Reducing the model's features

FEATURE ENGINEERING IN R

Jorge Zazueta
Research Professor and Head of the Modeling Group at the School of Economics, UASLP

Reasons to reduce the number of features

Eliminating irrelevant or low-information variables can have benefits, including

- Reduce model variance without significantly increasing bias
- Increase out-of-sample model performance
- Reducing computation time
- Decreasing model complexity
- Improving interpretability

Sifting data through variable importance

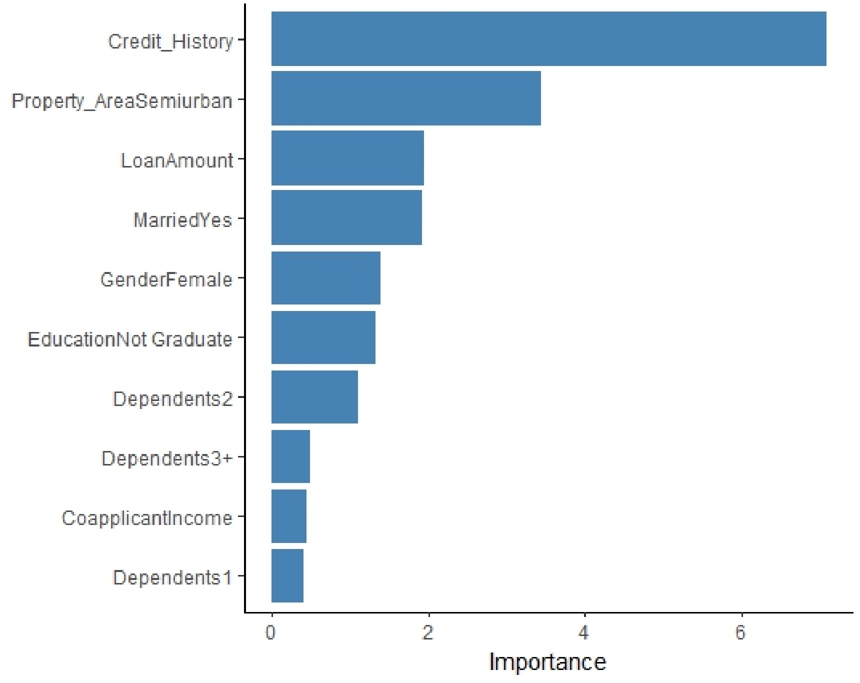
Fitting a model with all features

```
lr_recipe_full <-
  recipe(Loan_Status ~., data = train) %>%
  update_role(Loan_ID, new_role = "ID")

lr_workflow_full <-
  workflow() %>%
  add_model(lr_model) %>%
  add_recipe(lr_recipe_full)

lr_fit_full <-
  lr_workflow_full %>%
  fit(data = train)
```

Graphing variable vip

```
lr_fit_full %>%
  extract_fit_parsnip() %>%
  vip(aesthetics = list(fill = "steelblue"))
```

Variable importance



Build a reduced model using the formula syntax

We can add features directly by using the basic R formula syntax.

```
# Create recipe
recipe_formula <-
  recipe(Loan_Status ~ Credit_History + Property_Area +
           LoanAmount, data = train)

# Bundle with model
workflow_formula <- # Bundle with model
  workflow() %>% add_model(lr_model) %>%
  add_recipe(recipe_formula)
```

Build a reduced model by creating a features vector

A feature vector can be passed used to select features before training.

```
# Feature vector
features <- c("Credit_History", "Property_Area", "LoanAmount", "Loan_Status")

# Training and testing data
train_features <- train %>% select(all_of(features))
test_features <- test %>% select(all_of(features))

# Create recipe and bundle with model
recipe_features <- recipe(Loan_Status ~., data = train_features)
workflow_features <- workflow() %>% add_model(lr_model) %>%
  add_recipe(recipe_features)
```

Creating the augmented objects

Augmented objects for both approaches

```
lr_aug_formula <-
  workflow_formula %>%
  fit(data = train) %>%
  augment(new_data = test)
```

```
lr_aug_features <-
  workflow_features %>%
  fit(data = train_features) %>%
  augment(new_data = test_features)
```

Both ways return the same results

```
all_equal(lr_aug_features,
lr_aug_formula %>%
select(all_of(features),
starts_with(".pred")))
```

```
[1] TRUE
```

Comparing the full and reduced models

Using all features

```
lr_fit_full <- # Fit workflow
  lr_workflow_full %>%
  fit(data = train)
lr_aug_full <- # Augment
  lr_fit_full %>%
  augment(test)
lr_aug_full %>% # Evaluate
  class_evaluate(truth = Loan_Status,
                 estimate = .pred_class,
                 .pred_Y)
```

```
# A tibble: 2 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.842
2 roc_auc  binary         0.744
```

Using top 3 features*

```
lr_fit_formula <- # Fit workflow
  workflow_formula %>%
  fit(train)
lr_aug_formula <- # Augment
  lr_fit_formula %>%
  augment(new_data = test)
lr_aug_formula %>% # Evaluate
  class_evaluate(truth = Loan_Status,
                 estimate = .pred_class,
                 .pred_Y)
```

```
# A tibble: 2 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.842
2 roc_auc  binary         0.733
```

Let's practice!

FEATURE ENGINEERING IN R

Shrinkage methods

FEATURE ENGINEERING IN R

Jorge Zazueta
Research Professor and Head of the Modeling Group at the School of Economics, UASLP

Two common regularization techniques

Lasso

- Adds penalty term proportional to absolute value of model weights
- Encourages some weights to become exactly zero
- Effectively eliminates the corresponding features
- Can be an automated feature selection method

Ridge

- Adds penalty term proportional to square of model weights
- Does not shrink some coefficients to zero like Lasso
- But can effectively reduce overfitting

A first look at Lasso

Set up the model

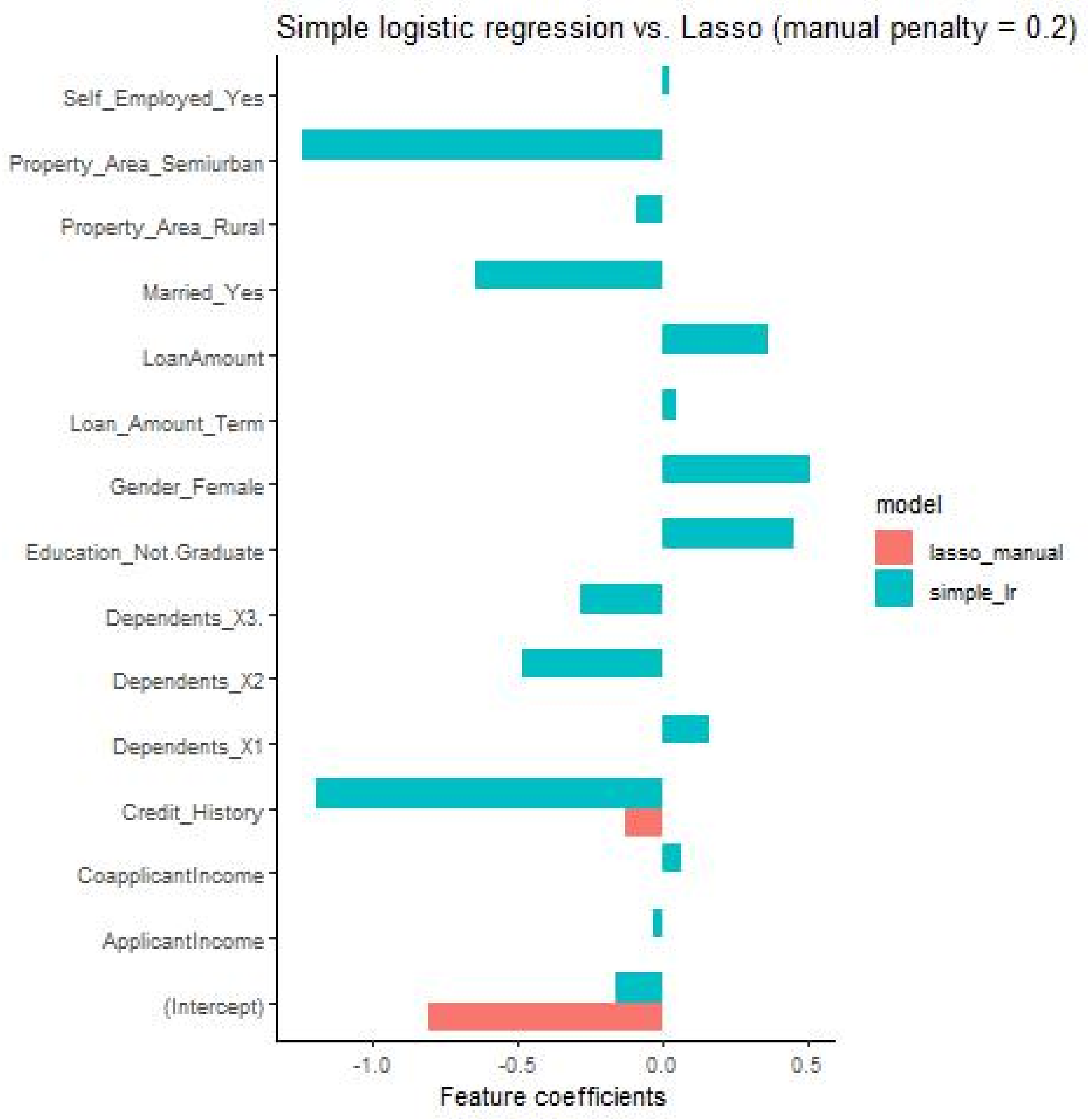
```
recipe <- # Define recipe
recipe(Loan_Status ~ ., data = train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  update_role(Loan_ID, new_role = "ID")
# set up model
model_lasso_manual <- logistic_reg() %>%
  set_engine("glmnet") %>%
  set_args(mixture = 1, penalty = .2)
# Bundle in workflow
workflow_lasso_manual <-
  workflow() %>%
  add_model(model_lasso_manual) %>%
  add_recipe(recipe)
```

Fit and inspect

```
fit_lasso_manual <- # Fit workflow
  workflow_lasso_manual %>%
  fit(train)
#Inspect coefficients
tidy(fit_lasso_manual)
```

```
# A tibble: 15 × 3
   term                       estimate penalty
   <chr>                         <dbl>   <dbl>
 1 (Intercept)                  -0.816     0.2
 2 ApplicantIncome               0         0.2
 3 CoapplicantIncome             0         0.2
 4 LoanAmount                    0         0.2
 5 Loan_Amount_Term              0         0.2
 6 Credit_History               -0.220     0.2
 7 Gender_Female                 0         0.2
 ...                             ...       ...
```

Simple logistic regression vs. Lasso



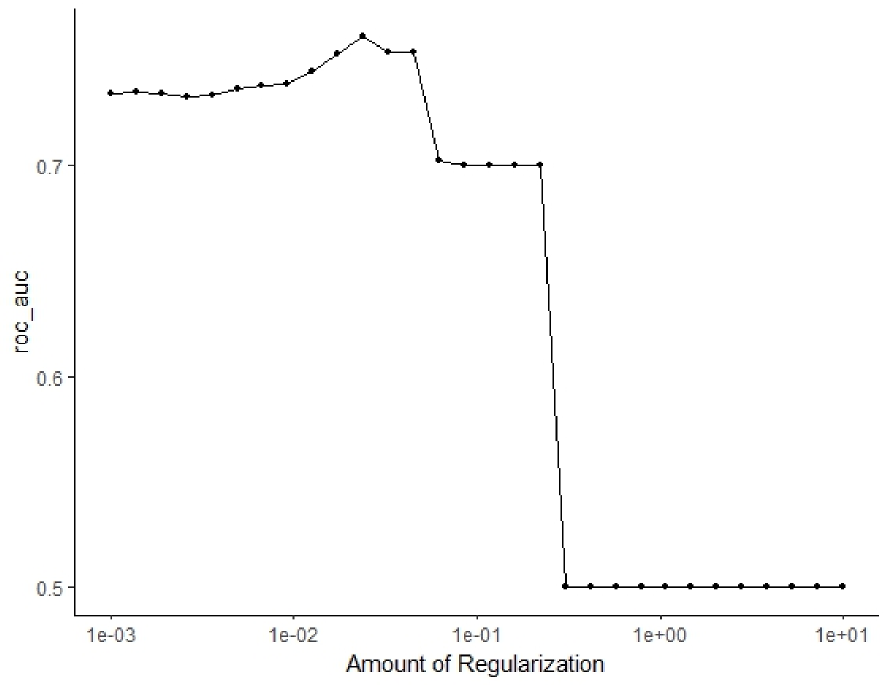
Hyperparameter tuning

Setting a model with tuning

```
model_lasso_tuned <- logistic_reg() %>%
  set_engine("glmnet") %>%
  set_args(mixture = 1,
  penalty = tune())

workflow_lasso_tuned <-
  workflow() %>%
  add_model(model_lasso_tuned) %>%
  add_recipe(recipe)

penalty_grid <- grid_regular(
  penalty(range = c(-3, 1)),
  levels = 30)
```

Looking at the tuning output

```
tune_output <- tune_grid(
  workflow_lasso_tuned,
  resamples = vfold_cv(train, v = 5),
  metrics = metric_set(roc_auc),
  grid = penalty_grid)
autoplot(tune_output)
```



Exploring the results

Auto-chosen features

```
best_penalty <-
select_by_one_std_err(tune_output,
metric = 'roc_auc', desc(penalty))

# Fit Final Model
final_fit<-
finalize_workflow(workflow_lasso_tuned,
best_penalty) %>%
  fit(data = train)
```

```
final_fit_se %>% tidy()
```

```
# A tibble: 15 × 3
   term                     estimate penalty
   <chr>                       <dbl>   <dbl>
 1 (Intercept)                -0.660  0.0452
 2 ApplicantIncome             0      0.0452
 3 CoapplicantIncome           0      0.0452
 4 LoanAmount                  0      0.0452
 5 Loan_Amount_Term            0      0.0452
 6 Credit_History             -0.948  0.0452
 7 Gender_Female               0      0.0452
 8 Married_Yes                -0.191  0.0452
 9 Dependents_X1               0      0.0452
10 Dependents_X2               0      0.0452
11 Dependents_X3.              0      0.0452
12 Education_Not.Graduate      0      0.0452
13 Self_Employed_Yes           0      0.0452
14 Property_Area_Rural         0      0.0452
15 Property_Area_Semiurban    -0.163  0.0452
```

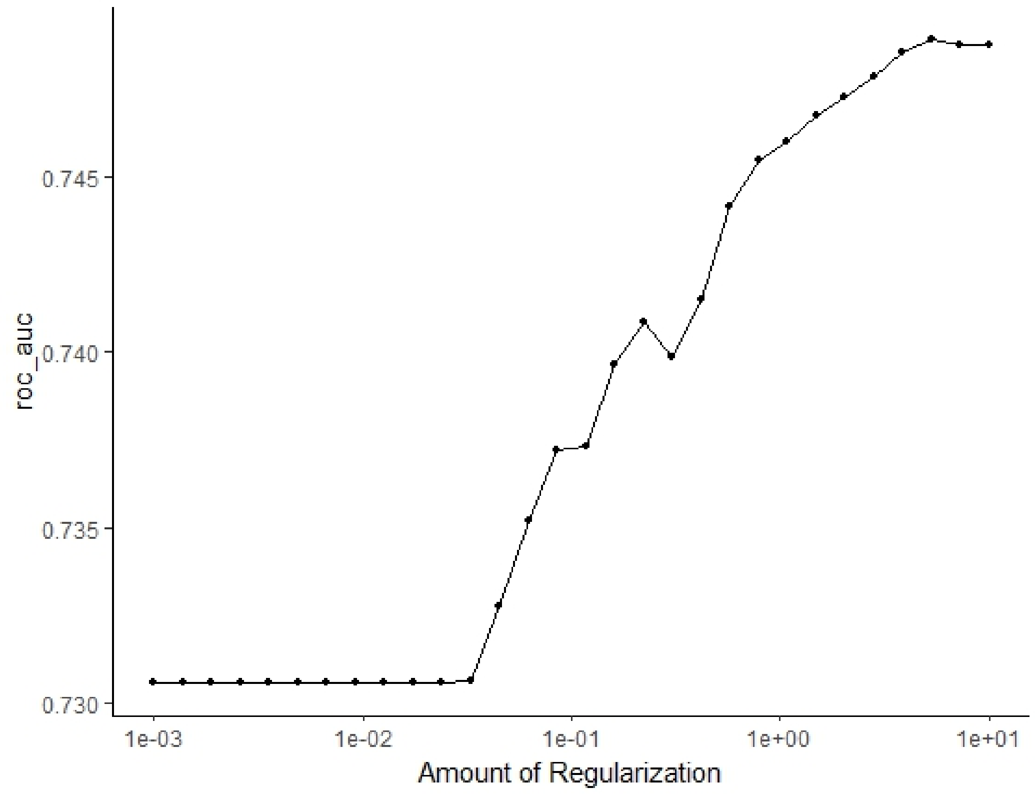
Simple logistic regression vs. tuned Lasso

Ridge regularization

Ridge is the option when mixture = 0

```
model_ridge_tuned <- logistic_reg() %>%
  set_engine("glmnet") %>%
  set_args(mixture = 0, penalty = tune())

workflow_ridge_tuned <-
  workflow() %>%
  add_model(model_ridge_tuned) %>%
  add_recipe(recipe)

tune_output <- tune_grid(
  workflow_ridge_tuned,
  resamples = vfold_cv(train, v = 5),
  metrics = metric_set(roc_auc),
  grid = penalty_grid)
```

```
tune_output <- tune_grid(
  workflow_ridge_tuned,
  resamples = vfold_cv(train, v = 5),
  metrics = metric_set(roc_auc),
  grid = penalty_grid)
 autoplot(tune_output)
```



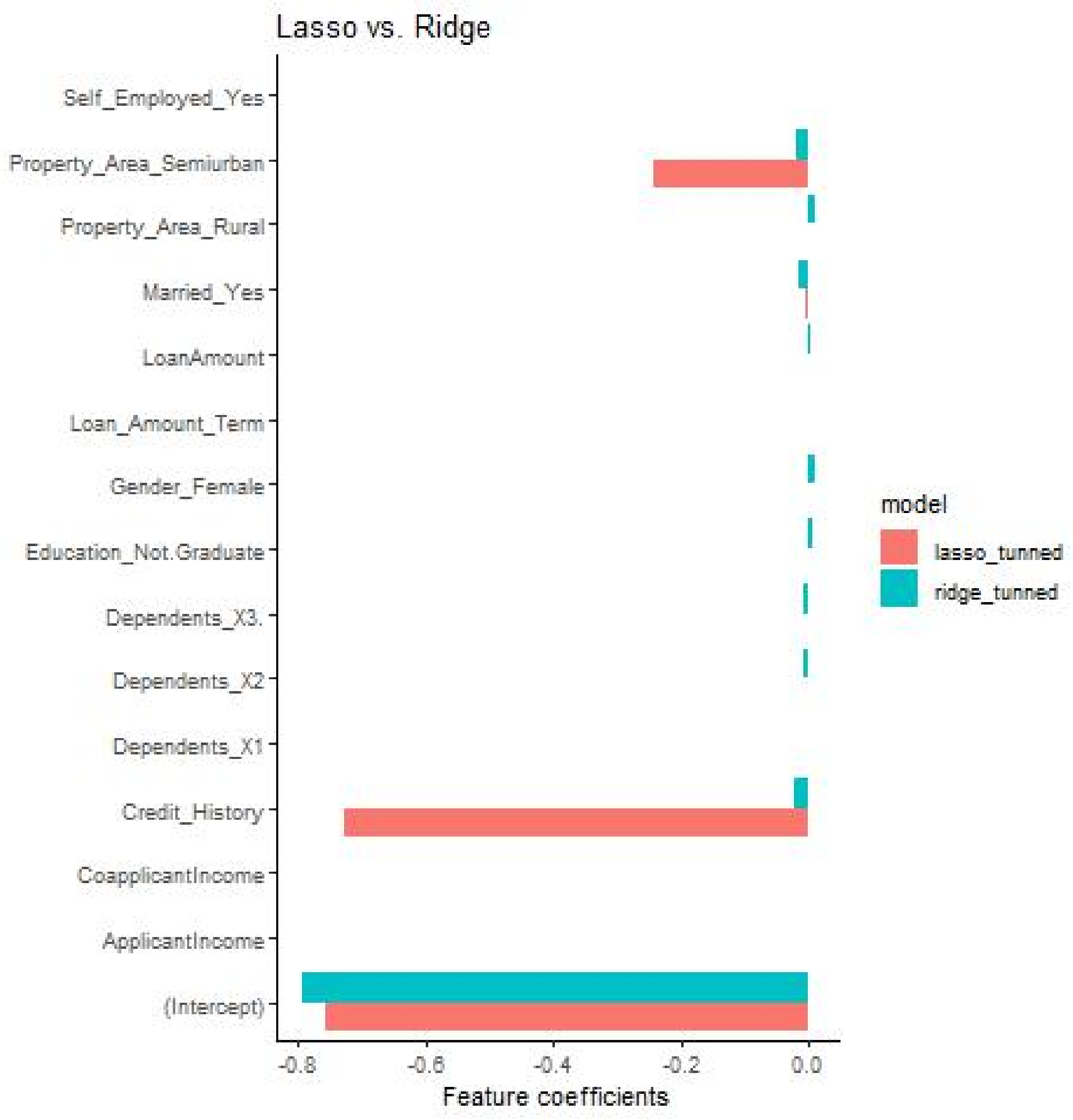
Ridge regularization

```
best_penalty <-
select_by_one_std_err(tune_output,
metric = 'roc_auc', desc(penalty))
best_penalty

final_fit<-
finalize_workflow(workflow_ridge_tuned,
best_penalty) %>%
  fit(data = train)
```

```
tidy(final_fit)
```

```
# A tibble: 15 × 3
   term                    estimate penalty
   <chr>                      <dbl>   <dbl>
 1 (Intercept)             -0.799        10
 2 ApplicantIncome          0.00232      10
 3 CoapplicantIncome        0.0000537    10
 4 LoanAmount               0.00291      10
 5 Loan_Amount_Term         0.00161      10
 6 Credit_History          -0.0245       10
 7 Gender_Female            0.00850      10
 8 Married_Yes             -0.0140       10
 9 Dependents_X1            0.00497      10
10 Dependents_X2           -0.0100       10
11 Dependents_X3.           0.00259      10
12 Education_Not.Graduate   0.00308      10
13 Self_Employed_Yes        0.00892      10
14 Property_Area_Rural      0.0109       10
```

Ridge vs. Lasso



Let's practice!

FEATURE ENGINEERING IN R

Putting it all together

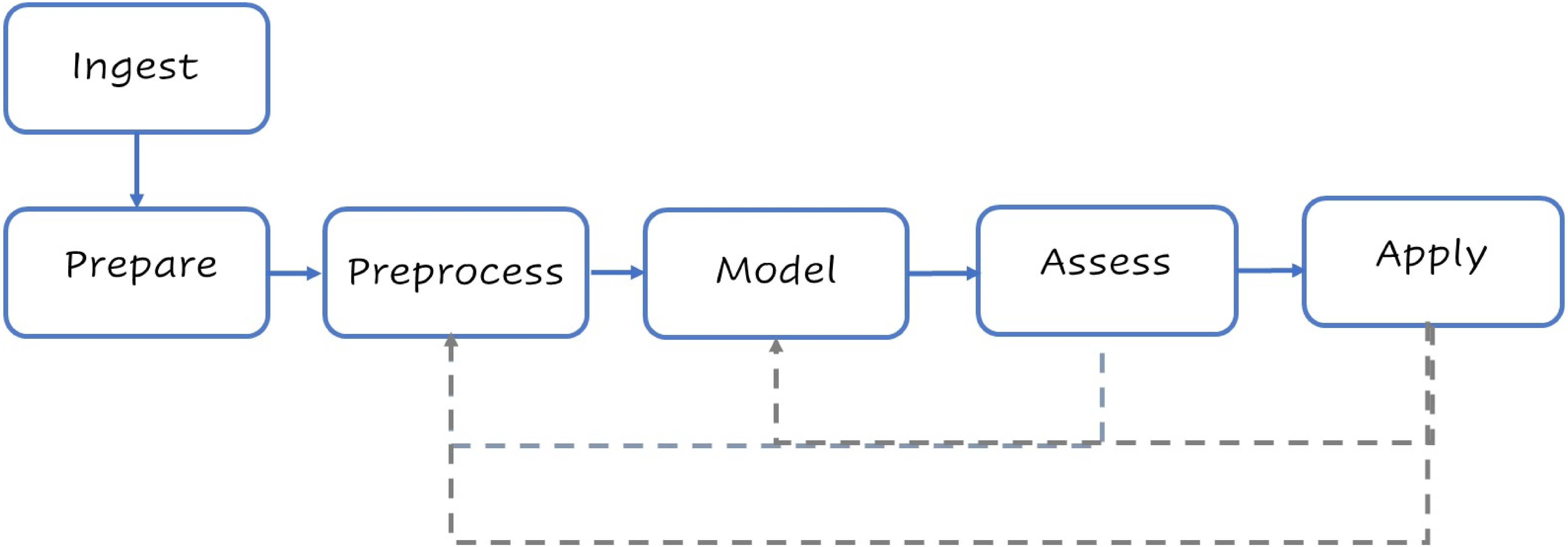
FEATURE ENGINEERING IN R

Jorge Zazueta

Research Professor. Head of the Modeling Group at the School of Economics, UASLP

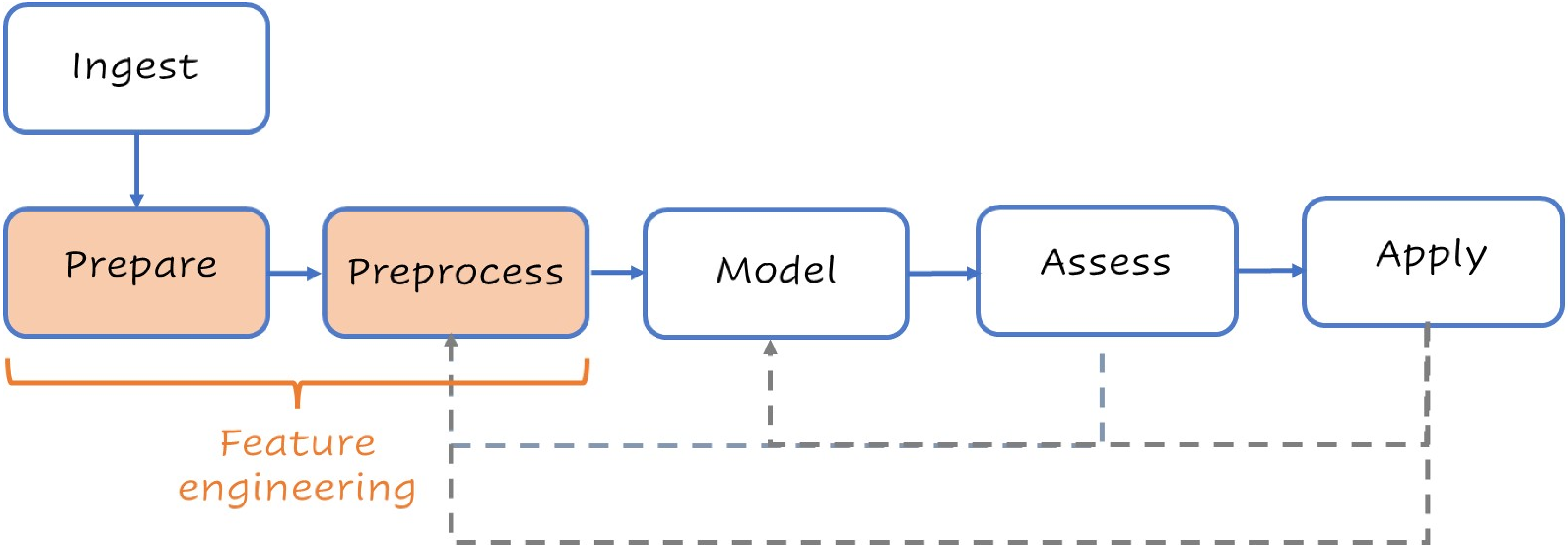
A stylized process modeling flow

Typical high-level modeling steps.



A stylized process modeling flow

Typical high-level modeling steps.



Prepare

Start by doing some basic housekeeping and setting up our splits.

```
loans <- # Basic housekeeping
  loans %>%
  mutate(across(where(is_character),
                as_factor)) %>%
  mutate(across(Credit_History,
                as_factor))

set.seed(123) # Set up splits
split <- initial_split(loans,
        strata = Loan_Status)
test <- testing(split)
train <- training(split)
```

```
glimpse(train)
```

```
Rows: 460
Columns: 13
$ Loan_ID           <fct> LP001003...
$ Gender            <fct> Male, Ma...
$ Married           <fct> Yes, No,...
$ Dependents        <fct> 1, 0, 0,...
$ Education         <fct> Graduate...
$ Self_Employed     <fct> No, No, ...
$ ApplicantIncome   <dbl> 4583, 18...
$ CoapplicantIncome <dbl> 1508, 28...
$ LoanAmount        <dbl> 128, 114...
$ Loan_Amount_Term  <dbl> 360, 360...
$ Credit_History    <fct> 1, 1, 0,...
$ Property_Area     <fct> Rural, R...
$ Loan_Status       <fct> N, N, N,...
```

Preprocess

Our recipe can be quite short or very complex.

```
recipe <- recipe(Loan_Status ~ .,
data = train) %>%
  update_role(Loan_ID,
  new_role = "ID") %>%
  step_normalize(all_numeric_predictors()) %>%
  step_impute_knn(all_predictors()) %>%
  step_dummy(all_nominal_predictors())
```

```
recipe
```

```
Recipe

Inputs:

      role #variables
        ID          1
   outcome          1
 predictor         11

Operations:

Centering and scaling for all_numeric_predictors()
K-nearest neighbor imputation for all_predictors()
Dummy variables from all_nominal_predictors()
```

Model

Set up workflow

```
lr_model <- logistic_reg() %>%
  set_engine("glmnet") %>%
  set_args(mixture = 1, penalty = tune())

lr_penalty_grid <- grid_regular(
  penalty(range = c(-3, 1)),
  levels = 30)

lr_workflow <-
  workflow() %>%
  add_model(lr_model) %>%
  add_recipe(recipe)
```

```
lr_workflow
```

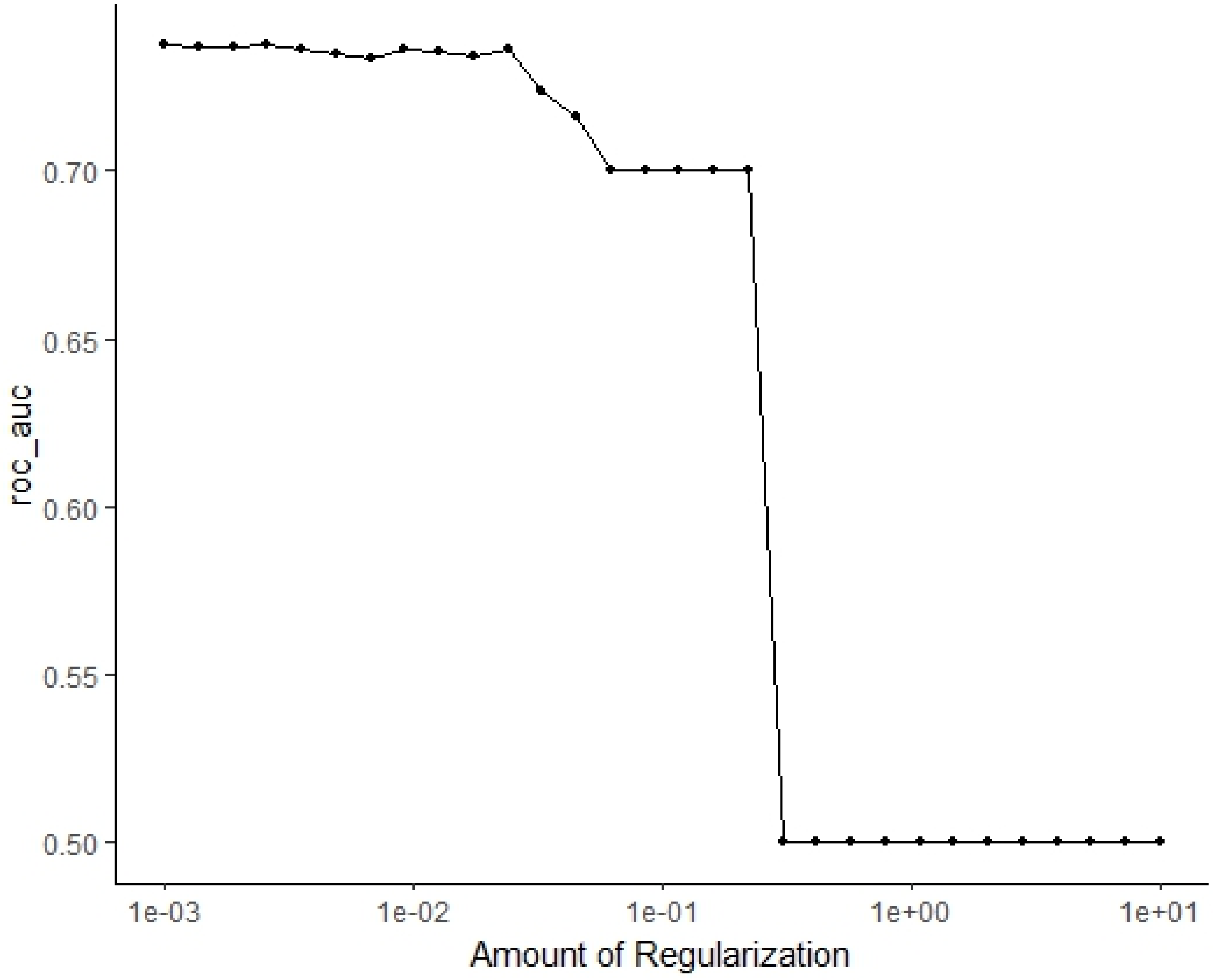
```
--Workflow --------------------------------
Preprocessor: Recipe
Model: logistic_reg()

-- Preprocessor ---------------------------
3 Recipe Steps
- step_normalize()
- step_impute_knn()
- step_dummy()

-- Model ----------------------------------
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = 1
Computational engine: glmnet
```

Assess

Tune penalty for Lasso

```
lr_tune_output <- tune_grid(
  lr_workflow,
  resamples = vfold_cv(train, v = 5),
  metrics = metric_set(roc_auc),
  grid = penalty_grid)

autoplot(tune_output)
```

ROC_AUC vs. Regularization



Assess

Fitting the final model

```
best_penalty <-
select_by_one_std_err(lr_tune_output,
metric = 'roc_auc', desc(penalty))

lr_final_fit<-
finalize_workflow(lr_workflow, best_penalty) %>%
  fit(data = train)

lr_final_fit %>%
  augment(test) %>%
  class_evaluate(truth = Loan_Status,
                 estimate = .pred_class,
                            .pred_Y)
```

Our performance metrics

```
# A tibble: 2 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.818
2 roc_auc  binary         0.813
```

Let's practice!

FEATURE ENGINEERING IN R

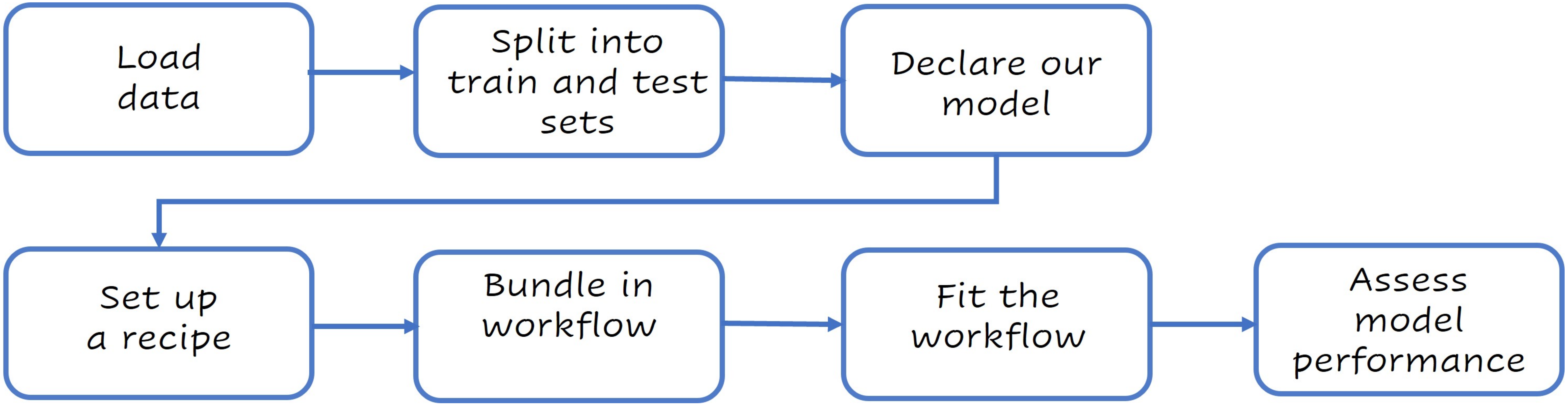
Congratulations!

FEATURE ENGINEERING IN R

Jorge Zazueta
Research Professor. Head of the Modeling Group at the School of Economics, UASLP

You've gone a great distance.

From zero to hero in four lessons!



Where to go from here?

Data science is a never ending journey that keeps refreshing itself. These are some datacamp courses that you might considering as next steps.

- Dimensionality reduction in R
- Advanced dimensionality reduction in R
- Modeling with tidymodels in R

Go get them all!

FEATURE ENGINEERING IN R