Supervised feature selection

DIMENSIONALITY REDUCTION IN R

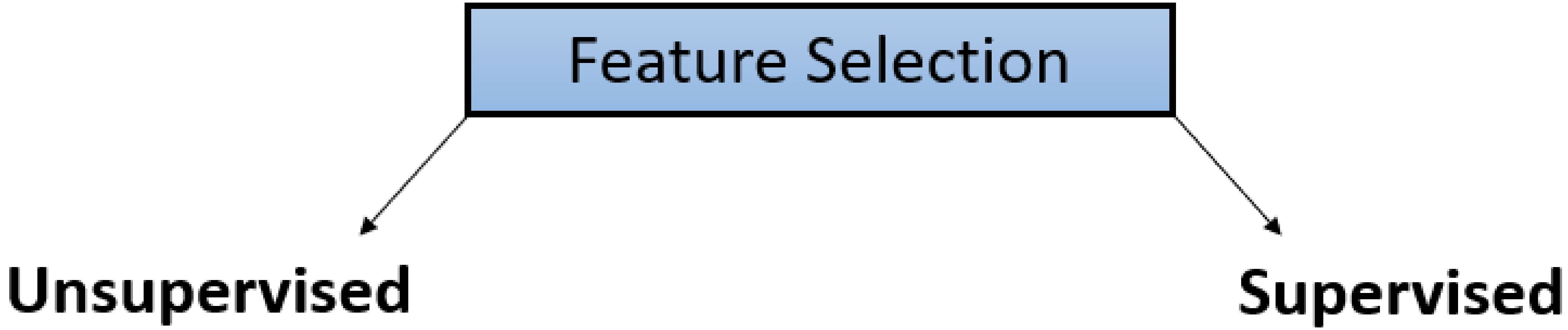
Matt Pickard
Owner, Pickard Predictives, LLC

datacamp

Unsupervised vs. supervised feature selection

Feature Selection

Unsupervised vs. supervised feature selection



Unsupervised vs. supervised feature selection

Feature Selection

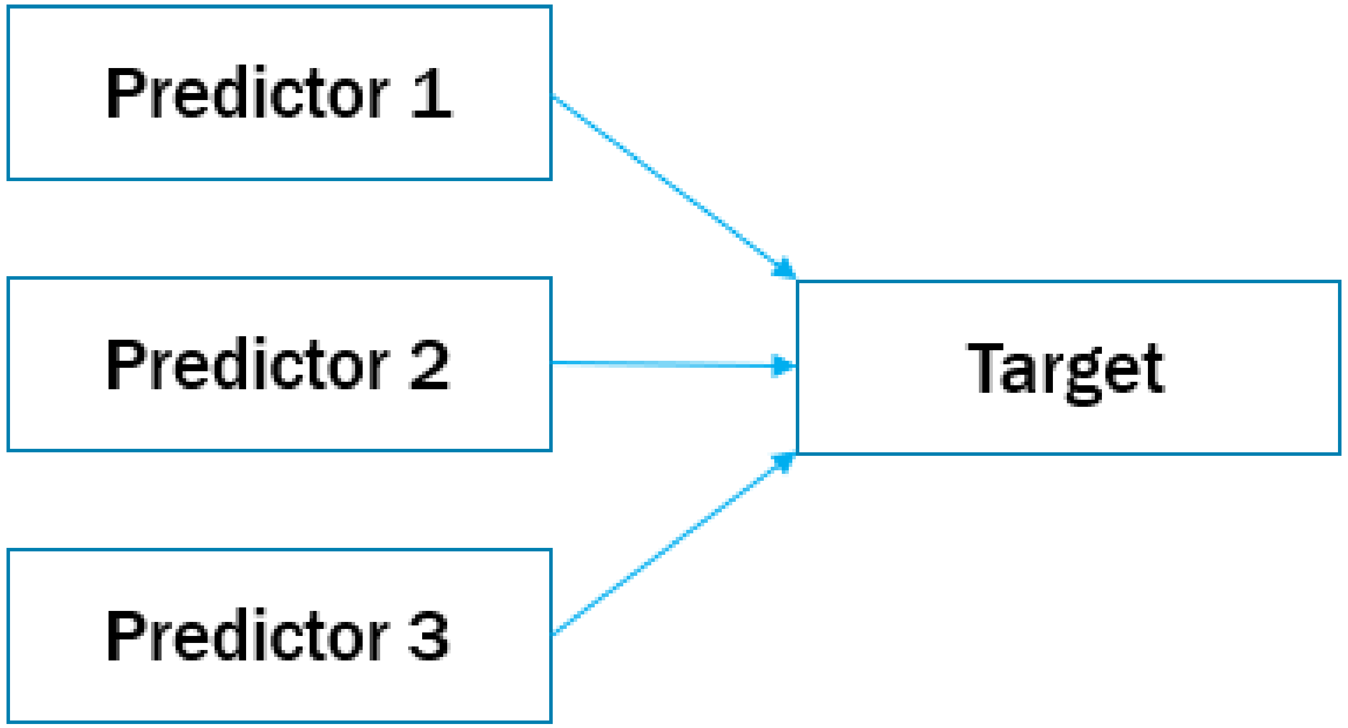
Unsupervised

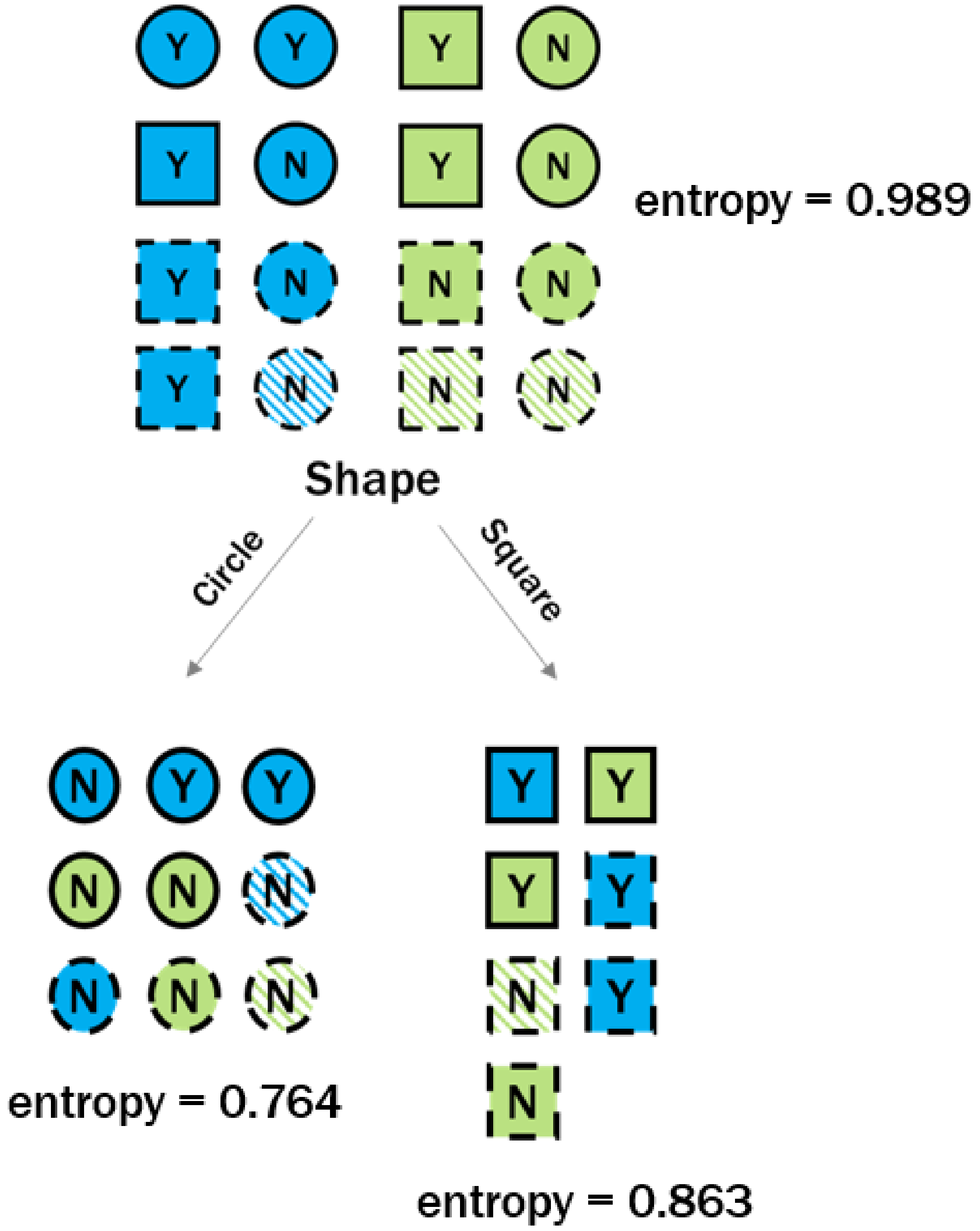
- Drop missing-value features
- Drop low-variance features
- Drop correlated features

Supervised

Supervised feature selection explained

Selection based on information the predictor provides about the target variable





Unsupervised vs. supervised feature selection

Feature Selection

Unsupervised

- Drop missing-value features
- Drop low-variance features
- Drop correlated features

Supervised

- Recursive feature elimination

Unsupervised vs. supervised feature selection

Feature Selection

- **Unsupervised**
  - Drop missing-value features
  - Drop low-variance features
  - Drop correlated features
- **Supervised**
  - Recursive feature elimination
  - Lasso regression

Unsupervised vs. supervised feature selection

Feature Selection

Unsupervised

- Drop missing-value features
- Drop low-variance features
- Drop correlated features

Supervised

- Recursive feature elimination
- Lasso regression
- Random forest

Let's practice!

DIMENSIONALITY REDUCTION IN R

Model Building and Evaluation with tidymodels

DIMENSIONALITY REDUCTION IN R

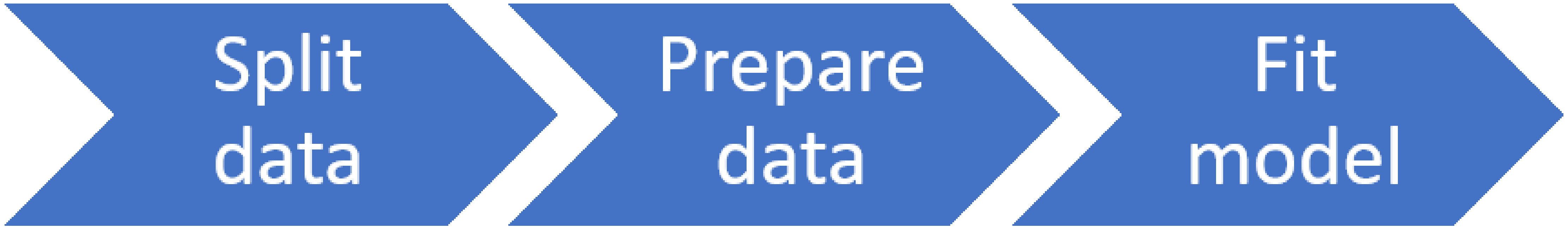
Matt Pickard
Owner, Pickard Predictives, LLC

datacamp

Model fitting process

Model fitting process

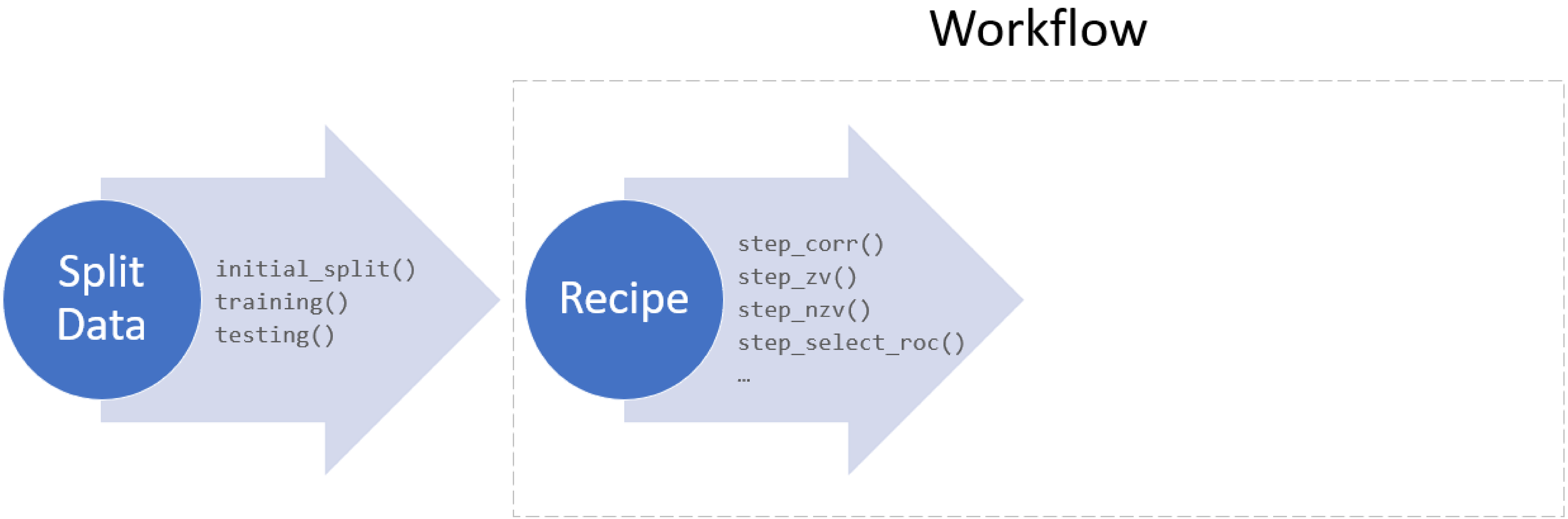
Model fitting process



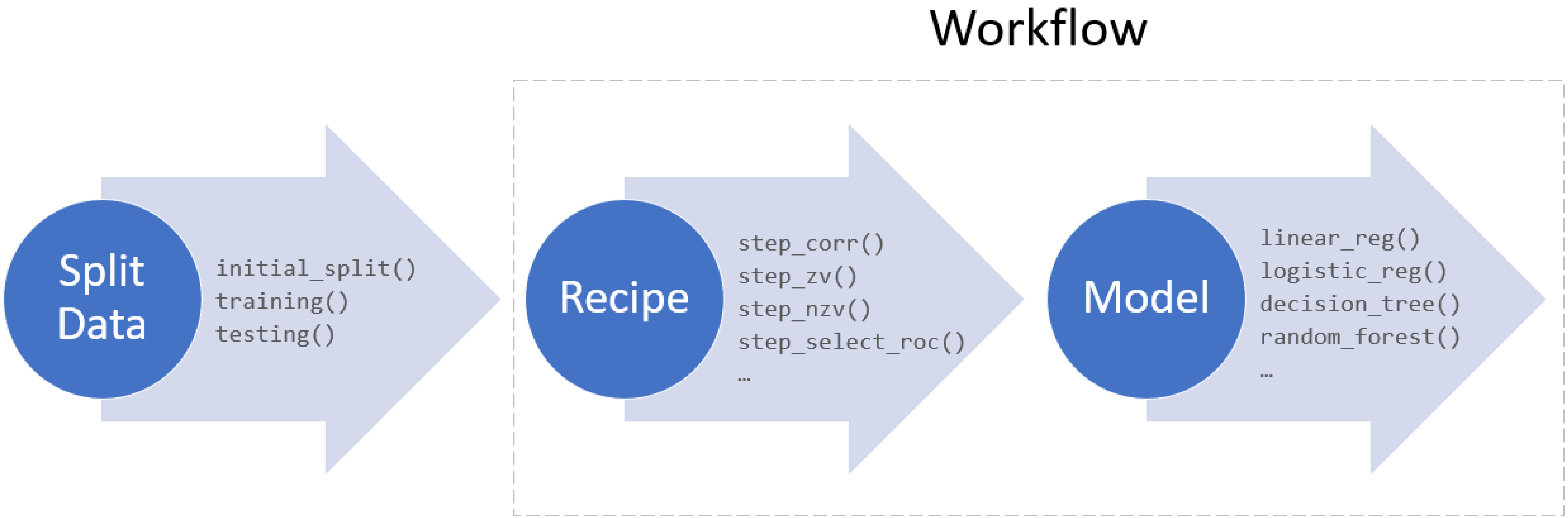
Model fitting process

Model fitting with tidymodels

Model fitting with tidymodels



Model fitting with tidymodels



Splitting out train and test sets

```
split <- initial_split(credit_df, prop = 0.8, strata = credit_score)

train <- split %>% training()

test <-  split %>% testing()
```

Creating a recipe and a model

```
feature_selection_recipe <-
  recipe(credit_score ~ ., data = train) %>%
  step_filter_missing(all_predictors(), threshold = 0.5) %>%
  step_scale(all_numeric_predictors()) %>%
  step_nzv(all_predictors()) %>%
  prep()
```

```
lr_model <- logistic_reg() %>%
  set_engine("glm")
```

Create and fit the workflow

```
credit_wflow <- workflow() %>%
  add_recipe(feature_selection_recipe) %>%
  add_model(lr_model)

credit_fit <-
  credit_wflow %>% fit(data = train)
```

Evaluate the model

```
# Predict test data
credit_pred_df <- predict(credit_fit, test) %>%
  bind_cols(test %>% select(credit_score))

# Evaluate F score
f_meas(credit_pred_df, credit_score, .pred_class)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  macro          0.519
```

Explore the recipe with tidy()

```
tidy(feature_selection_recipe, number = 1)
```

```
# A tibble: 2 × 2
  terms            id
  <chr>            <chr>
1 age              filter_missing_gVVfc
2 outstanding_debt filter_missing_gVVfc
```

Explore the model with tidy()

```r
# Display model estimates
tidy(credit_fit)
```

```
# A tibble: 44 × 5
   term                    estimate std.error statistic p.value
   <chr>                      <dbl>     <dbl>     <dbl>   <dbl>
 1 (Intercept)               2.88       0.918    3.13   0.00173
 2 monthAugust              -0.449      0.236   -1.91   0.0565
 3 monthFebruary            17.7      677.        0.0262 0.979
 4 monthJanuary             17.7      661.        0.0268 0.979
 ...                        ...       ...         ...     ...
```

Let's practice!

DIMENSIONALITY REDUCTION IN R

Lasso Regression

DIMENSIONALITY REDUCTION IN R

Matt Pickard
Owner, Pickard Predictives, LLC

datacamp

Lasso regression overview

- Supervised feature selection
- L1 regularization
- Penalizes the regression coefficients
- Shrinks coefficients
- Less important coefficients shrink to zero
- Naturally performs feature selection

```
linear_reg(engine = "glmnet", penalty = 0.001 , mixture = 1)
```

Standardize data

- Standardize data first, so penalty applies equally across features
- Use `scale()` for target variable
  - returns matrix, so convert to vector with `as.vector()`
- Use `step_normalize()` for predictor variables

Example

```
# Scale target variable
df <- df %>% mutate(target = as.vector(scale(target)))
...
# Scale predictor variables
recipe() %>% step_normalize(all_numeric_predictors())
```

Choosing a penalty value

- Penalty is a hyperparameter to optimize
- Search for the best penalty value
- Use `tune()` in `tidymodels`

```
linear_reg(engine = "glmnet", penalty = tune() , mixture = 1)
```

Preparing the data

Scale the target variable

```
house_sales_subset_df <- house_sales_subset_df %>%
  mutate(price = as.vector(scale(price)))
```

Create the training and testing sets

```
split <- initial_split(house_sales_subset_df, prop = 0.8)
train <- split %>% training()
test <-  split %>% testing()
```

Create a recipe

Create a recipe

```
lasso_recipe <-
  recipe(price ~ ., data = train) %>%
  step_normalize(all_numeric_predictors())
```

Create the workflow

Create the model spec

```
lasso_model <- linear_reg(penalty = 0.01, mixture = 1, engine = "glmnet")
```

Create the workflow

```
lasso_workflow <- workflow(preprocessor = lasso_recipe, spec =  lasso_model)
```

Fit the workflow

```
tidy(lasso_workflow %>% fit(train)) %>% filter(estimate > 0)
```

```
# A tibble: 9 × 3
  term           estimate penalty
  <chr>             <dbl>   <dbl>
1 bathrooms       0.0477     0.01
2 sqft_living     0.434      0.01
3 floors          0.0262     0.01
4 waterfront      0.133      0.01
5 view            0.0510     0.01
6 condition       0.0319     0.01
...               ...        ...
```

Create a tunable model workflow

Create a tunable model spec

```
lasso_model <- linear_reg(penalty = tune(), mixture = 1, engine = "glmnet")
lasso_workflow <- workflow(preprocessor = lasso_recipe, spec =  lasso_model)
```

Create cross validation training sample

```
train_cv <- vfold_cv(train, v = 5)
```

Create grid of penalty values

```
penalty_grid <- grid_regular(penalty(range = c(-3, -1)), levels = 20)
```

- A penalty range of 0.001 to 0.1 is specified as `range = c(-3, -1)`

Fit a grid of models

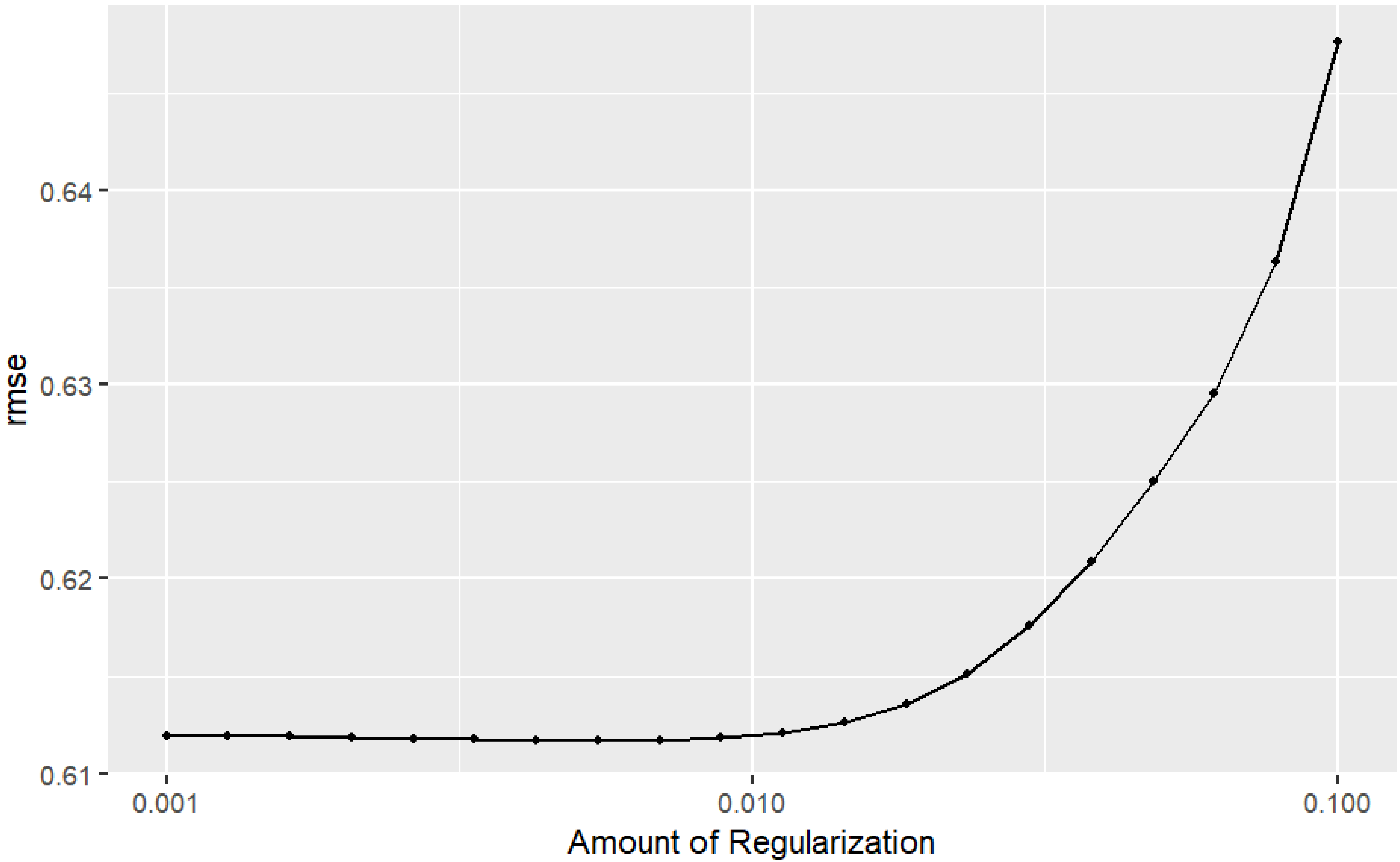
Create grid of fitted models

```
lasso_grid <- tune_grid(
  lasso_workflow,
  resamples = train_cv,
  grid = penalty_grid)
```

Plot model performances

```
autoplot(lasso_grid, metric = "rmse")
```

Penalty performance plot



Finalize the model

Retrieve the penalty value for the best model

```
best_rmse <- lasso_grid %>% select_best("rmse")
```

Refit the best model

```
final_lasso <-
  finalize_workflow(lasso_workflow, best_rmse) %>%
  fit(train)
```

Display the best model's coefficients

```
tidy(final_lasso) %>% filter(estimate > 0)
```

Let's practice!

DIMENSIONALITY REDUCTION IN R

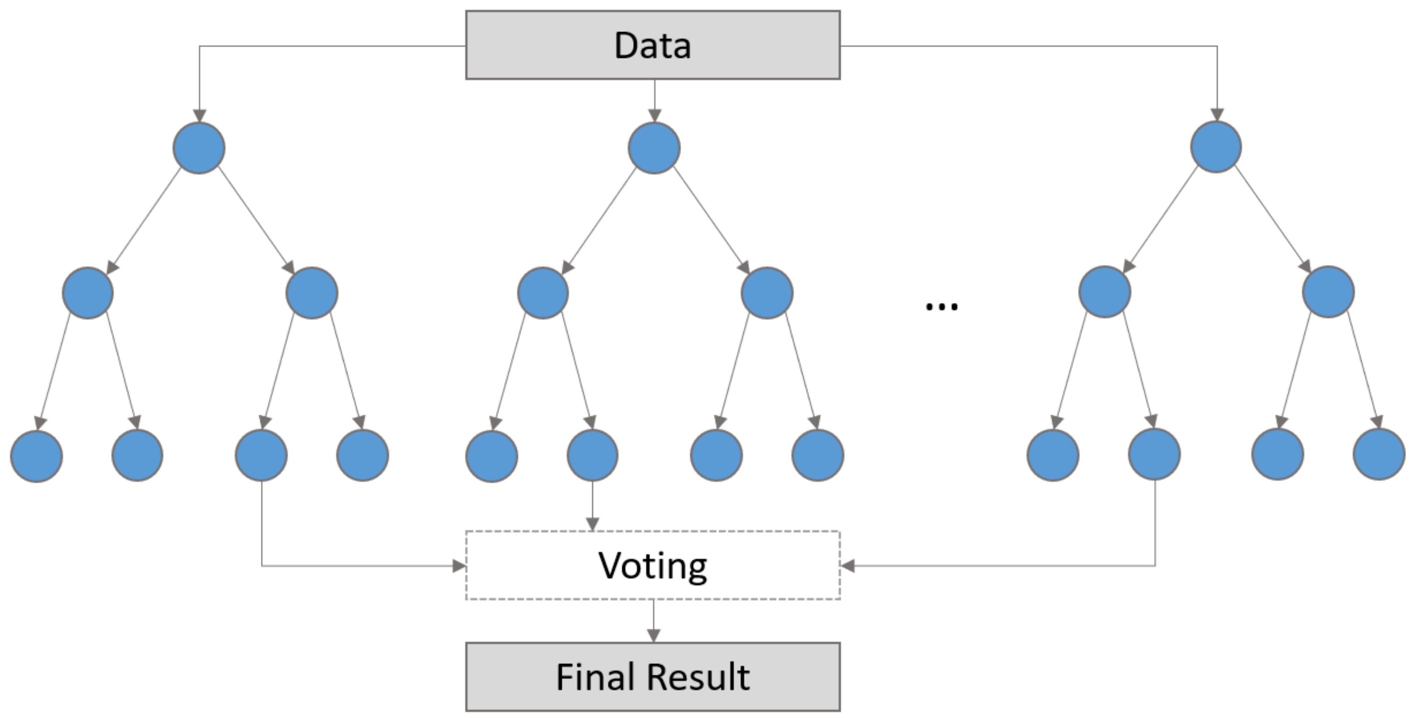
Random forest models

DIMENSIONALITY REDUCTION IN R

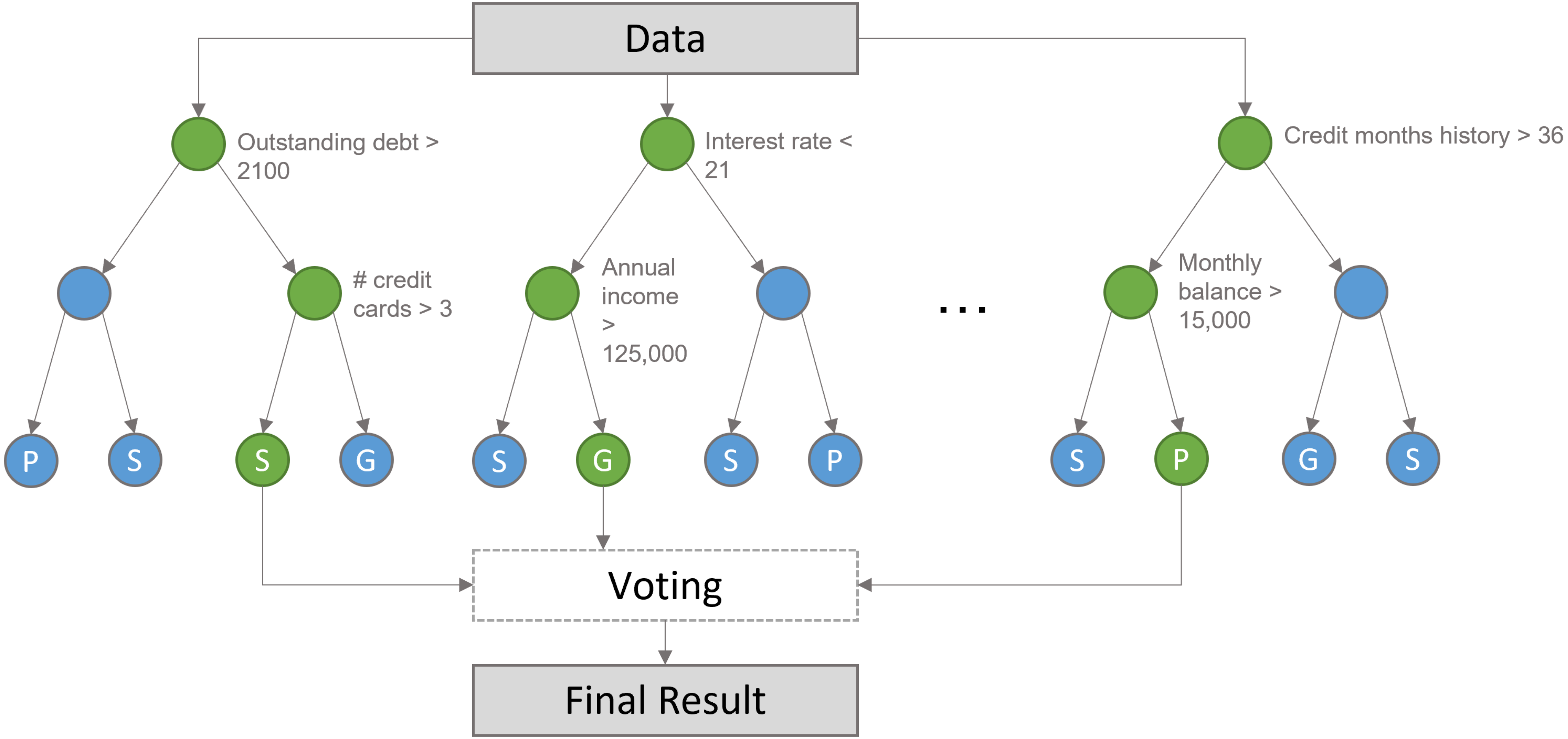
Matt Pickard
Owner, Pickard Predictives, LLC

Random Forest

- An ensemble model
  - a "wisdom of the crowds" approach
- Aggregates predictions of many random trees
- Random uncorrelated trees mitigate error
- Avoids overfitting
- Accurate
- Performs feature selection



Random Forest



Train a Random Forest

```r
library(tidymodels)

rf <- rand_forest(mode = "classification",  trees = 200) %>%
  set_engine("ranger", importance = "impurity")

rf_fit <- rf %>%
  fit(credit_score ~ ., data = train)

predict_df <- test %>%
  bind_cols(predict = predict(rf_fit, test))
```

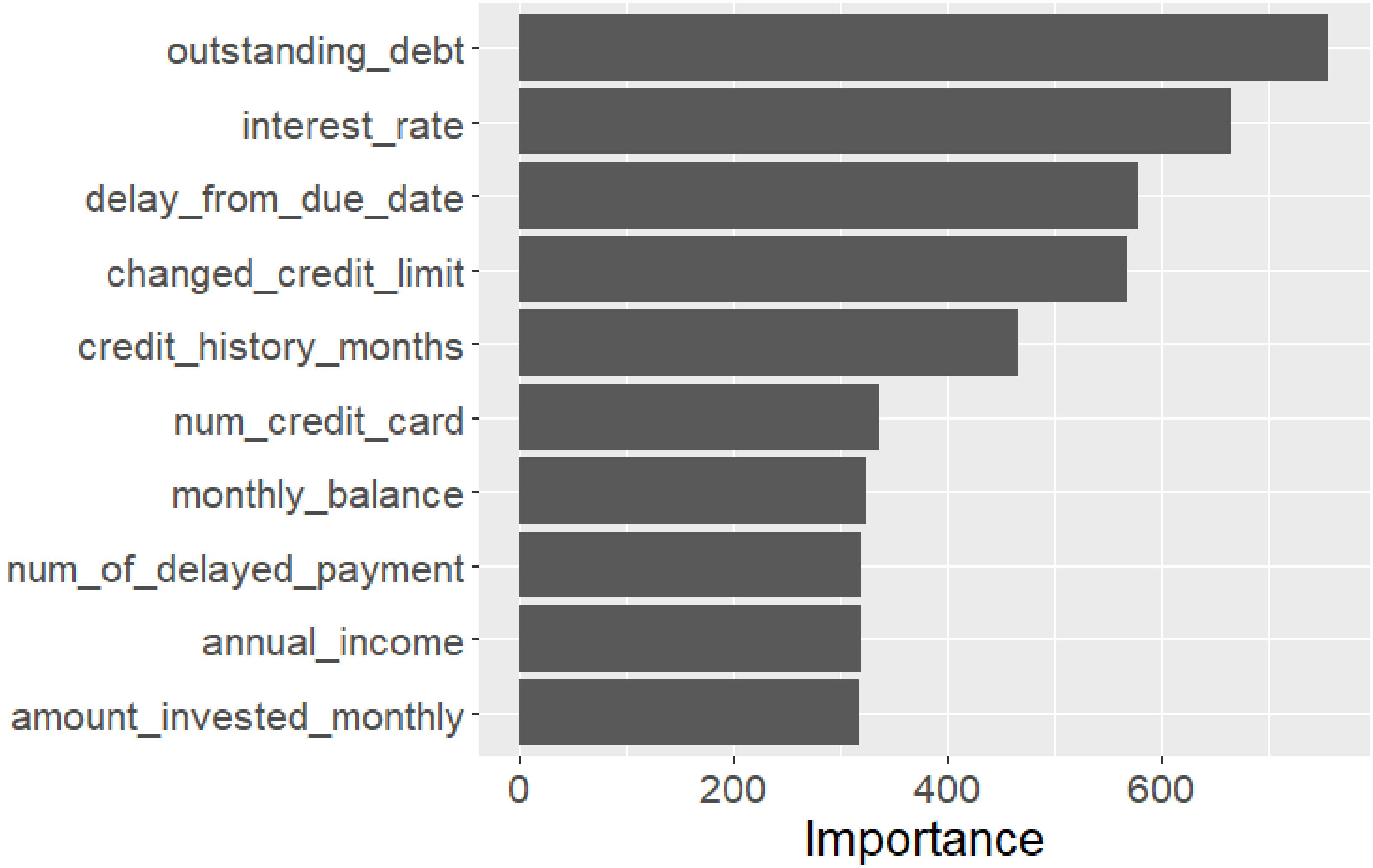
Evaluate the Model

```
f_meas(predict_df, credit_score, .pred_class)
```

```
0.6895
```

Variable Importance

```
library(vip)
rf_fit %>% vip()
```



Feature Mask

```
top_features <- rf_fit %>%
  vi(rank = TRUE) %>%
  filter(Importance <= 10) %>%
  pull(Variable)
top_features
```

```
 [1] "outstanding_debt"         "interest_rate"
 [3] "delay_from_due_date"      "changed_credit_limit"
 [5] "credit_history_months"    "num_credit_card"
 [7] "monthly_balance"          "num_of_delayed_payment"
 [9] "annual_income"            "amount_invested_monthly"
```

Reduce the data

```
train_reduced <- train[top_features]
test_reduced <- test[top_features]
```

Performance

```
rf_fit <- rf %>%
  fit(credit_score ~ ., data = train_reduced)
predict_reduced_df <- test_reduced %>%
  bind_cols(predict = predict(rf_fit, test_reduced))
f_meas(predict_reduced_df, credit_score, .pred_class)
```

```
0.6738
```

F-score of the unreduced model:

```
0.6895
```

Let's practice!

DIMENSIONALITY REDUCTION IN R