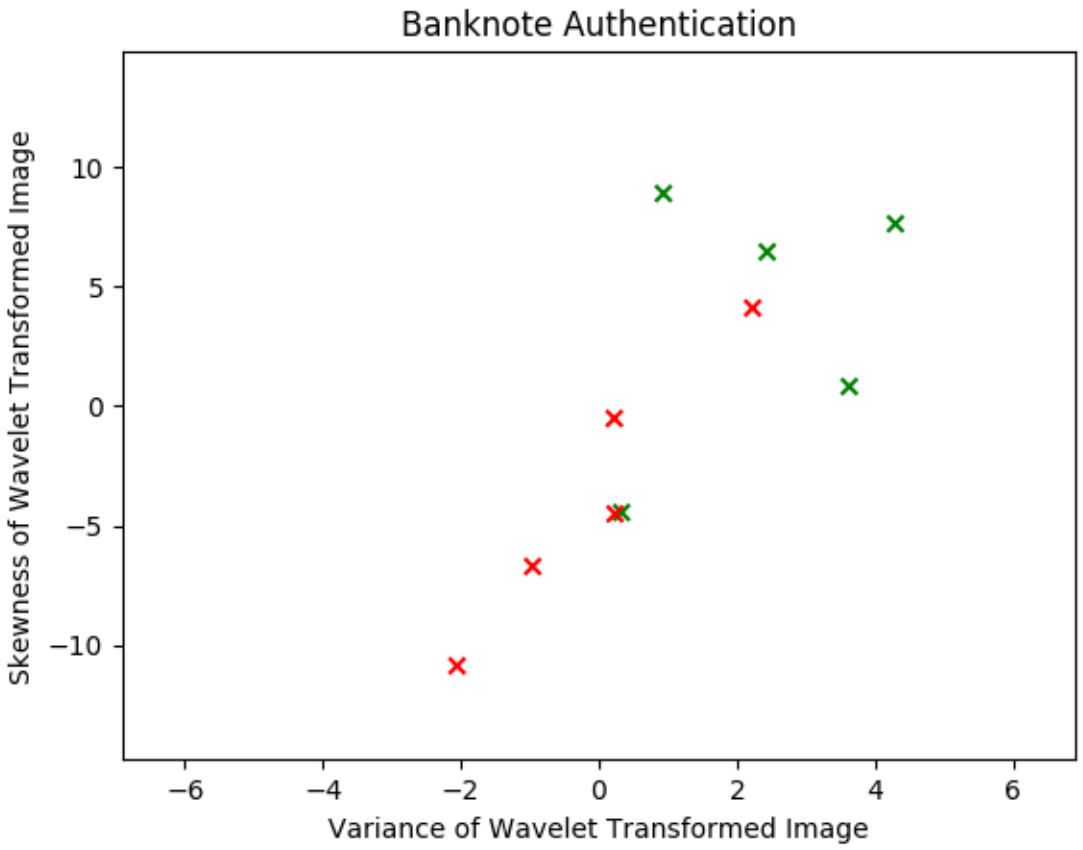
Counterfeit Money Debrief

Solution

The graph below shows the correct labels for the test data you were given. Lest you should feel cheated by this, please keep in mind that this is real world data. Real world data is messy and gray areas are common. We almost never expect to get 100% accuracy. Sometimes 60% to 70% is a big victory.



Questions

1. How accurate was your theory on the training data?

2. How accurate was your theory on the testing data?

3. How could you improve your theory to do better on the testing data?

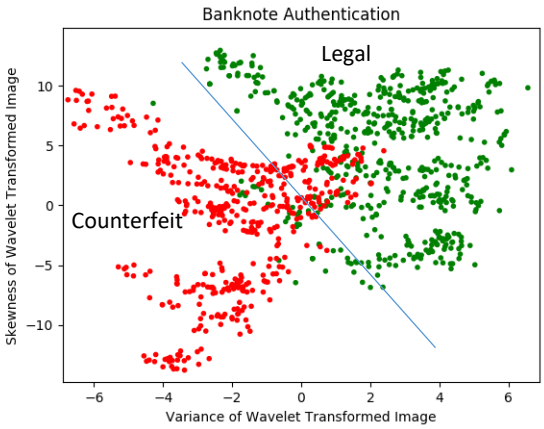
Kinds of Theories for Numeric Data

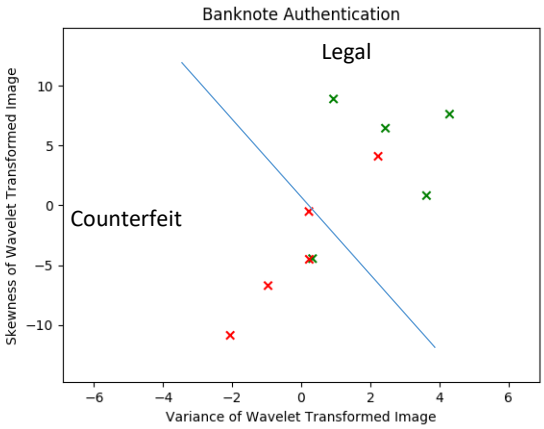
For a data with numeric features, you can think of a "theory" as a way to carve up the two-dimensional **feature space** into regions labelled "Counterfeit" and "Legal". Then previously unseen examples are given a label corresponding to the regions they occupy. Different learning algorithms might carve up the feature space in different ways.

Straight Lines (Linear Classifiers)

Some classification algorithms (e.g. simple Neural Networks and Logistic Regression) form a theory that can be interpreted in two dimensions as a straight line decision boundary. These algorithms work well if the data can be separated well in this a way. Another way of saying this is that they work well when the data is **linearly separable**.

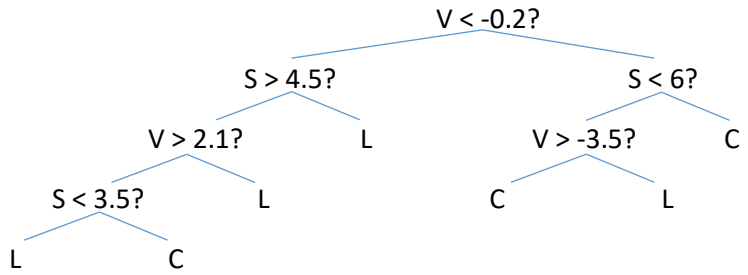
Below is an example of a linear decision boundary that might be used by such a learner. This theory achieves 80% accuracy on the test set.





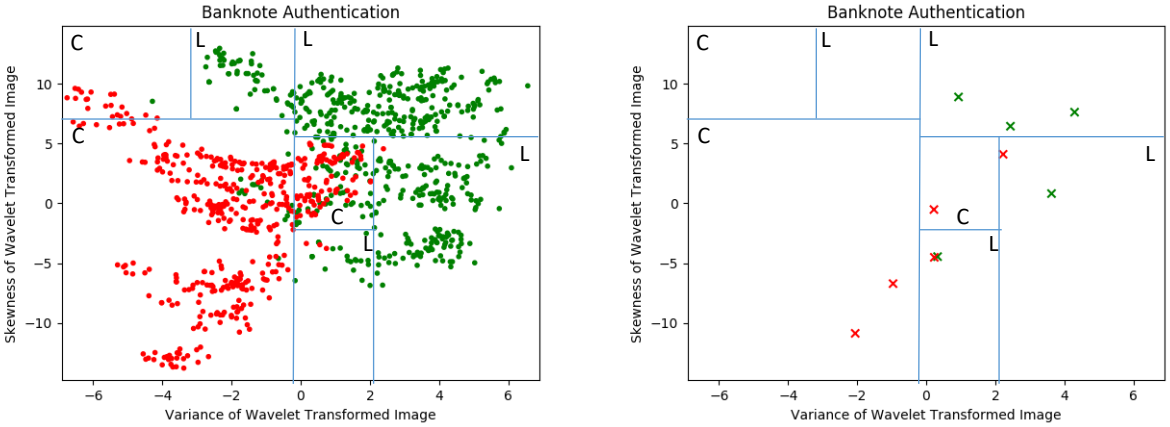
Rectangles (Rule and Decision Tree Classifiers)

Some machine learning algorithms (e.g. those that learn logical rules or decision trees) form a theory that can be interpreted as vertical and horizontal lines chopping up the feature space. Here's an example of a theory that might be learned by such an algorithm (V = variance, S = skewness, C = counterfeit, L = legal).



Note: "Yes" is to the right, "No" is to the left

And here is how this theory breaks up the feature space. It achieves 80% accuracy as well.



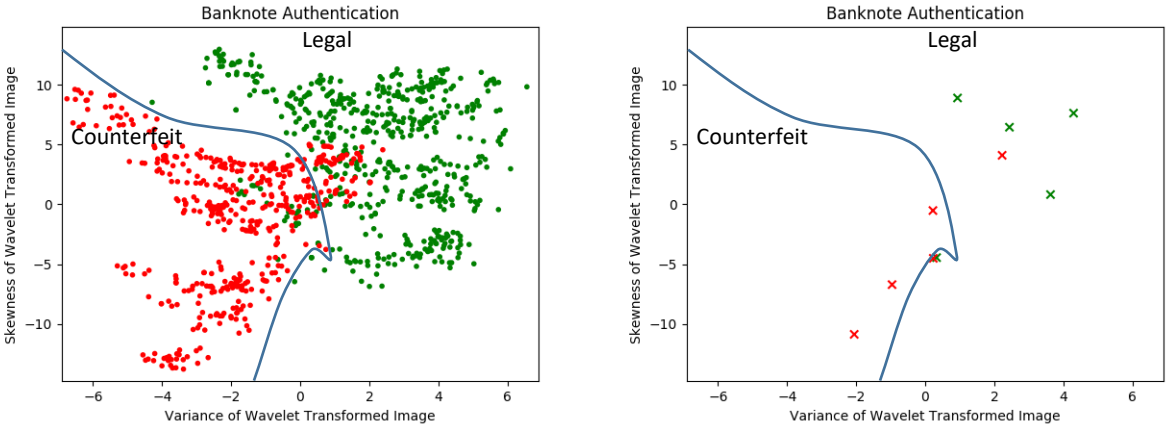
You might notice that in this case, there are a lot of errors in the training set. That is to say, many regions contain both legal and counterfeit examples. This is done on purpose to avoid the problem of **overfitting** in which a learner gets too specific in fitting the idiosyncrasies of the training data, and thereby misses a crucial **generalization** and scores lower on the testing set as a result.

There is nothing to stop the classifier from adding nodes to the decision tree (corresponding to horizontal and vertical decision boundaries in the feature space), effectively chopping the space down into smaller and smaller regions until every region is pure. But this will often not generalize as well to previously unseen data.

Other Shapes (Naïve Bayes, Deep Neural Networks, k-Nearest Neighbour, etc.)

Other machine learning algorithms form theories that carve up the feature space in more complex ways. Some of these techniques can be very powerful, but just like any other learning algorithm, they can end up overfitting the training data and hence performing poorly on unseen data.

On this particular data set, no learning algorithm is likely to do much better than 80% on the test set. An example of a decision boundary that might be formed by a **Deep Learning** Neural Network is shown below.



Final Note on the Counterfeit Money Problem

It is worth noting that this classification problem becomes easier when the other two features (kurtosis and entropy) are included. Including all four features makes the problem more difficult for a human (because it's impossible to visualize a four-dimensional space) but enables some learning algorithms to achieve 100% accuracy on a much larger test set.

Summary: Machine Learning for Classification

Supervised classification is the task of applying pre-defined labels to previously unseen objects.

The **training data** is a set of examples that have already been labelled. The examples and the labels are given to the learning algorithm. The training set should be large and balanced.

Each example consists of a list of **values** for a pre-defined set of **features**.

The learning algorithm uses the training data to form a **model** that tells it when to apply each label.

The **testing data** is a set of examples that have already been labelled. These examples are used to test the model and see how accurate it is.

When features are numeric, it is often helpful to think about how examples cluster within the **feature space**, and to think of the model as carving out **regions** within the feature space.

A common problem in machine learning is **overfitting**, in which the training data is learned too well and classification accuracy on the testing data suffers.