation values make the evaluator fail.