Feature Engineering Methods - Model Performance

Merve Topdemir

08 04 2021

This study aims to examine the relationship between males at high risk of heart disease and characteristics such as age, obesity, and alcohol consumption using logistic regression. The data to be used in this study is the "heart" from the "catdata" package. In addition, the performance of the model will be tried to be increased by using data "Feature Engineering" methods.

```
#install.packages("tidymodels")
library(tidymodels)

#install.packages("catdata")
library(catdata)

data("heart")
data <- heart
head(data,5)
```

```
##   y  sbp   tobacco  ldl   adiposity famhist typea obesity   alcohol age
## 1 1  160    12.00   5.73   23.11       1     49    25.30    97.20   52
## 2 1  144    0.01    4.41   28.61       0     55    28.87    2.06    63
## 3 0  118    0.08    3.48   32.28       1     52    29.14    3.81    46
## 4 1  170    7.50    6.41   38.03       1     51    31.99    24.26   58
## 5 1  134    13.60   3.50   27.78       1     60    25.99    57.34   49
```

Data Structure

The data set contains 462 observations and 10 variables.

Dependent (target) variable:

- **y** : Heart disease (yes = 1, no = 0)

Independent (feature) variables:

- **Sbp** : Systolic blood pressure
- **Tobacco** : Cumulative tobacco
- **Ldl** : Low density lipoprotein cholesterol
- **Adiposity**
- **Famhist :** Family history of heart disease
- **Typea :** Type-A behavior
- **Obesity**
- **Alcohol :** Alcohol consumption
- **Age**

```
str(data)
```

```
## 'data.frame':    462 obs. of  10 variables:
##  $ y        : Factor w/ 2 levels "0","1": 2 2 1 2 2 1 1 2 1 2 ...
##  $ sbp      : num  160 144 118 170 134 132 142 114 114 132 ...
##  $ tobacco  : num  12 0.01 0.08 7.5 13.6 6.2 4.05 4.08 0 0 ...
##  $ ldl      : num  5.73 4.41 3.48 6.41 3.5 6.47 3.38 4.59 3.83 5.8 ...
##  $ adiposity: num  23.1 28.6 32.3 38 27.8 ...
##  $ famhist  : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 1 2 2 2 ...
##  $ typea    : num  49 55 52 51 60 62 59 62 49 69 ...
##  $ obesity  : num  25.3 28.9 29.1 32 26 ...
##  $ alcohol  : num  97.2 2.06 3.81 24.26 57.34 ...
##  $ age      : num  52 63 46 58 49 45 38 58 29 53 ...
```

```
summary(data)
```

```
##    y          sbp             tobacco            ldl          adiposity
##  0:302   Min.   :101.0    Min.   : 0.0000   Min.   : 0.980   Min.   : 6.74
##  1:160   1st Qu.:124.0    1st Qu.: 0.0525   1st Qu.: 3.283   1st Qu.:19.77
##          Median :134.0    Median : 2.0000   Median : 4.340   Median :26.11
##          Mean   :138.3    Mean   : 3.6356   Mean   : 4.740   Mean   :25.41
##          3rd Qu.:148.0    3rd Qu.: 5.5000   3rd Qu.: 5.790   3rd Qu.:31.23
##          Max.   :218.0    Max.   :31.2000   Max.   :15.330   Max.   :42.49
```

```
##  famhist      typea           obesity          alcohol           age
##   0:270    Min.   :13.0    Min.   :14.70    Min.   :  0.00   Min.   :15.00
##   1:192    1st Qu.:47.0    1st Qu.:22.98    1st Qu.:  0.51   1st Qu.:31.00
##            Median :53.0    Median :25.80    Median :  7.51   Median :45.00
##            Mean   :53.1    Mean   :26.04    Mean   : 17.04   Mean   :42.82
##            3rd Qu.:60.0    3rd Qu.:28.50    3rd Qu.: 23.89   3rd Qu.:55.00
##            Max.   :78.0    Max.   :46.58    Max.   :147.19   Max.   :64.00
```

Factors are used to represent categorical data. Numerical variable is one that may take on any value within a finite or infinite interval.

- y is a factor variable, it has 2 levels, 160 yes(1) and 302 no(0).
- Famhist is a factor variable, it has 2 levels, 192 (1) and 270 (0).
- Sbp variable minimum value is 101, maximum value is 218, median is 134, mean is 138.3, first quartile is 124 and third quartile is 148.
- Tobacco minimum value is 0, maximum value is 31.2, median is 2, mean is 3.6356, first quartile is 0.0525 and third quartile is 5.5.
- Age variable minimum value is 15, maximum value is 64, median is 45, mean is 42.82, first quartile is 31 and third quartile is 55.

- Obesity minimum value is 14.70, maximum value is 46.58, median is 25.80, mean is 26.04, first quartile is 22.98 and third quartile is 28.50.
- All other variables are numerical variables and their descriptive statistics are given in the output above.

Splitting Data

The following code splits 80% of the data selected randomly into training set and the remaining 20% sample into test set.

```
#install.packages("dplyr")
library(dplyr)

set.seed(123) # for reproducebility


# data splitting
split <- initial_split(data, prop = 0.80, strata = y)

# creating train set
train <- split %>%
training()

# creating test set
test <- split %>%
testing()

table(train$y)
##   0     1
##   242   129

table(test$y)
##   0   1
##   60   31
```

The number of "no(0)" observations is greater than the number of "yes(1)" observations. In this case, there may be an imbalance problem.

Train a LRM on Train Set

```
logistic_model <- logistic_reg() %>%
set_engine('glm') %>% set_mode('classification')

# Fitting a linear regression model
logistic_fit <- logistic_model %>%
fit(y ~ ., data = train)

# Obtaining the estimated parameters
tidy(logistic_fit)
```

```
## # A tibble: 10 x 5
##    term         estimate  std.error  statistic  p.value
##    <chr>          <dbl>     <dbl>      <dbl>     <dbl>
##  1 (Intercept)  -6.83       1.50      -4.55      0.00000547
##  2 sbp           0.00592    0.00651    0.909     0.364
##  3 tobacco       0.0931     0.0309     3.01      0.00263
##  4 ldl           0.157      0.0661     2.37      0.0177
##  5 adiposity     0.00532    0.0323     0.165     0.869
##  6 famhist1      0.840      0.259      3.25      0.00115
##  7 typea         0.0437     0.0144     3.02      0.00251
##  8 obesity      -0.0345     0.0483    -0.714     0.475
##  9 alcohol      -0.00553    0.00496   -1.12      0.265
## 10 age           0.0530     0.0137     3.88      0.000105
```

→ One-unit increase in the variable **sbp** is increases the target variable 0.00592 times of that unit.

→ One-unit increase in the variable **tobacco** is increases the target variable 0.0931 times of that unit.

→ One-unit increase in the variable **lbl** is increases the target variable 0.157 times of that unit.

→ One-unit increase in the variable **adiposity** is increases the target variable 0.00532 times of that unit.

→ One-unit increase in the variable **famhist1** is increases the target variable 0.840 times of that unit.

→ One-unit increase in the variable **typea** is increases the target variable 0.0437 times of that unit.

→ One-unit increase in the variable **obesity** is decreases the target variable 0.0345 times of that unit.

→ One-unit increase in the variable **alcohol** is decreases the target variable 0.00553 times of that unit.

→ One-unit increase in the variable **age** is increases the target variable 0.0137 times of that unit.

Hypothesis

$H_0 = \beta_j = 0$

$H_1 = \beta_j \neq 0$

P-value less than 0.05 is statistically significant. Therefore, we can reject the null hypothesis.

P-value higher than 0.05 is not statistically significant. Therefore, we cannot reject the null hypothesis.

- Tobacco(0.00263), ldl(0.0177), famhist1(0.00115), typea(0.00251) and age(0.00105) variables are statistically significant. If the variables are statistically significant, it means they have an effect on the target variable and H_0 rejected.

- Sbp (0.364), adiposity(0.869), obesity(0.475) and alcohol(0.265) variables are not statistically significant. If the variables are not statistically significant, it means they have no effect on the target variable and H_0 not rejected.

Making Predictions (Train)

```
# Estimated labels
labels_train <- logistic_fit %>%
predict(new_data = train , type = 'class')
head(labels_train,5)
```

```
## # A tibble: 5 x 1
##   .pred_class
##   <fct>
## 1  1
## 2  0
## 3  0
## 4  1
## 5  1
```

```
# Estimated probabilities
preds_train <- logistic_fit %>%
predict(new_data = train, type = 'prob')
head(preds_train, 5)
```

```
## # A tibble: 5 x 2
##    .pred_0  .pred_1
##     <dbl>    <dbl>
## 1   0.359    0.641
## 2   0.599    0.401
## 3   0.708    0.292
## 4   0.273    0.727
## 5   0.317    0.683
```

```
result_train <- train %>%
dplyr::select(y) %>%
bind_cols(preds_train,labels_train)
head(result_train,5)
```

```
##   y   .pred_0     .pred_1     .pred_class
## 1 1  0.3594703   0.6405297         1
## 2 1  0.5994656   0.4005344         0
## 3 0  0.7084139   0.2915861         0
## 4 1  0.2725499   0.7274501         1
## 5 1  0.3174165   0.6825835         1
```

Prediction are based on the highest probability. Since "Yes(1)" has higher probability value for the first observation we will assume that this observation predicts as "1".

Model Performance for Train

```
conf_mat(result_train, truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0  209  55
##          1  33   74
```

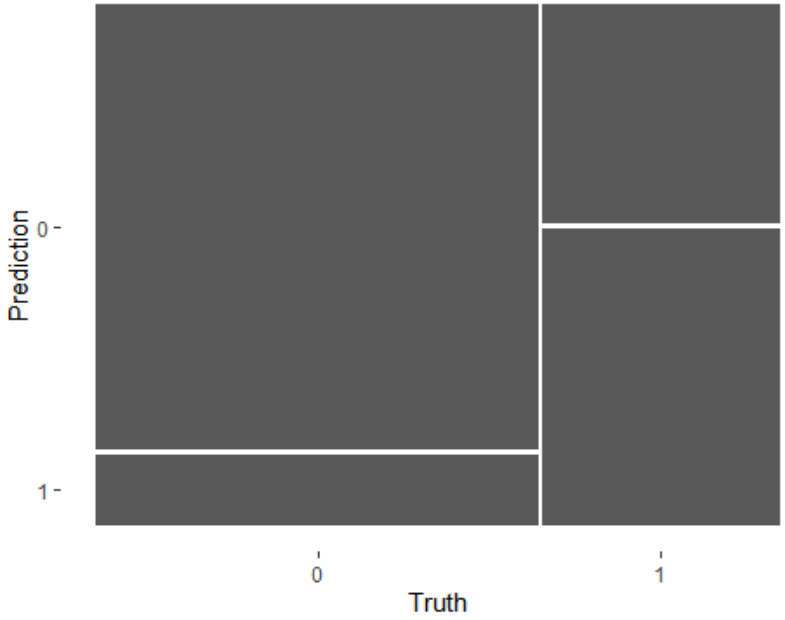
```
conf_mat(result_train, truth = y, estimate = .pred_class) %>%
summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.763
##  2 kap                  binary         0.455
##  3 sens                 binary         0.864
##  4 spec                 binary         0.574
##  5 ppv                  binary         0.792
##  6 npv                  binary         0.692
##  7 mcc                  binary         0.460
##  8 j_index              binary         0.437
##  9 bal_accuracy         binary         0.719
## 10 detection_prevalence binary         0.712
## 11 precision            binary         0.792
## 12 recall               binary         0.864
## 13 f_meas               binary         0.826
```

According to the result, the response variables in the test data are classified correctly by 76%.

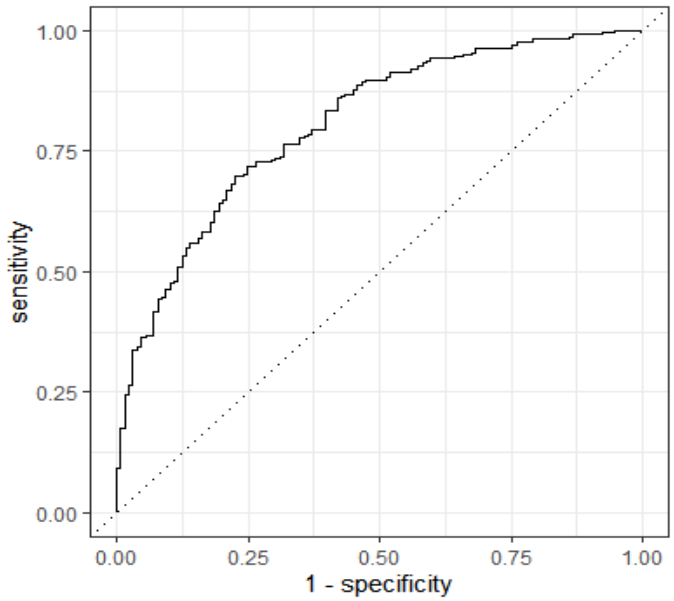
Visualizing Model Performance

```
conf_mat(result_train, truth = y, estimate = .pred_class) %>%
autoplot(type = 'mosaic')
```



ROC Curves

```
# Visualizing performance across thresholds
result_train %>%
roc_curve(truth = y, estimate = .pred_0) %>%
autoplot()
```



The ROC curve shows the trade-off between sensitivity and specificity. Classifiers that give curves closer to the top-left corner indicate a better performance.

AUC: Area Under the ROC Curve

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0 ; one whose predictions are 100% correct has an AUC of 1.

```
# Calculating ROC AUC
roc_auc(result_train, truth = y, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator   .estimate
##   <chr>    <chr>           <dbl>
## 1 roc_auc  binary          0.806
```

Making Predictions (Test)

```
# Estimated labels
labels_test <- logistic_fit %>%
predict(new_data = test , type = 'class')
head(labels_test,5)
```

```
## # A tibble: 5 x 1
##   .pred_class
##   <fct>
## 1 0
## 2 1
## 3 1
## 4 0
## 5 0
```

```
# Estimated probabilities
preds_test <- logistic_fit %>%
predict(new_data = test, type = 'prob')
head(preds_test, 5)
```

```
## # A tibble: 5 x 2
##    .pred_0  .pred_1
##     <dbl>    <dbl>
## 1   0.859    0.141
## 2   0.409    0.591
## 3   0.285    0.715
## 4   0.514    0.486
## 5   0.566    0.434
```

```
result_test <- test %>%
dplyr::select(y) %>%
bind_cols(preds_test,labels_test)
head(result_test,5)
```

```
##    y    .pred_0     .pred_1      .pred_class
## 9  0   0.8593225   0.1406775           0
## 15 0   0.4085836   0.5914164           1
## 25 0   0.2847260   0.7152740           1
## 27 0   0.5144042   0.4855958           0
## 32 1   0.5660336   0.4339664           0
```

Prediction are based on the highest probability. Since "No(0)" has higher probability value for the first observation we will assume that this observation predicts as "0".

Model Performance for Test

```
conf_mat(result_test, truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0  48  17
##          1  12  14
```

```
conf_mat(result_test, truth = y, estimate = .pred_class) %>%
summary()
```

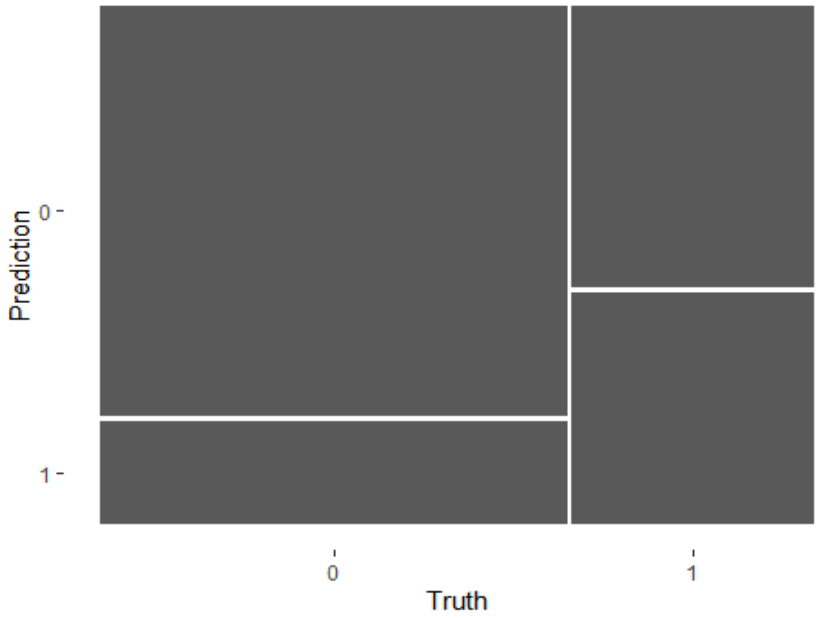
```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.681
##  2 kap                  binary         0.262
##  3 sens                 binary         0.8
##  4 spec                 binary         0.452
##  5 ppv                  binary         0.738
##  6 npv                  binary         0.538
##  7 mcc                  binary         0.264
##  8 j_index              binary         0.252
##  9 bal_accuracy         binary         0.626
## 10 detection_prevalence binary         0.714
## 11 precision            binary         0.738
## 12 recall               binary         0.8
## 13 f_meas               binary         0.768
```

According to the result, the response variables in the test data are classified correctly by 68%.

- The train set accuracy is 0.76, while the test set accuracy is 0.68. Therefore, there might be a overfitting problem.

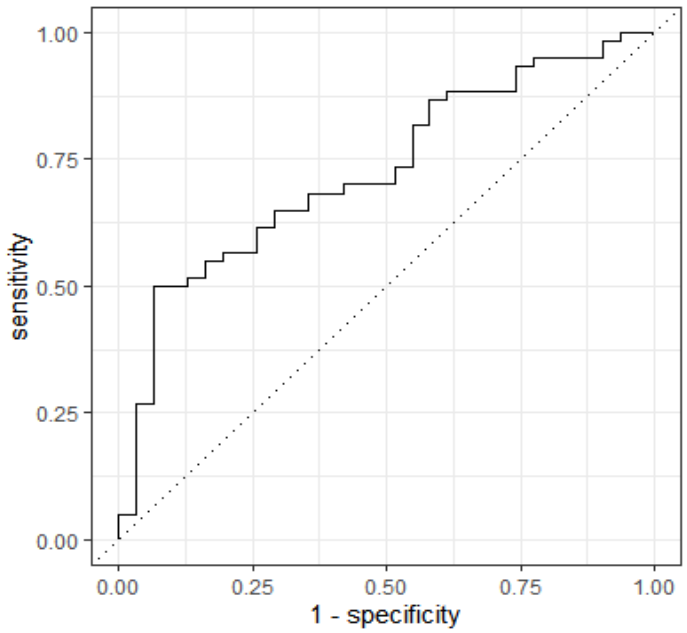
Visualizing Model Performance

```
conf_mat(result_test, truth = y, estimate = .pred_class) %>%
autoplot(type = 'mosaic')
```



ROC Curves

```
# Visualizing performance across thresholds
result_test %>%
roc_curve(truth = y, estimate = .pred_0) %>%
autoplot()
```



The ROC curve shows the trade-off between sensitivity and specificity. Classifiers that give curves closer to the top-left corner indicate a better performance.

AUC: Area Under the ROC Curve

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0 ; one whose predictions are 100% correct has an AUC of 1.

```
# Calculating ROC AUC
roc_auc(result_test, truth = y, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric   .estimator   .estimate
##   <chr>     <chr>           <dbl>
## 1 roc_auc   binary          0.728
```

- We can interpret the results as train set model predictions are 80% correct and test set model predictions are 72% correct.

Feature Engineering for Test

In this section, feature engineering methods were applied to the data and model performances were calculated again.

```
# creating a recipe object
data_recipe <- recipe(y ~ ., data = train) %>%

# Removed correlated predictors
step_corr(all_numeric(), threshold = 0.8) %>%

# Normalize numeric predictors
step_normalize(all_numeric()) %>%

# Create dummy variables
step_dummy(all_nominal(), -all_outcomes())

# Train recipe
data_recipe_prep <- data_recipe %>%
prep(training = train)

# Transform training data
data_train_prep <- data_recipe_prep %>%
bake(new_data = train)

# Transform test data
data_test_prep <- data_recipe_prep %>%
bake(new_data = test)
```

```r
# Train logistic model
logistic_fit <- logistic_model %>%
fit(y ~., data = data_train_prep)

# Obtain class predictions
class_preds_test <- predict(logistic_fit, new_data = data_test_prep, type = 'class
')

# Obtain estimated probabilities
prob_preds_test <- predict(logistic_fit, new_data = data_test_prep, type = 'prob')
```

```r
# Combine test set results
result_feature <- data_test_prep %>%
dplyr::select(y) %>%
bind_cols(class_preds_test, prob_preds_test)
head(result_feature,5)
```

```
## # A tibble: 5 x 4
##   y     .pred_class  .pred_0  .pred_1
##   <fct>  <fct>       <dbl>    <dbl>
## 1  0      0          0.859    0.141
## 2  0      1          0.409    0.591
## 3  0      1          0.285    0.715
## 4  0      0          0.514    0.486
## 5  1      0          0.566    0.434
```

Values for the performance of the model are calculated below. **(Accuracy – Sensitivity - Specificity)**

```r
# calculating accuracy of the model
accuracy(result_feature, truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##    .metric   .estimator   .estimate
##    <chr>       <chr>        <dbl>
## 1  accuracy    binary       0.681
```

```r
# calculating sensivity of the model
sens(result_feature, truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##    .metric   .estimator .estimate
##    <chr>      <chr>       <dbl>
## 1  sens       inary       0.8
```

```r
# calculating specificity of the model
spec(result_feature, truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##    .metric  .estimator  .estimate
##    <chr>      <chr>       <dbl>
## 1  spec       binary      0.452
```

Feature Engineering for Train

```
# Obtain class predictions
class_preds_train <- predict(logistic_fit, new_data = data_train_prep, type = 'cla
ss')

# Obtain estimated probabilities
prob_preds_train <- predict(logistic_fit, new_data = data_train_prep, type = 'prob
')


# Combine test set results
result_feature2 <- data_train_prep %>%
dplyr::select(y) %>%
bind_cols(class_preds_train, prob_preds_train)
head(result_feature2,5)
```

```
## # A tibble: 5 x 4
##   y     .pred_class   .pred_0  .pred_1
##   <fct> <fct>           <dbl>   <dbl>
## 1  1     1              0.359   0.641
## 2  1     0              0.599   0.401
## 3  0     0              0.708   0.292
## 4  1     1              0.273   0.727
## 5  1     1              0.317   0.683
```

Values for the performance of the model are calculated below. **(Accuracy – Sensitivity - Specificity)**

```
# calculating accuracy of the model
accuracy(result_feature2, truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##    .metric  .estimator   .estimate
##    <chr>     <chr>          <dbl>
## 1  accuracy  binary         0.763
```

```
# calculating sensivity of the model
sens(result_feature2, truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator   .estimate
##    <chr>    <chr>          <dbl>
## 1  sens     binary         0.864
```

```
# calculating specificity of the model
spec(result_feature2, truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##    .metric  .estimator   .estimate
##    <chr>     <chr>          <dbl>
## 1  spec      binary         0.574
```

Result

All performance metrics are equal. As a result, there was no positive or negative change in the performance of the model after the application of feature engineering methods.

	Accuracy	Sensitivity	Specificity
Without Feature Engineering (Train)	0.763	0.864	0.574
Without Feature Engineering (Test)	0.681	0.8	0.452
Feature Engineering (Train)	0.763	0.864	0.574
Feature Engineering (Test)	0.681	0.8	0.452