Explainability and Model Analysis

- Explainability
- Model Confidence Scores

Explainability

Explainability is a big challenge in machine learning.

I wrote a blog post about the ELI5 library and how it can be applied to NLP models.

Introducing ML customers to explainability early on can be a great way to build trust. A colleague suggests using Streamlit tools to allow customers to play with models and understand the contribution different features have had to a particular decision.

Model Confidence Scores

Many ML classification models can provide a confidence score which tells the user how confident the model is that it has made the correct choice.

The values of these confidence scores and what constitutes a "good" or "bad" score can vary a lot depending on the type and behaviour of the model. We often get asked why a particular model only ever seems to be 20% confident when a different model gives 99% confidence. Here's why that happens.

Confidence vs Certainty

Confidence and certainty are related but distinct concepts. We want models to be confident that they are correct (a high score is allocated to one of the labels and a low score allocated to the remaining labels) but also we want it to be certain it is correct (we want it to rely on features/cues that lead to the actual correct outcome).

Neural networks are known to often be over-confident in their results but still incorrect due to the way that modern deep learning learns. The paper On Calibration of Neural Networks dives into this further.

In order for us to be able to trust confidence scores, models must also be well calibrated.

Confidence in Random Forest Models

Random Forest Models are made up of an ensemble of decision tree models which are trained based on randomly selected sub-samples of the full training set - allowing the trees to learn different feature priorities based on the variance in the data that they are "assigned".

A single decision tree model cannot easily tell you how confident it is - the data is passed in and the algorithm traverses the branches in the tree until it reaches a decision. Within the random forest, confidence is calculated by assigning each tree a "vote" on the outcome class and then working out the distributions of votes across the possible outcome classes as a percentage.

> In scikit-learn decision trees[do have confidence scores](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.predict_proba) calculated as the proportion of each class that ended on a given leaf node during training

Say we train a random forest model containing 100 trees on a company sector/industry classification problem with 5 classes. In theory, some of the trees will learn to prioritise the most important features in the dataset. Likewise, we can assume that some of the trees will be trained on less representative sub-samples of the training data and will prioritise less discriminative features.

When we predict on an unseen data sample we might get an output like this:

> We specialise in using AI to improve user experience for customers of high end grocery stores

35 Trees votes for "Consumer Goods"
25 Trees voted for "IT & Technology"
22 Trees voted for "Health & Beauty"
12 Trees voted for "Retail"
6 Trees voted for "Automotive Manufacturing"

I used 100 random trees in my model for easy maths, so we would say in this case that the model is 35% confident that this description is from a company in consumer goods.

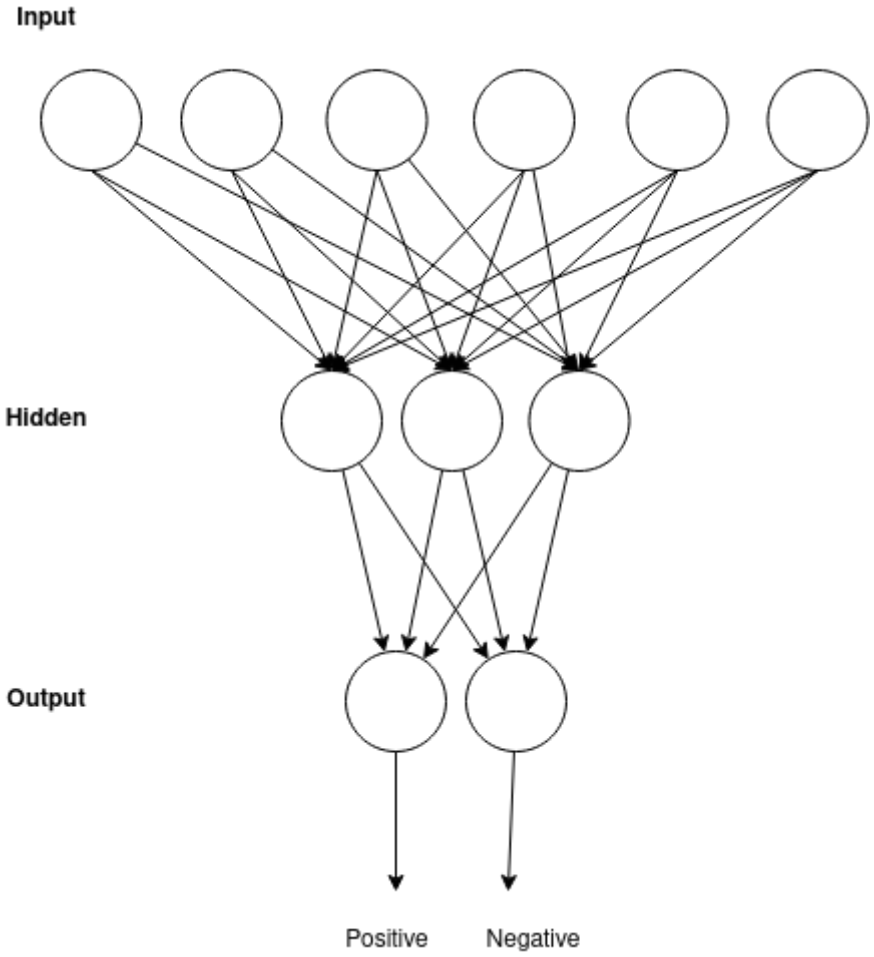
Can We Calibrate Confidence in Random Forest Models?

Confidence calibration is a technique that allows us to more closely map the confidence of an ML model (via its decision function) onto the real life probability that a sample belongs to a particular class.

- Some work has been done on calibration of random forests for probability
- Scikit-Learn provide an out of the box class for providing this functionality CalibratedClassifierCV

Confidence in Neural Networks

Neural classification models can have many different internal structures but tend to have a set of inputs that correspond to the feature vector of the task being learned (e.g. a sparse bag-of-words, a set of RGB pixel values or numerical sensor readings) and a set of outputs equal to the number of classes being predicted.



Activation functions in hidden layers can produce wildly different and un-normalised values depending on the inputs, random initialisations and what is learned (although we can use normalisation constraints in our learning to prevent individual weights from going too far).

The output of the model is therefore typically normalised using SoftMax which essentially takes a weighted average of the results - its a bit like saying "which output received a higher percentage of the overall signal propagated through the network?"

Can We Calibrate Confidence in Neural Networks?

- Some research has been done on this subject