random forest

2025-03-30

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
df <- read.csv("/Users/joyceli/Desktop/consolidated_df_1.csv")

df <- df |>
  mutate(Ethnicity_Match = as.factor(Ethnicity_Match),
         log_Match.Length = log(Match.Length+1))

df_rf_recipe <- recipe(log_Match.Length ~ Big.Gender + Ethnicity_Match + Big.Level.of.Education +
    age_gap_abs + Program.Type + SP500_Level +
    avg_change_density + has_interests + personality_compatibility +
    has_goals + Occupation_Category + latest_sentiment_net +
    avg_days_between_logs + topic_5_scaled + avg_engagement_words +
    avg_change_words, data = df) |>
  step_unknown(Big.Gender, SP500_Level, Occupation_Category) |>  # Handle NAs in categorical variables
  step_log(age_gap_abs, avg_change_density, avg_days_between_logs, offset = 1) |>
  step_dummy(all_factor_predictors()) |>
  step_normalize(all_numeric_predictors())

df_rf_spec <- rand_forest(trees = 100, mtry = 4) |>
  set_mode("regression") |>
  set_engine("ranger", importance = "impurity")

df_rf_workflow <-
  workflow() |>
  add_recipe(df_rf_recipe) |>
  add_model(df_rf_spec)

# Stratified 5-fold cross-validation
set.seed(1234)
folds <- vfold_cv(df, v = 10, strata = Match.Length)

reg_metrics <- metric_set(yardstick::rmse, yardstick::mae, yardstick::rsq)

rf_cv <- fit_resamples(df_rf_workflow, resamples = folds, metrics = reg_metrics)
show_best(rf_cv, metric = "rmse")
```

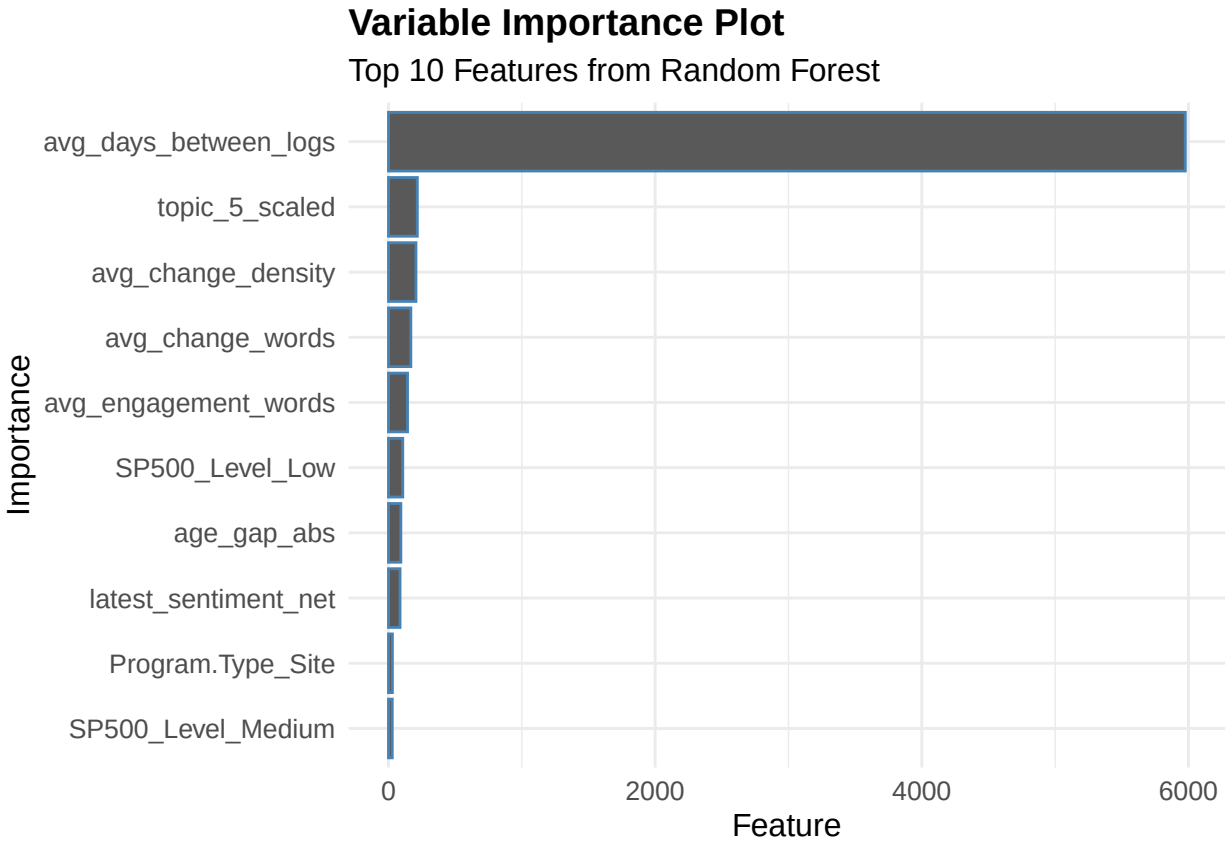
```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
```

```
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.517    10 0.00480 Preprocessor1_Model1
```

```
(rf_fit <- fit(df_rf_workflow, data = df))
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_unknown()
## * step_log()
## * step_dummy()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~4,      x), num.trees = ~100, impor
##
## Type:                             Regression
## Number of trees:                  100
## Sample size:                      3275
## Number of independent variables:  40
## Mtry:                             4
## Target node size:                 5
## Variable importance mode:         impurity
## Splitrule:                        variance
## OOB prediction error (MSE):       0.268242
## R squared (OOB):                  0.6161515
```

```
vip(rf_fit,
    num_features = 10,      # Show top 10 features
    aesthetics = list(color = "steelblue")) + # Customize colors and sizes
  theme_minimal() +         # Use a minimal theme
  labs(
    title = "Variable Importance Plot",
    subtitle = "Top 10 Features from Random Forest",
    x = "Importance",
    y = "Feature"
  ) +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    plot.subtitle = element_text(size = 12),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10)
  )
```




```
# training accuracy
rf_predictions <- augment(rf_fit, new_data = df)
rmse(rf_predictions, truth = log_Match.Length, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimator
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.338
```

Boosting

```
boost_grid <- grid_regular(trees(range = c(100, 300)), tree_depth(range = c(3, 10)), learn_rate(range =

# Data for 10-fold CV
set.seed(1234)
boost_folds <- vfold_cv(df, v = 5, strata = log_Match.Length)

df_boost_tune_spec <- boost_tree(trees = tune(), tree_depth = tune(), learn_rate = tune()) |>
  set_mode("regression") |>
  set_engine("xgboost")

df_boost_tune_workflow <-
  workflow() |>
  add_recipe(df_rf_recipe) |>
  add_model(df_boost_tune_spec)

df_boost_tune <- tune_grid(
  df_boost_tune_workflow,
```

```
  resamples = boost_folds,
  grid = boost_grid,
  metrics = reg_metrics
)

boost_best <- select_best(df_boost_tune, metric = "rmse")
df_boost_wf <- finalize_workflow(df_boost_tune_workflow, boost_best)
df_boost_fit <-  fit(df_boost_wf, data = df)

# training accuracy
df_boost_train_preds <- augment(df_boost_fit, new_data = df)
rmse(df_boost_train_preds, log_Match.Length, .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.337
```