tidymodels Discussion

Max Kuhn (RStudio)

On the Horizon

There is a project list in the `tidymodels` org that has a list of activities and potential projects that we will be tackling.

Pipelines

As previously mentioned, the modeling *process* includes pre-modeling activities (e.g. feature engineering) as well as post-processing actions such as

- choosing an appropriate probabilitiy threshold
- calibrating probabilities
- appling equivocal zones and model applicability domain analyses

Modeling pipelines exist in python and spark.

Our implmentation will be tidy and allow users to quickly try different cpmbinations of technqiues.

Pipelines Syntax

Suppose we need to impute some data, fit a logistic regression, then choose an appropriate probability threshold.

Although it isn't finalized, the syntax will look something like:

```
data(credit_data)

imputer <-
  recipe(Status ~ ., data = credit_data) %>%
  step_knnimpute(Home, Marital, Job, Income, Assets, Debt) %>%
  step_downsample(Status)

credit_pln <-
  pipeline() %>%
  add_recipe(imputer) %>%
  add_model(logistic_reg() %>% set_engine("glmnet")) %>%
  add_cutoff(0.25)

trained <- fit(credit_pln, training = credit_data)

predict(credit_pln, new_data = new_customer)
```

Automatically Identify Tunable Parameters

```
imputer <-
  recipe(Status ~ ., data = credit_data) %>%
  step_knnimpute(Home, Marital, Job,
                 Income, Assets, Debt,
                 neighbors = varying()) %>%
  step_downsample(Status)

mod <-
  logistic_reg(
    mixture = varying(),
    penalty = varying()
  ) %>%
  set_engine("glmnet")

credit_pln <-
  pipeline() %>%
  add_recipe(imputer) %>%
  add_model(mod) %>%
  add_cutoff(threshold = varying())
```

```
varying_args(credit_pln)
```

```
## # A tibble: 4 x 4
##   name      varying id             type
##   <chr>     <lgl>   <chr>          <chr>
## 1 neighbors TRUE    step_knnimpute step
## 2 penalty   TRUE    model          model_spec
## 3 mixture   TRUE    model          model_spec
## 4 threshold TRUE    cutoff         cutoff
```

Model Tuning Syntax Prototype

```r
resamp <- vfold_cv(credit_data)

grid_search(credit_pln, resamp, levels = 5)

# or
grid_racing(credit_pln, resamp, levels = 5, initial = 3)

# or
rnd_param <- random_search(credit_pln, resamp, size = 25)

# and/or
bayes_search(credit_pln, resamp, initial = rnd_param, num_iter = 20)

# Loop back to the pipeline to update
finalized_pln <-
  update(credit_pln, param_best(bayes_search)) %>%
  fit(training = credit_data)
```

Principles of Modeling Packages and Templates

We are in the process of developing a set of *guidelines* for making good modeling packages. For example:

- Separate the interface that the **modeler** uses from the code to do the computations. They serve two very different purposes.
- Have multiple interfaces (e.g. formula, x/y, etc).
- The *user-facing interface* should use the most appropriate data structures for the data (as opposed to the computations). For example, factor outcomes versus 0/1 indicators and data frames versus matrices.
- `type = "prob"` for class probabilities .
- Use S3 methods.
- The `predict` method should give standardized, predictable results.

Rather than try to make methodologists into software developers, we will provide **GitHub repositories** with template packages that can be used to meet these guidelines (along with documentation and examples on *why*).