# **TidyModels**

<https://www.tidymodels.org/start/models/>

## Packages in TidyModels:

* Recipes: Do preprocessing of predictors or create new predictors
* parsnip: 30 machine learning models to choose from and their respective engines
* tune: Hyperparameter tuning for tidymodels packages
* rsample: Random sampling. Resampling of data
* workflows: Bundle pre-processing, modelling, post-processing
* yardstick: Estimate how well models are working
* dials: Tools for tuning hyperparamters in predictive models
* vip: Estimate variable importance
* parallel: Use multiple cores to fit models. Good for model tuning.

## Splitting of Work

* Time Series (Using modeltime) [Ian]
* Feature Engineering [Marcus, Qing Lin, Darren]
  + Current: Lags, Google search trends, VIX
  + Technical Indicators: MACD,
  + Alternative Data: News Articles, Github
  + Fundamental Data: Fed Funds Rate
* Feature Selection [Read up]
* Machine Learning Modelling [Dylan, Lester]
* Metrics

## Overall Flow

1. Data Splitting and Resampling
2. Create Recipe
3. Build Model
   1. Define Parallel Processing, if required
4. Create Workflow
5. Create grid for Tuning
6. Train and tune the Model
7. Collect Metrics
8. Repeat Steps 2 to 7 on another model.
9. Compare models based on collected metrics
10. Use best model for Last Fit
    1. Use the entire training data
    2. Make predictions with test data
    3. Get metrics

## Details

* Initial\_Split(): Partition raw data into train and test. Train and test retain some proportions
* rsample: Apply resampling methods to split data into train, test, validation
  + analysis(): extract resampled “training data”
  + assessment(): extract resampled “test data”
* Recipes: Do preprocessing of predictors or create new predictors
  + Define 2 roles: Outcome ~ Predictors
  + Update\_role: Specify what variables to exclude from the modelling but retained in the dataset (eg. for id)
  + Step\_: Data feature engineering from current data
  + Recipe selections: all\_of(), one\_of(). Apply steps to multiple variables at one time.
* Specify and train models with different engines using **Parnsnip Package**
  + Defines all the machine learning models with a simple interface. Used to add to workflow later on. Create model objects.
  + Set\_engine(): Define method for training model. Specify number of cores for parallel processing if necessary
  + Set\_mode(): Regression/Classification
  + Tune(): Estimate the best values for hyperparameters by training many models on resampled data sets and finding the values with best performance.
    - After tuning, a single numeric value will be selected for each hyperparameter
    - Grid\_regular(): Choose sensible values to try for each hyperparameter. Tries out different combinations.
* Workflow(): Pairs a model and recipe together. Bundle pre-processing, modelling, post-processing
  + Add\_model()
  + Add\_recipe()
    - Add\_formula(): If no recipe was defined
  + Tune\_grid(): If tune was used previously when defining the model. Fit models with different hyperparameters previously defined.
    - Show\_best()
    - Select\_best(): Pull out single set of hyperparameter values that has best performance
    - Finalize\_workflow(): Update workflow with the best model selected
  + Fit(): Train the model on training data
    - If you don’t use recipe, have to define model formula here
  + Pull\_workflow\_fit() %>% tidy(): Get model coefficients
  + Pull\_workflow\_fit() %>% vip(): Get variable importance
  + Tidy(lm\_fit): Return description of fitted model parameters estimates. Nicer view than summary()
  + Fit\_resamples(): Use rsample object previously defined for resampling to fit into model.
    - Collect\_metrics(): Estimate performance of model on “unseen”data
  + Predict(lm\_fit, new\_data = new\_points): Applies recipe to new data, passes them to fitted model.
  + Piping functions from yardstick package to measure performance of models

## Concepts

**Ensemble**: Using multiple machine learning algorithms to obtain better predictive performance than a single algo

**Class imbalance**: Can use recipe (upsample, downsample), themis package

**Resampling**: Simulate how well model will perform on new data since test set should only be used as the final check for model’s performance. Create datasets from training data (split training data) for the purpose of calculating performance metrics without predicting the training set directly as a whole. Models from training are not kept, because their only purpose if calculating performance metrics. The final resampling estimates for the model are the **averages** of the performance statistics replicates. Resampling allows us to simulate how well our model will perform on new data, and the test set acts as the final, unbiased check for our model’s performance.

* Train Set
  + Method 1: Cross-Validation (Diagram 1)
    - Analysis Set
    - Assessment Set
    - Each set is called a fold
  + Method 2: Single resample (Diagram 2)
    - Training
    - Validation
* Test Set

Diagram

Description automatically generated

Diagram

Description automatically generated

**XGBoost:** Optimized distributed gradient boosting library which implements machine learning algorithms under the Gradient Boosting framework

## Bitcoin Rmd File

Objective: Predict 1/0 buy sell signal from a bunch of datapoints. Supervised machine learning.

1. Getting Data (Price, Indicators, Alternative Data, Stock Index)
2. Feature Engineering. Column bind all the data into one big df
3. Predictive Modelling
   1. Define recipe: Datapoints
   2. Define model: Machine learning model and its parameters
   3. Define workflow: Add recipe and model into workflow
   4. Set cross validation
   5. Run model in parallel. Running multiple models, tuning.
4. Making Predictions
   1. Choose best model
   2. Fit resamples
5. Evaluate Model
   1. Visualising results. Compared to benchmark strategy
   2. Descriptive Stats
      1. Confusion Matrix, Log loss
      2. Finance Stats: Returns, SD, Sharpe, Drawdown, Sortino