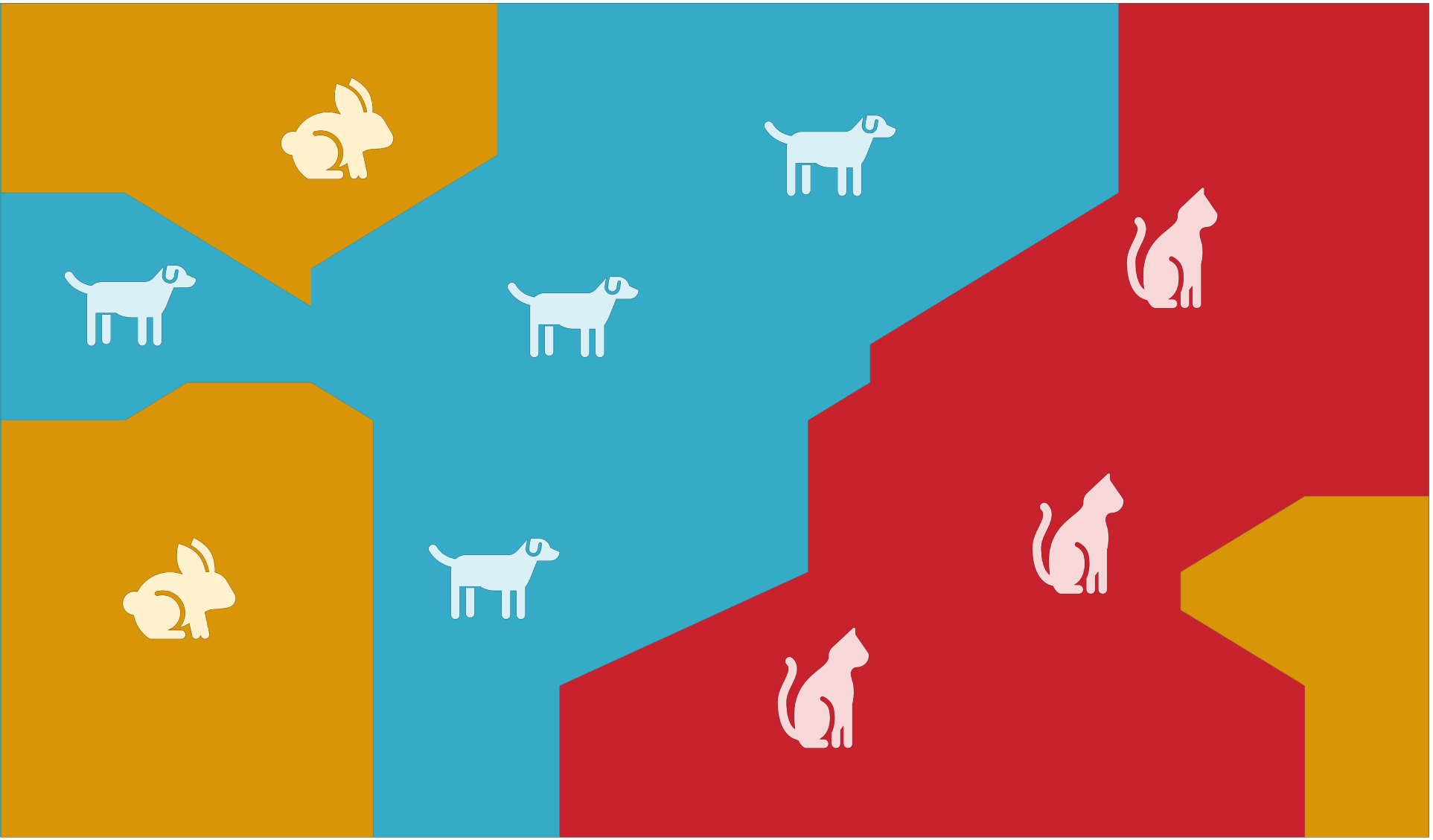
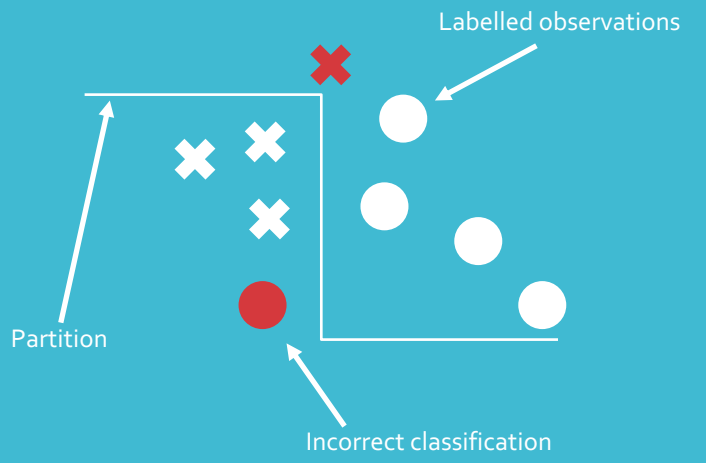


This slideshow aims to explain the ideas behind the codebase in the genetic partitions repo

GENETIC PARTITION CLASSIFIER

CLASSIFICATION IS A SEARCH FOR AN OPTIMAL PARTITIONING OF THE FEATURE SPACE



Consider data represented in a table with rows and columns: call the rows observations, and the columns features, and let there be one distinguished feature which we call the label of each observation.

We want to train a model to predict the label of observations where such a label is not yet known: ***This type of model is known as a classifier***.

Any classifier can be seen as a ***partitioning*** of the feature space, where each block / class[1] in the partition contains (ideally) observations with only one label, or typically, as few as possible distinct labels (Gini impurity is low).

There are many techniques to construct the partitioning of the feature space: either in a "single step" such as SVM, decision tree, logistic regression, or ensemble and iterative methods that refine the partitioning such as gradient boosting machines, random forest, etc.

Here I propose another alternative: **generate a population of random partitions and use a genetic algorithm to find a partition that will best classify the observations.**

1. A partition induces an equivalence relation where the blocks of the partition correspond to distinct classes in the equivalence relation.

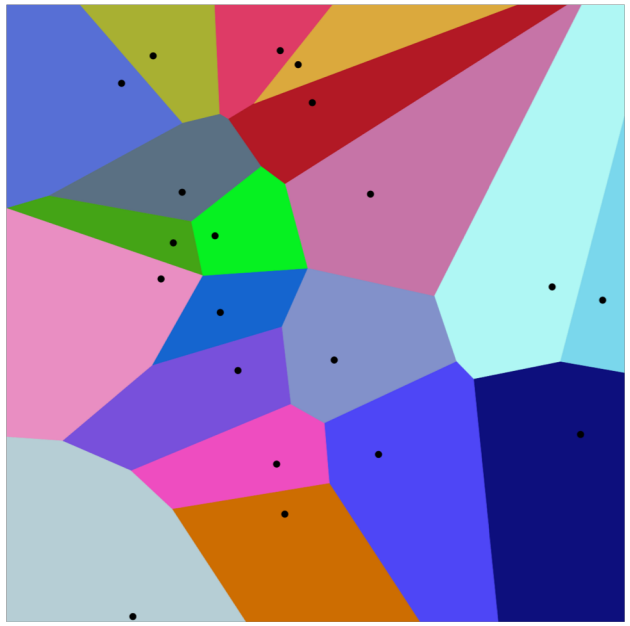
HOW CAN WE REPRESENT PARTITIONS?

In this project partitions are represented as

1. centroids: nodes which represent the centre of a block in the partition.
2. the blocks themselves are then determined in terms of distance to the centroid.

This representation is the same as in a Voronoi diagram.

Note that distance can be measured in different ways, i.e. Manhattan vs. Euclidean, etc. This becomes a parameter that can be specified to the classifier. The default is the Manhattan norm.



Voronoi diagram where centroids are indicated. Here the Euclidean distance determines the blocks.

HOW CAN WE GENERATE PARTITIONS?

With genetic algorithms forming the basis for finding the optimal partitioning, it is desirable to generate partitions as simply as possible. So far we use:

1. Simple random generation: centroids are randomly generated based on bounds parameters specifying the minimum and maximum amount of centers.
2. Clustering: use a fast clustering algorithm and extract the centroids.

HOW DOES THE GENETIC ALGORITHM WORK?

At this stage we use a simple genetic algorithm:

1. The fitness function can be specified in terms of information gain, accuracy or auc.
2. Breeding: Individuals (partitions) with a high score (ito the fitness function) are more likely to breed. Breeding consists of splitting centroids into two groups in each partition and then exchanging groups: this results in two new individuals.
3. Mutation: After breeding mutation is applied. Coordinates of centroids are changed based on a given probability and strength factor.
4. Elitism: the best individuals of a given generation are transferred to the next – the percentage can be specified.
5. Introduction of strangers / aliens: for each generation some new partitions are also generated, in the same way as described on previous slide. This is just to prevent getting stuck in local minima.

THANK YOU

The project is open to anyone: your participation will be much appreciated.