Lesson 5&6 Questions

The chapter summary is really useful:

We suggest starting your analysis with a random forest. This will give you a strong baseline, and you can be confident that it's a reasonable starting point. You can then use that model for feature selection and partial dependence analysis, to get a better understanding of your data.

From that foundation, you can try neural nets and GBMs, and if they give you significantly better results on your validation set in a reasonable amount of time, you can use them. If decision tree ensembles are working well for you, try adding the embeddings for the categorical variables to the data, and see if that helps your decision trees learn better.

For tabular data:

Random forests are the easiest to train, because they are extremely resilient to hyperparameter choices and require very little preprocessing. They are very fast to train, and should not overfit if you have enough trees. But they can be a little less accurate, especially if extrapolation is required, such as predicting future time periods.

Gradient boosting machines in theory are just as fast to train as random forests, but in practice you will have to try lots of different hyperparameters. They can overfit, but they are often a little more accurate than random forests.

Neural networks take the longest time to train, and require extra preprocessing, such as normalization; this normalization needs to be used at inference time as well. They can provide great results and extrapolate well, but only if you are careful with your hyperparameters and take care to avoid overfitting.

1. What is a continuous variable?

Numerical data, data that takes numbers that with preprocessing can be directly fed to a model.

1. What is a categorical variable?

A variable that cannot be fed directly to a model. This is true even if it is a number. E.g. Movie ID, with values 1,2,3,4. Because the numbers don’t mean anything with respect to mathematical operations like addition, we need to convert them.

1. Provide two of the words that are used for the possible values of a categorical variable

Ordinal, meaning there is an implied order to it. E.g. rate satisfaction on a scale 1-5.

Nominal, if the categories are named. E.g. which house type you live in?

A categorical variable can be both types.

Extra:

We need to convert categorical variables into something better for the model to handle.

One approach is one-hot encoding. For sex, male or female, simply create two more columns called male and female, then make them 1 or 0 depending. When it comes to model predictions, if someone is male, then male is 1 and this variable contributes to the prediction, female is 0 and it does not. And vice versa. You could also just remove one of the columns. Removing female for instance does not lose any information since the model can tell sex already from the male column alone.

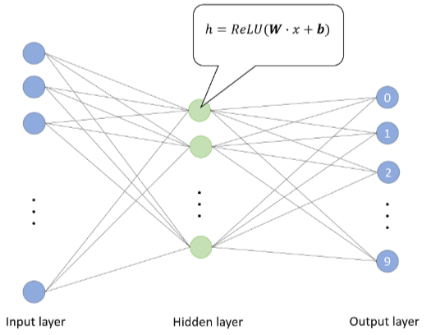
However, there are other approaches to dealing with categorical variables.

Entity embedding can reduce memory usage and speed up neural networks compared to one-hot encoding. It maps similar categorical values close to each other in the embedding space, which can reveal properties about them, which could help with prediction.

1. What is a "dense layer"?

A dense layer is a layer where there is a fully connected layer. So the input data goes to hidden layer(s), then an output layer, where all inputs and outputs at every layer are connected.

For example:



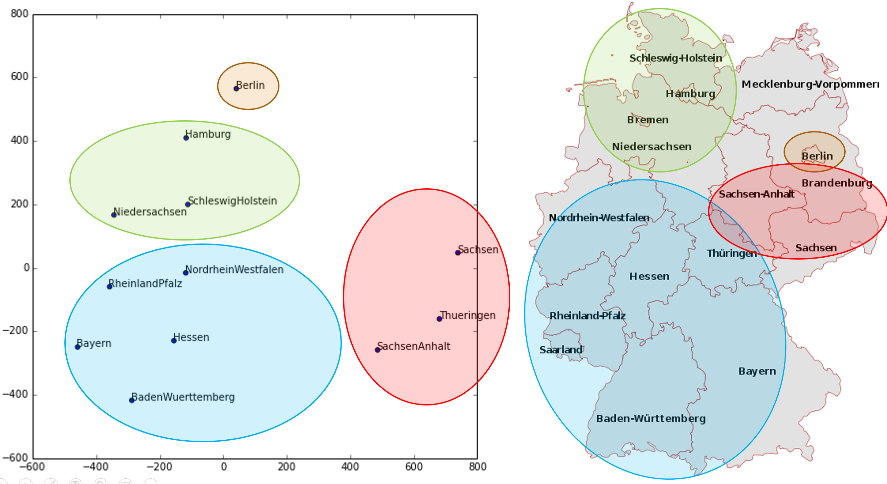
I think of it as a mini-neural network.

1. How do entity embeddings reduce memory usage and speed up neural networks?

Because entity embeddings in comparison to one hot encoding can reduce memory usage and speed up neural networks.

They can embed the categorical variables into inputs that are continuous (so can be fed into the model) but also meaningful (so are both likely better for the model to learn from, and easier for humans to interpret).

For example, say location of German cities for retail stores is a categorical variable. We use entity embedding on it and can get a plot of the embedding matrix: (LHS)



This plot reflects the real physical geography quite well! It is meaningful to both the model and to us humans!

1. What kinds of datasets are entity embeddings especially useful for?

Tabular datasets (or structured data), where there are categorical variables about discrete entities. For example, the location of German retail stores, or the days of the week.

1. What are the two main families of machine learning algorithms?

Recent studies have shown for Modern machine learning that there are only a couple of key techniques that are best for the vast majority of datasets:

1. Ensembles of decision trees (I.E, ensembles of random forests and gradient boosting machines). This is for structured/tabular data.
2. Multilayered neural networks that learn with SGD (I.E, shallow and/or deep learning). This is for unstructured data like audio, images, NLP.

2. is almost always better than 1. For unstructured data. But for structured data, they can give similar results. But 1. is better, because they tend to train faster with less computation, and are often easier to interpret. The latter is critical: we can ask questions like which columns in a tabular dataset are most important for a prediction?

There are two exceptions to this rule: if the tabular data has categorical variables that can take say, thousands of possible values, it is high-cardinality, and 2. might be better. Furthermore, if some data columns contain data like audio, images, or most likely text, 2. is better suited for them.

1. Why do some categorical columns need a special ordering in their classes? How do you do this in Pandas?

We need ordering for ordinal categorical columns so that the model understands the order as we do.

Say we have an ordinal column ProductSize with orderings ‘Medium', 'Small', 'Large / Medium', 'Mini', 'Large', 'Compact'.

sizes = 'Large','Large / Medium','Medium','Small','Mini','Compact'

df['ProductSize'] = df['ProductSize'].astype('category')

df['ProductSize'].cat.set\_categories(sizes, ordered=True, inplace=True)

1. Summarize what a decision tree algorithm does.

A decision tree is just a series of binary splits. Each split is based on the best column and column value to split by. Continue this until each leaf (bottom of the tree) is <= a certain group size we set.

Splitting by best column consecutively is called ‘greedy’ in cs. There may be times where this isn’t optimal. I suppose it depends most on how/how well you evaluate how good a split is.

1. Why is a date different from a regular categorical or continuous variable, and how can you preprocess it to allow it to be used in a model?

A date is kinda like an ordinal value, because it has order, it is meaningful to say 25/09/2022 is higher than 26/09/2022, but there’s other information like the day of the week, whether it’s a holiday etc, so it requires more careful preprocessing and a different approach than regular categorical or continuous variables.

To do this, we replace each date column with a set of date metadata containing this useful information. We replace date with a set of categorical fields like saleWeek, SaleMonth, etc

1. Should you pick a random validation set in the bulldozer competition? If no, what kind of validation set should you pick?

No. It’s time based data, so we should set a validation set to include a time period of dates that are after the training data.

1. What is pickle and what is it useful for?

Pickle is a python system fastai uses to save Python objects. Useful to save the preprocessed data so we don’t have to run all the code cells again in the future.

1. How are mse, samples, and values calculated in the decision tree drawn in this chapter?

The top node represents the entire dataset, before any splits, simply predicting the average value of the independent value of the training data. Each node after binary splits contains the predictions after that split.

The mse, samples, and values, are for each node as described.

The dtreeviz library can plot a decision tree for you.

1. How do we deal with outliers, before building a decision tree?

Decision trees are pretty good at dealing with outliers, they don’t affect predictions too badly, but affect visualisation and understandability.

We could simply set the outliers to be an arbitrary understandable value as a result.

On overfitting: We need a minimum sample size at leaf nodes to ensure overfitting doesn’t occur. Having a limitless number of leaf nodes will be like a game of 20 questions, but with an infinite number of questions. Eventually you’ll just make a path for each individual data point, which is what we see, the number of leaf nodes will equal the number of training data points.

1. How do we handle categorical variables in a decision tree?

We don’t. Research shows ordering them leads to the same performance as not. The tree will figure out itself how to split best for categorical variables, we don’t need to one-hot encode them. We don’t have embedding layers in a decision tree.

1. What is bagging?

Creating many trained models, based on a random subset of the training data, then making an ensemble of them. Each model will have random error in its predictions, but as each model’s prediction errors are uncorrelated with eachother, they do well to cancel out!

Random forest uses bagging, as well as picking random columns too use too.

1. What is the difference between max\_samples and max\_features when creating a random forest?

Max\_samples is the number of rows to sample for each tree. This means like take a random subset of 0.75 of the training data to train each tree on.

Max\_features is how many columns to sample at each split point. A value of 0.5 means to take half of the total number of columns at each split. I think this means how many columns to split with each time. Like a binary split means to split with 1 column. So for a dataset with 2 columns, a max\_features of 0.5 means to binary split?

But random forests are so sensitive to max\_features or max\_samples. Max\_samples at default usually works well, but if you have a lot of data, you may want to increase it to improve training time without lowering accuracy.

1. If you increase n\_estimators to a very high value, can that lead to overfitting? Why or why not?

No. The more trees the better. N\_estimators the highest the best.

It doesn’t overfit because it uses an ensemble approach.

Usually after 30 trees, performance doesn’t improve. =<100 usually is a good idea.

1. In the section "Creating a Random Forest", just after <>, why did preds.mean(0) give the same result as our random forest?

They both are the same thing. The mean of the every tree in the forest’s predictions.

1. What is "out-of-bag-error"?

For each of your decision trees, use the unused training data as validation data. Then find the average validation accuracy as the OOB error.

1. Make a list of reasons why a model's validation set error might be worse than the OOB error. How could you test your hypotheses?

If the OOB error is better than the validation error, then something else other than overfitting is causing non generalisability. For example, for time based data like predicting stock price, perhaps there are other new events like a recession affecting the price action.

1. Explain why random forests are well suited to answering each of the following question:
   * How confident are we in our predictions using a particular row of data?

Find how much each tree’s predictions vary with eachother. If each tree predicts similarly, then we are confident.

* + For predicting with a particular row of data, what were the most important factors, and how did they influence that prediction?

Use the treelinterpreter library.

* + Which columns are the strongest predictors?
  + How do predictions vary as we vary these columns?

Feature\_importances in sklearn tells this by exploring each branch recursively and looking to see what features were used, and how much the model improved as a result.

1. What's the purpose of removing unimportant variables?

Decreases training time. Less columns makes the model more simple and much easier to interpret, making it easier to maintain.

You can also merge very similar columns. Some columns might literally be the same data.

24.5. What is data leakage?

It’s when you model uses data that would not be present or valid in the real world or practically. For example, say you wanted a model to predict whether researchers would receive grants. If the researchers that don’t receive grants always leave the ‘feedback’ column empty, then the model may latch onto that to use as a very accurate predictor, which is obviously not what we want to do.

There’s also the issue when the data used in prediction would not be available during when predictions are being made.

To check for data leakage:

Check whether the accuracy is too good to be true. Look for important predictor columns that don’t make sense. Look for partial dependence plots that don’t make sense.

1. What's a good type of plot for showing tree interpreter results?

A waterfall plot. It shows the positive and negative contributions from the columns result in the final prediction. For example, in predicting car price, that the year made negatively contributed significantly.

1. What is the "extrapolation problem"?

Same as overfitting and not being generalisable.

Random forests are really bad at predicting data outside the range of training data. This means they are especially bad for data where there is a trend over time, like inflation.

1. How can you tell if your test or validation set is distributed in a different way than your training set?

Use a random forest to predict whether a row is in the validation data or training data. If it predicts columns that differ significantly between the two, we should investigate why!

This is a good way to discover subtle domain shift issues.

1. Why do we ensure saleElapsed is a continuous variable, even although it has less than 9,000 distinct values?

We don’t want to treat saleElapsed as a categorical variable. If we did, then it wouldn’t be able to predict outside its range of values, by definition. Because it would just be discrete values. We definitely want a time based variable to be able to work outside of its range of values.

27.5: Using an ensemble between random forests and neural networks can give a nice performance boost!

1. What is "boosting"?

Another meta technique using trees like random forest. Instead of averaging tree predictions, we add them up.

Gradient boosting machines (GBMs) and an example.

Unlike random forest, overfitting is possible, and it is very sensitive to hyperparameters.

1. How could we use embeddings with a random forest? Would we expect this to help?

You obtain embeddings from trained neural networks and use them to improve the performance of other ML methods like random forest and KNN. You use categorical embeddings from the neural network in the random forest instead of raw categorical columns.

1. Why might we not always use a neural net for tabular modeling?

Because it takes long to train, and requires a lot of careful data preprocessing, as well as care to avoid overfitting.