vignette

```r
source("functions.R")
source("make_tidy.R")
```

Preparing Data

We generate a data set called 'dat' fist. T hen w e s plit t he d ata s et i nto a t raining s et a nd a t esting set. We build a recipe by using recipe() function. Also, we use step_dummy() creates a secification o f a recipe step that will convert nominal data into one numeric binary model terms for the levels of the original data. We remove the variables that contain only a single value by step_zv() and normalize numeric data to have a standard deviation of one and a mean of zero by step_normalize().

```r
set.seed(123)
n = 1000
dat <- tibble(x = seq(-3,3, length.out = n),
              w = 3*cos(3*seq(-pi,pi, length.out = n)),
              y = rbinom(n,size = 1, prob = 1/(1 + exp(-w+2*x)) )%>% as.numeric %>% factor,
              cat = sample(c("a","b","c"), n, replace = TRUE)
              )

split <- initial_split(dat, strata = c("cat"))

train <- training(split)
test <- testing(split)

rec <- recipe(y ~ . , data = train) %>%
  step_dummy(all_nominal(), -y) %>% step_zv(all_outcomes()) %>%
  step_normalize(all_numeric(), -y)
```

Fit model

We fit logistic Lasso model by using 'fit_logistic_lasso' function. We noticed that with $\lambda = 0.1$, the value of β will converge after 18 iterations. The value of β are x=-3.60651540,w=2.09516310,cat_b =-0.10289392 and cat_c=-0.07120818 with intercept = -0.03768201.

```r
lambda = 0.01
spec <- IRLS(penalty=lambda) %>% set_engine("fit_logistic_lasso")

fit <- workflow() %>% add_recipe(rec) %>% add_model(spec) %>% fit(train)

predict(fit, new_data = test) %>% bind_cols(test %>% select(y)) %>%
  conf_mat(truth = y, estimate = .pred_class)
```

```
##             Truth
## Prediction   0   1
##          0 115  18
##          1  11 105
```

```
fit$fit
```

```
## $actions
## $actions$model
## $spec
## Model Specification (classification)
##
## Main Arguments:
##   penalty = lambda
##
## Computational engine: fit_logistic_lasso
##
##
## $formula
## NULL
##
## attr(,"class")
## [1] "action_model" "action_fit"   "action"
##
##
## $fit
## parsnip model object
##
## Fit time:  50ms
## $beta
##           x           w       cat_b       cat_c
## -3.60651540  2.09516310 -0.10289392 -0.07120818
##
## $intercept
##
## -0.03768201
##
## $lambda
## [1] 0.01
##
## $fct_levels
## [1] "0" "1"
##
## $iter
## [1] 18
##
## $converged
## [1] TRUE
##
##
## attr(,"class")
## [1] "stage_fit" "stage"
```

Compare with logistic regresstion and glm model

check that we if got the answers correct by comparing with glm. We find that the glm estimate result is similar to IRLS estimate result which is not bad. The test return is 'False' which indicates that there exist some differences between the prediction results of the logistic Lasso model and the results of the glm model.

```
# Make the data
ddat<- rec %>% prep(train) %>% juice

# fit the glm reg
model1 <- glm(y ~ -1+., family = "binomial", data = ddat)
model1 %>% tidy %>% select(term, estimate) %>%
  mutate(IRLS_estimate = fit$fit$fit$fit$beta, err = estimate - IRLS_estimate)
```

```
## # A tibble: 4 x 4
##   term   estimate IRLS_estimate       err
##   <chr>    <dbl>         <dbl>     <dbl>
## 1 x      -3.61          -3.61   -0.00578
## 2 w       2.10           2.10    0.00513
## 3 cat_b  -0.103         -0.103   0.000354
## 4 cat_c  -0.0694        -0.0712  0.00186
```

```
# Make the test data
test_dat <- rec %>% prep(train) %>% bake(test)
glm_pred <- (predict(model1, test_dat, type = "response") > 0.5) %>% as.numeric
preds <- predict(fit, new_data=test) %>% bind_cols(glm_pred = glm_pred)
any(preds$.pred_class != preds$glm_pred)
```

```
## [1] FALSE
```

Tune model

I try my best, but I still cannot figure it out. SORRY:(

```
grid <- grid_regular(penalty(), levels = 10)
spec_tune <- logistic_lasso(penalty=tune()) %>% set_engine("fit_logistic_lasso")

wf <- workflow() %>% add_recipe(rec) %>% add_model(spec_tune)
folds <- vfold_cv(train)
fit_tune <- wf %>%
  tune_grid(resamples = folds, grid = grid)
fit_tune %>% collect_metrics() %>% ggplot(aes(penalty, mean)) + geom_line() +
  facet_wrap(~.metric)
```