11. Ensembles

Decision trees models are

~ Interpretable

~ They can capture non-linear relationships

~ They don’t require scaling of the data and theoretically can work with categorical features.

But with a single decision trees are likely to overfit.

**Random Forest**:

~ Decide how many decision trees we want to build

~ can control with n\_estimators hyperparameter

~ fit a diverse set of that many decision trees by injecting randomness in the classifier construction

~ predict by voting (classification) or averaging (regression) of predictions given by individual models/trees

Strengths:

1. Don’t require scaling of data

2. Less likely to overfit

3. Slower than decision trees because we are fitting multiple trees but can easily parallelize training because all trees are independent of each other

4. In general, able to capture a much broader picture of the data compared to a single decision tree.

Weaknesses:

1. Require more memory

2. Hard to interpret

3. Tend not to perform well on high dimensional sparse data such as text data

**How Random forest works:**

**1. Create a collection (ensemble) of trees. Grow each tree on an independent bootstrap sample from the data.**

**2. At each node:**

**2.1 Randomly select a subset of features out of all features (independently for each \*\*node\*\*).**

**2.2 Find the best split on the selected features.**

**2.3 Grow the trees to maximum depth.**

**3.Prediction time**

**Vote the trees to get predictions for new example.**

11.1 Ensembles

Model that combines multiple ML models to create a more powerful model

11.2 Gradient boosted tree

unlike random forest where each tree is independent, GBT has no ramdomization and builds trees in a serial manner, where each tree tries to correct the mistakes of the previous one. (GBT combines many simple models/shallow tree with depth 1-5)

hyperparam of GBT:

1.n\_estimators: the higher the more complex

2.learning\_rate: controls how strongly each tree tries to correct the mistakes of the previous trees. Higher means more complex as well

In general for tree based models, you don't need to scale features (that's usually used when you are using linear models)

XGBoost: Not part of sklearn but has similar interface.

~ Supports missing values

~ GPU training, networked parallel training

~ Supports sparse data

~ Typically better scores than random forests

LightGBM: Not part of sklearn but has similar interface.

~ Small model size

~ Faster

~ Typically better scores than random forests

CatBoost: Not part of sklearn but has similar interface.

~ Usually better scores but slower compared to XGBoost and LightGBM

"predict" of random forest is done by voting(classification) or average(regression) of predictions given by individual tree/model

n\_estimators decide how many decision trees we want to build

Ensembles:

1. Averaging: Use different models and let them vote during prediction time. (E.g. XGBoost+LightGBM+CatBoost, just like random forests)

2. Stacking: Use their outputs as inputs to another model.

* By default for classification, it uses logistic regression.
  + We don’t need a complex model here necessarily, more of a weighted average.
  + The features going into the logistic regression are the classifier outputs, not the original features!
  + So the number of coefficients = the number of base estimators!

Graphical user interface, text, application, email

Description automatically generated

What’s going on in here?

* It is **doing cross-validation by itself by default** (see [documentation](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html))
  + It is **fitting the base estimators on the training fold**
  + And the **predicting on the validation fold**
  + And then **fitting the meta-estimator (final estimator** – logistics regression by default) on that output (**on the** **validation fold**)

Note that estimators\_ are fitted on the full X while final\_estimator\_ is trained using cross-validated predictions of the base estimators using cross\_val\_predict.