Wine Quality Regularized Logistic Regression

Jim Harner

1/6/2020

6.5 Wine Quality Logistic Regression

This section explores feature selection based on regularization.

6.5.1 Regularized Logistic Regression with Spark

We now revisit the Wine Quality Data Set analyzed in Section 6.4.2. Our goal is to continue with advanced analyses.

We read the `winequality-red.csv` file into a Spark DataFrame using `spark_red_csv`. We want to ensure the training and test data frames are identical to those in Section 6.4.2.

```
wine_red_sdf <- spark_read_csv(sc, "wine_red_sdf",
    path = "file:///home/rstudio/rspark-tutorial/data/wine/winequality-red.csv",
    delimiter = ";" )
wine_red_tbl <- sdf_register(wine_red_sdf, name = "wine_red_tbl")
```

We split `wine_red_sdf` into a training and a test Spark DataFrame as before and cast `quality` as numeric in order to binarize it with a threshold.

```
wine_red_partition <- wine_red_tbl %>%
  mutate(quality = as.numeric(quality)) %>%
  ft_binarizer(input_col = "quality", output_col = "quality_bin",
               threshold = 5.0) %>%
  sdf_random_split(training = 0.7, test = 0.3, seed = 2)
wine_red_train_sdf <- wine_red_partition$training
wine_red_test_sdf <- wine_red_partition$test
```

The full model is now run on the training data.

```
wine_red_br_full_model <- wine_red_train_sdf %>%
  ml_logistic_regression(quality_bin ~ fixed_acidity + volatile_acidity
                         + citric_acid + residual_sugar + chlorides
                         + free_sulfur_dioxide + total_sulfur_dioxide
                         + density + pH + sulphates + alcohol)
summary(wine_red_br_full_model)
```

```
## Coefficients:
##          (Intercept)        fixed_acidity     volatile_acidity
##           0.10386908           0.10927004          -3.10374892
##          citric_acid       residual_sugar            chlorides
##          -1.16424525           0.03660755          -3.76527703
##  free_sulfur_dioxide total_sulfur_dioxide              density
##           0.02952574          -0.01835148          -7.82197894
##                   pH            sulphates              alcohol
##          -0.55318652           2.70374827           0.91992634
```

The coefficients and AUC can be extracted from the `ml_model` object by:

```
wine_red_br_full_model$coefficients
```

```
##          (Intercept)        fixed_acidity     volatile_acidity
##           0.10386908           0.10927004          -3.10374892
##          citric_acid       residual_sugar            chlorides
##          -1.16424525           0.03660755          -3.76527703
##  free_sulfur_dioxide total_sulfur_dioxide              density
##           0.02952574          -0.01835148          -7.82197894
##                   pH            sulphates              alcohol
##          -0.55318652           2.70374827           0.91992634
```

```
wine_red_br_full_model$summary$area_under_roc
```

```
## function ()
## invoke(jobj, "areaUnderROC")
## <bytecode: 0x55ac5d043a38>
## <environment: 0x55ac5d046090>
```

However, it is preferable to use an evaluator, in this case `ml_binary_classification_evaluator`, to compute the performance metrics.

```
wine_red_br_full_predict <- ml_predict(wine_red_br_full_model, wine_red_train_sdf)
wine_red_br_auc <- data.frame(lambda = 0,
                              auc = ml_binary_classification_evaluator(wine_red_br_full_predict))
wine_red_br_coef <- as.data.frame(wine_red_br_full_model$coefficients[-1])
wine_red_br_coef
```

```
##                      wine_red_br_full_model$coefficients[-1]
## fixed_acidity                                     0.10927004
## volatile_acidity                                 -3.10374892
## citric_acid                                      -1.16424525
## residual_sugar                                    0.03660755
## chlorides                                        -3.76527703
## free_sulfur_dioxide                               0.02952574
## total_sulfur_dioxide                             -0.01835148
## density                                          -7.82197894
## pH                                               -0.55318652
## sulphates                                         2.70374827
## alcohol                                           0.91992634
```

Next we define a model function with the `reg_param` as an argument.

```
wine_red_br_model <- function(l) {
  wine_red_train_sdf %>%
    ml_logistic_regression(quality_bin ~ fixed_acidity + volatile_acidity
                           + citric_acid + residual_sugar + chlorides
                           + free_sulfur_dioxide + total_sulfur_dioxide
                           + density + pH + sulphates + alcohol,
                           elastic_net_param = 1, reg_param = l)
}
```

We are dealing with a lasso since the `elastic_net_param` is 1.

We now calculate the `coefficients` and `auc` for each of the models.

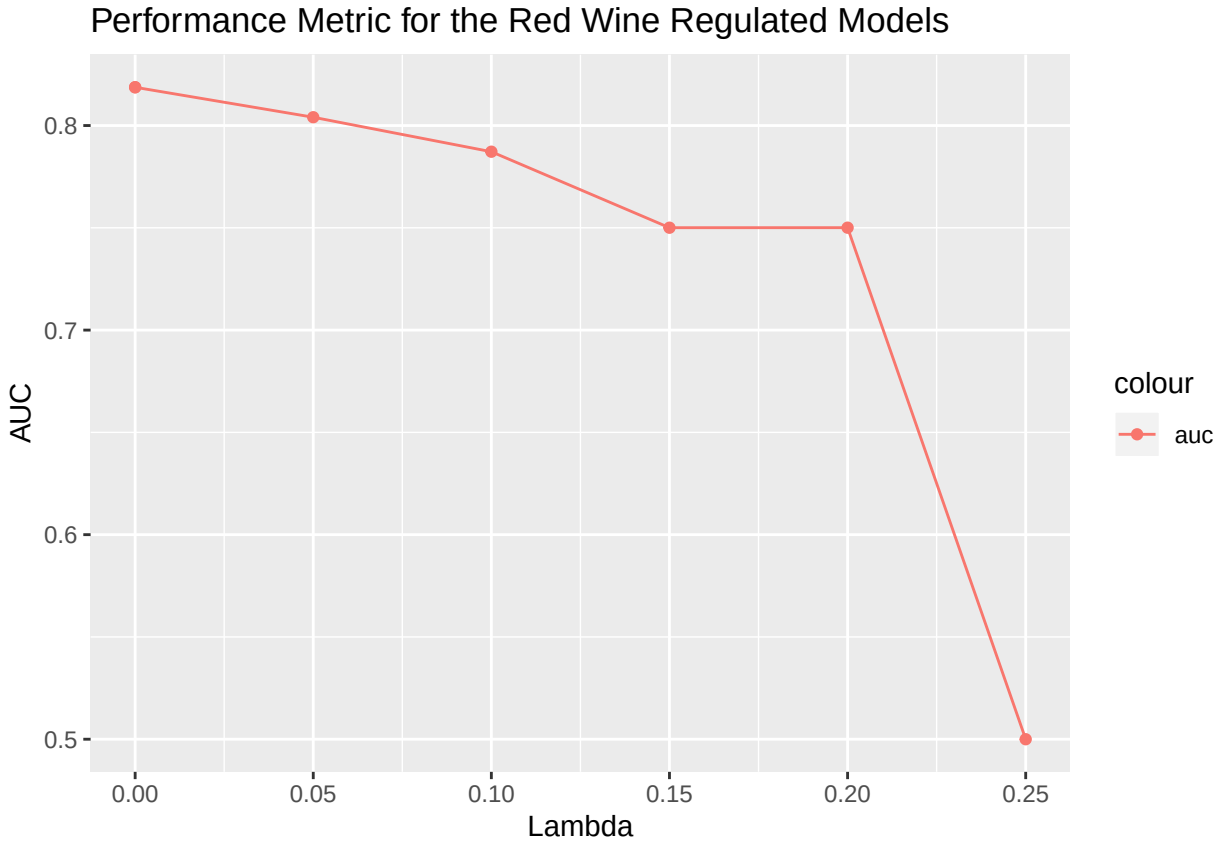
```
reg_parm <- c(0.0, 0.05, 0.1, 0.15, 0.2, 0.25)
for(l in reg_parm) {
  wine_red_br_fit <- wine_red_br_model(l)
```

```
  wine_red_br_predict <- ml_predict(wine_red_br_fit, wine_red_train_sdf)
  wine_red_br_auc <- data.frame(lambda = l,
                        auc = ml_binary_classification_evaluator(wine_red_br_predict)) %>%
    rbind(wine_red_br_auc, .)
  wine_red_br_coef <-
    as.data.frame(wine_red_br_fit$model$coefficients) %>%
    cbind(wine_red_br_coef, .)
}
wine_red_br_auc
```

```
##   lambda       auc
## 1   0.00 0.8187073
## 2   0.00 0.8187073
## 3   0.05 0.8040567
## 4   0.10 0.7871996
## 5   0.15 0.7500512
## 6   0.20 0.7500701
## 7   0.25 0.5000000
```

We plot AUC, the chosen performance metric, against λ.

```
library(ggplot2)
wine_red_br_auc %>%
  ggplot(aes(x = lambda)) +
  geom_point(aes(y = auc, color = 'auc')) +
  geom_line(aes(y = auc, color = 'auc')) +
  ggtitle("Performance Metric for the Red Wine Regulated Models") +
  xlab("Lambda") + ylab("AUC")
```



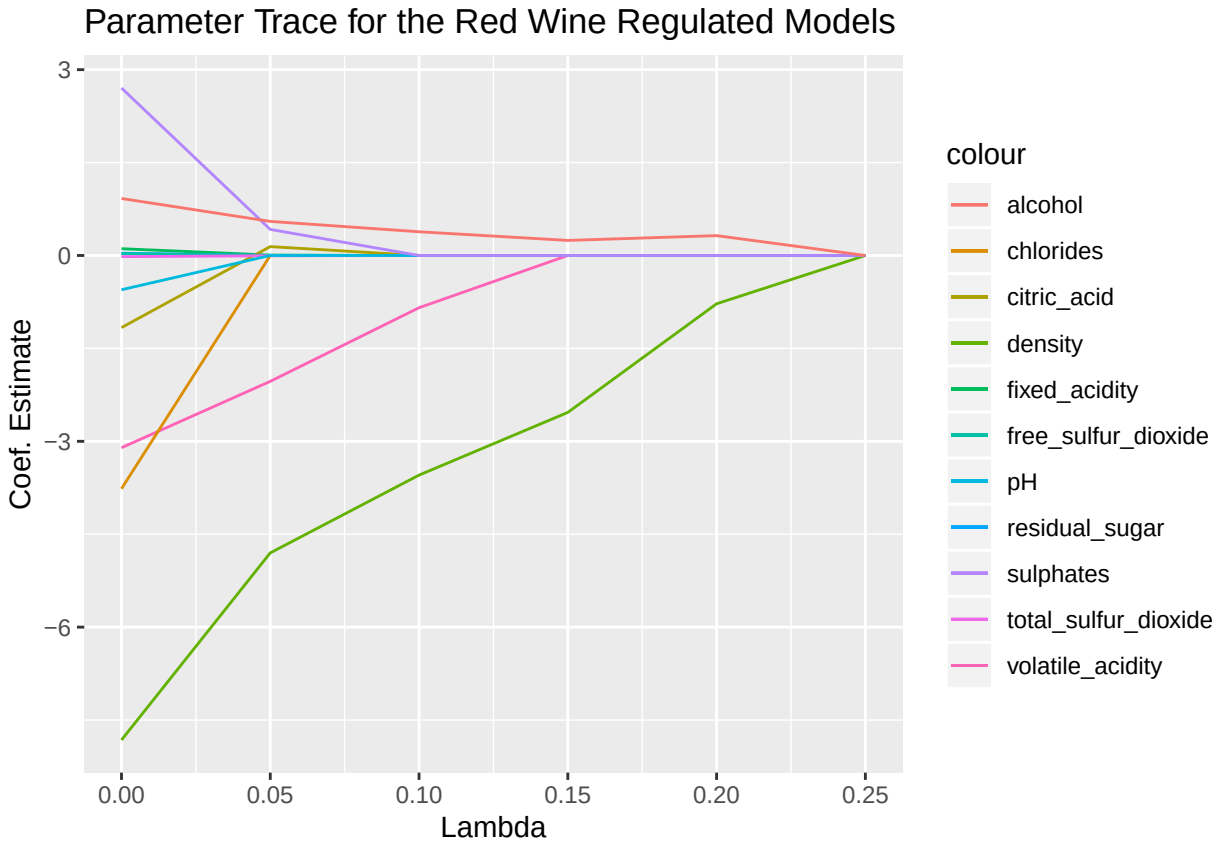
The AUC decreases with λ and thus little if any regularization should be done.

```
names(wine_red_br_coef) <- as.character(rbind(c(0.0, reg_parm)))
wine_red_br_coef
```

```
##                              0           0          0.05        0.1
## fixed_acidity         0.10927004  0.10927004  0.0063349132  0.0000000
## volatile_acidity     -3.10374892 -3.10374892 -2.0322000066 -0.8434967
## citric_acid          -1.16424525 -1.16424525  0.1432684769  0.0000000
## residual_sugar        0.03660755  0.03660755  0.0000000000  0.0000000
## chlorides            -3.76527703 -3.76527703  0.0000000000  0.0000000
## free_sulfur_dioxide   0.02952574  0.02952574  0.0000000000  0.0000000
## total_sulfur_dioxide -0.01835148 -0.01835148 -0.0047816112  0.0000000
## density              -7.82197894 -7.82197894 -4.8010143001 -3.5460383
## pH                   -0.55318652 -0.55318652 -0.0001675281  0.0000000
## sulphates             2.70374827  2.70374827  0.4208536232  0.0000000
## alcohol               0.91992634  0.91992634  0.5500803602  0.3827777
##                           0.15        0.2 0.25
## fixed_acidity         0.0000000  0.0000000    0
## volatile_acidity      0.0000000  0.0000000    0
## citric_acid           0.0000000  0.0000000    0
## residual_sugar        0.0000000  0.0000000    0
## chlorides             0.0000000  0.0000000    0
## free_sulfur_dioxide   0.0000000  0.0000000    0
## total_sulfur_dioxide  0.0000000  0.0000000    0
## density              -2.5337397 -0.7795136    0
## pH                    0.0000000  0.0000000    0
## sulphates             0.0000000  0.0000000    0
## alcohol               0.2430855  0.3200494    0
```

The interpretation is better if we visualize the coefficient traces.

```
library(ggplot2)
as.data.frame(cbind(lambda = c(0.0, reg_parm), t(wine_red_br_coef))) %>%
  ggplot(aes(x = lambda)) +
  geom_line(aes(y = fixed_acidity, color = 'fixed_acidity')) +
  geom_line(aes(y = volatile_acidity, color = 'volatile_acidity')) +
  geom_line(aes(y = citric_acid, color = 'citric_acid')) +
  geom_line(aes(y = residual_sugar, color = 'residual_sugar')) +
  geom_line(aes(y = chlorides, color = 'chlorides')) +
  geom_line(aes(y = free_sulfur_dioxide, color = 'free_sulfur_dioxide')) +
  geom_line(aes(y = total_sulfur_dioxide, color = 'total_sulfur_dioxide')) +
  geom_line(aes(y = density, color = 'density')) +
  geom_line(aes(y = pH, color = 'pH')) +
  geom_line(aes(y = sulphates, color = 'sulphates')) +
  geom_line(aes(y = alcohol, color = 'alcohol')) +
  ggtitle("Parameter Trace for the Red Wine Regulated Models") +
  xlab("Lambda") + ylab("Coef. Estimate")
```



The coefficients go to 0 very quickly. Based on regularization, `alcohol` and `density` are still standing at $\lambda = 0.2$, but then they too go to 0. However, for $\lambda > 0$ the AUC is degraded.

We now `collect` the training and test Spark DataFrames into R as regular data frames. If you experiment with `alpha` and `lambda`,i.e., invoke the elastic net, you will see the coefficients that are driven to 0 vary greatly.

```
wine_red_train_df <- collect(wine_red_partition$training)
wine_red_test_df <- collect(wine_red_partition$test)
```

6.5.2 Regularized Logistic Regression with glmnet

We can now use `glmnet` to model the wine quality.

```
wine_red.x <- model.matrix(as.factor(quality_bin) ~ fixed_acidity
                           + volatile_acidity + citric_acid + residual_sugar
                           + chlorides + free_sulfur_dioxide
                           + total_sulfur_dioxide + density + pH + sulphates
                           + alcohol,
                           data = wine_red_train_df)[, -1]
wine_red.y <- wine_red_train_df$quality_bin

wine_red_bin <- glmnet(x = wine_red.x, y = wine_red.y, family = "binomial",
                       alpha = 1, lambda = c(0.0, 0.05, 0.1, 0.15, 0.2),
                       standardize = TRUE)
coef(wine_red_bin, s = c(0.0, 0.05, 0.1, 0.15, 0.2))
```

```
## 12 x 5 sparse Matrix of class "dgCMatrix"
```

```
##                                1            2          3          4
## (Intercept)           74.757333746 -5.844010852 -4.1092098 -2.5414788
## fixed_acidity          0.214653313  .            .          .
## volatile_acidity      -3.355637998 -1.762840484 -0.8015376  .
## citric_acid           -1.472263078  .            .          .
## residual_sugar         0.068535001  .            .          .
## chlorides             -2.891671705  .            .          .
## free_sulfur_dioxide    0.026636291  .            .          .
## total_sulfur_dioxide  -0.017697980 -0.004566334  .          .
## density              -84.379561425  .            .          .
## pH                    -0.005033606  .            .          .
## sulphates              2.651073108  0.719790932  .          .
## alcohol                0.830848216  0.638758858  0.4466272  0.2543576
##                                5
## (Intercept)          -0.49113816
## fixed_acidity         .
## volatile_acidity      .
## citric_acid           .
## residual_sugar        .
## chlorides             .
## free_sulfur_dioxide   .
## total_sulfur_dioxide  .
## density               .
## pH                    .
## sulphates             .
## alcohol               0.05708543
```

Based on feature importance, `alcohol`, was indeed most important. You can experiment with different values of `alpha` and `lambda`.

```
wine_red_glm_lasso_cv <- cv.glmnet(x =  wine_red.x, y = wine_red.y, family = "binomial",
                                   nfolds = 10, standardize = TRUE,
                                   type.measure = "auc", alpha = 1,
                                   lambda = c(0.0, 0.05, 0.1, 0.15, 0.2))
wine_red_glm_lasso_cv$lambda.min
```

```
## [1] 0
```

`glmnet` confirms that no regularization is needed and thus no variable selection is done. At this point it is not clear how to proceed since the standard errors of the coefficient estimates are not available and thus testing is not possible.

Since we do not have a well-determined final model, we will not compute predictions or performance metrics on the test data set. Of course `glm` in base R was used in Section 6.4.2. `pH`, `density`, and `residual_sugars` were removed using AIC as a criterion, but further analysis is needed. `ml_generalized_linear_regression` does provide information on the AIC and thus could also be used for variable selection as was done in Section 6.4.2.

```
spark_disconnect(sc)
```