Regression and classification

Anna Yeaton

Fall 2018

Lab Section

In this lab, we will go over regression, classification and performance metrics. We will be using the caret package in R. https://topepo.github.io/caret/train-models-by-tag.html

Perfomance Metrics

K- fold cross validatation - Resampling method

Randomly split the training data into k folds. If you specify 10 folds, then you split the data into 10 partitions. Train the model on 9 of those partitions, and test your model on the 10th partition. Iterate through until every partition has been held out.

A smaller k is more biased, but a larger k can be very variable.

Bootstrapping - Resampling method

Sample with replacement. Some samples may be represented several times within the boostrap sample, while others may not be represented at all. The samples that are not selected are called out of bag samples.

Boostrap error rates usually have less uncertainty than k-fold cross validation, but higher bias.

Error

Deviation of the observed value to the true value (population mean)

Residual

Deviation of the observed value to the estimated value (sample mean)

$$residual = y_i - \hat{y}_i$$

where $\hat{y}_i$ is the estimated value

Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE)

Same units as original data.

$$RMSE = \sqrt{MSE}$$

Total Sum of Squares (TSS)

$$TSS = \sum (y_i - \bar{y_i})^2$$

$\bar{y}$ is the mean of y

Residual Sum of Squares

$$RSS = \sum (y_i - \hat{y_i})^2$$

Explained Sum of Squares

$$ESS = \sum (\hat{y_i} - \bar{y_i})^2$$

R^2

Proportion of information explained by the model. It is a measure of correlation, not accuracy.

$$1 - RSS/TSS$$

L2 regularization : Ridge regression. Regularize by adding the sum of the coefficients, squared, to the function.

$$RidgeRegression = \sum_{i=1}^{n} (y_i - w_0 - \sum_{j=1}^{p} w_j x_{ij})^2 + \lambda \sum_{j=1}^{p} (w_j)^2$$

L1 regularization : Lasso Regression. Regularize by adding the sum of the absolute value of the coefficients to the model. Coefficient estimates may be pushed to zero – Lasso can perform variable selection

$$LassoRegression = \sum_{i=1}^{n} (y_i - w_0 - \sum_{j=1}^{p} w_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |w_j|$$

Sensitivity or True Positive Rate

TP = True Positives TN = True Negatives FP = False Positives - Type I error FN = False Negatives - Type II error N = actual negative samples P = actual positive samples

$$TPR = TP/(TP + FN)$$

Specificity or True Negative Rate

$$TNR = TN/(TN + FP)$$

Receiver Operating Characteristics (ROC)

plot of True Positive Rate (sensitivity) against False Positive Rate, or plots the True Positive Rate (sensitivity) against specificity.

Either way, a good ROC curves up through the left corner, and has a large area underneath.

Area under ROC curve (AUC)

The area underneath the ROC curve

Cohen's Kappa

Like percent of correct predictions out of all predictions, but it also takes into account the probability of a correct prediction occuring by chance.

Logistic function:

$$P(X) = e^{w_0+w_1X}/1 + e^{w_0+w_1X}$$

The broad steps of Machine learning in R.

1. Split the data into training and test. Set test aside.
2. Fit a good model to the training data. This includes using bootstapping, cross validation etc. to resample the training data and fit a good model.
3. Visualize if your model learned on the training data by looking at ROC curve and AUC.
4. Test how your model performs on the test data.

Broad steps for choosing between models according to Max Kuhn and Kjell Johnson

1. Start with several models that are the least interpretable and the most flexible, like boosted trees and svms. These models are the often the most accurate.
2. Investigate simpler models that are less opaque, like partial least squares, generalized additive models, or naive bayes models.
3. Consider using the simplest model that reasonable approximates the performance of more complex models

Regression

1. Split data into training and test set

```
train_size <- floor(0.75 * nrow(airquality))
set.seed(543)
train_pos <- sample(seq_len(nrow(airquality)), size = train_size)
train_regression <- airquality[train_pos,-c(1,2)]
test_regression <- airquality[-train_pos,-c(1,2)]

dim(train_regression)
```

```
## [1] 114   4
```

```
dim(test_regression)
```

```
## [1] 39  4
```

Linear Regression

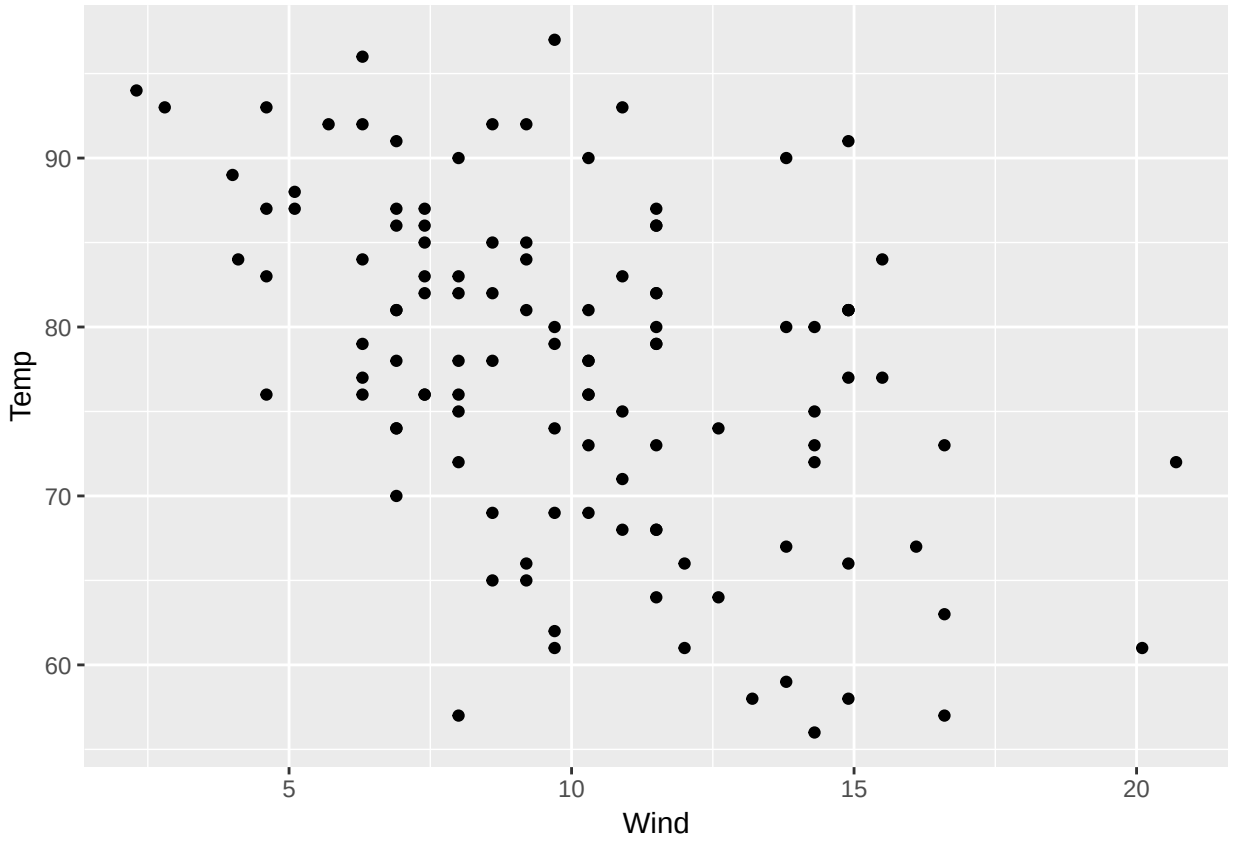
- Assumes a linear relationship.
- Independent variables should not be correlated (no mulitcollinearity)
- The number of observations should be greater than the number of independent variables.

$$RSS = \sum (y_i - \hat{y}_i)^2$$

We will predict the response of the Temperature based on Wind.

This is the data we will fit a linear model to.

```
ggplot(data = train_regression) +
  geom_point(aes(x=Wind, y=Temp))
```



2. Create and fit a linear model using the training set

```
#help(train)
linear_regression <- train(Temp ~ Wind, data=train_regression, method = "lm")
```

```
linear_regression
```

```
## Linear Regression
##
## 114 samples
##   1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 114, 114, 114, 114, 114, 114, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   8.554026  0.2541839  6.974301
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
summary(linear_regression)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.2740  -5.7910   0.8357   6.1686  19.5403
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  90.4935     2.4296   37.25  < 2e-16 ***
## Wind         -1.2774     0.2281   -5.60 1.55e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.703 on 112 degrees of freedom
## Multiple R-squared:  0.2188, Adjusted R-squared:  0.2118
## F-statistic: 31.36 on 1 and 112 DF,  p-value: 1.546e-07
```

4. Explore how the model performs on the test data

- The residuals should be close to zero.
- There should be equal variance around the regression line (homoscedasticity).
- Residuals should be normally distributed.
- Independent variables and residuals should not be correlated.

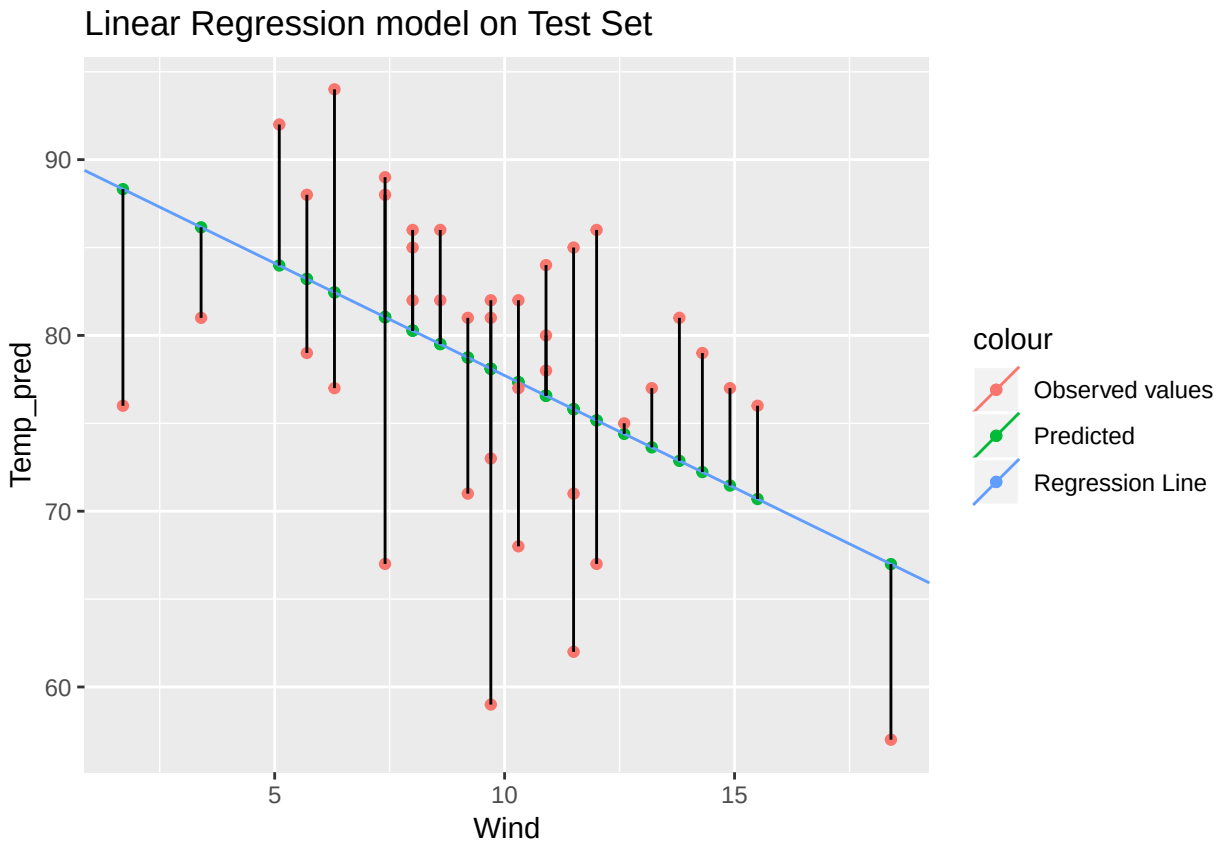
Visualize the predicted values. Look at for homoscedasticity

```r
#predict Temperature
linear_predict <- predict(linear_regression, newdata=test_regression)
plot_lin_pred <- data.frame(Temp_pred = linear_predict, Wind = test_regression$Wind, Temp = tes

# Extract coefficients from the model, plot the regression line on the predicted values, plot
linear_regression$finalModel$coefficients
```

```
## (Intercept)        Wind
##   90.493474   -1.277433
```

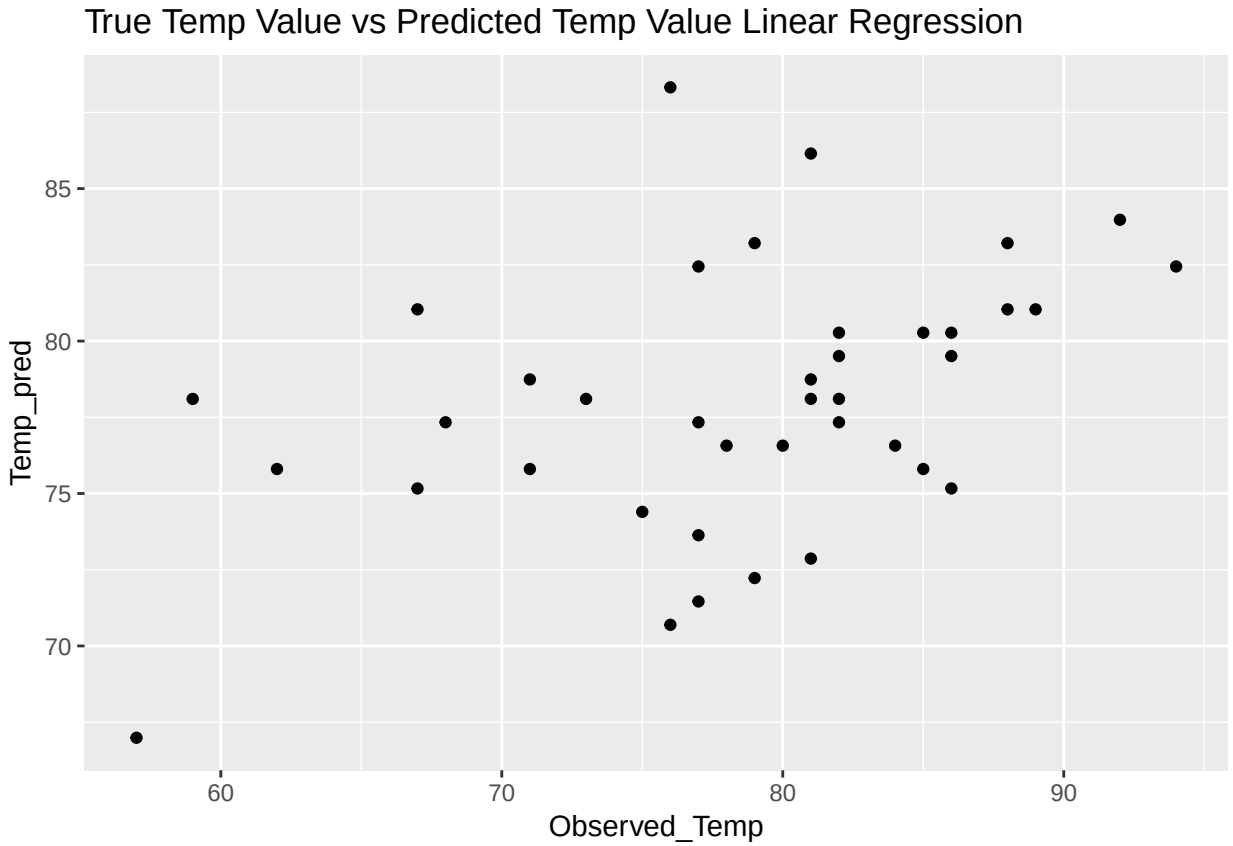
```r
ggplot(data = plot_lin_pred)+
  geom_point(aes(x=Wind, y = Temp_pred, col =  "Predicted")) +
  ggtitle("Linear Regression model on Test Set") +
  geom_abline(aes(intercept = 90.493474, slope = -1.27743, col="Regression Line")) +
  geom_point(aes(x = Wind, y = Temp, col = "Observed values")) +
  geom_segment(aes(x = Wind, xend = Wind, y = Temp,yend = Temp_pred))
```



Examine the residuals by comparing predicted temperature to the observed temperature

```
#plot predicted vs observed temp. A strong model should show a strong correlation
plot_lin_pred_temp <- data.frame(Temp_pred = linear_predict, Observed_Temp = test_regression$Te

ggplot(data = plot_lin_pred_temp) +
  geom_point(aes(x=Observed_Temp, y = Temp_pred)) +
  ggtitle("True Temp Value vs Predicted Temp Value Linear Regression")
```

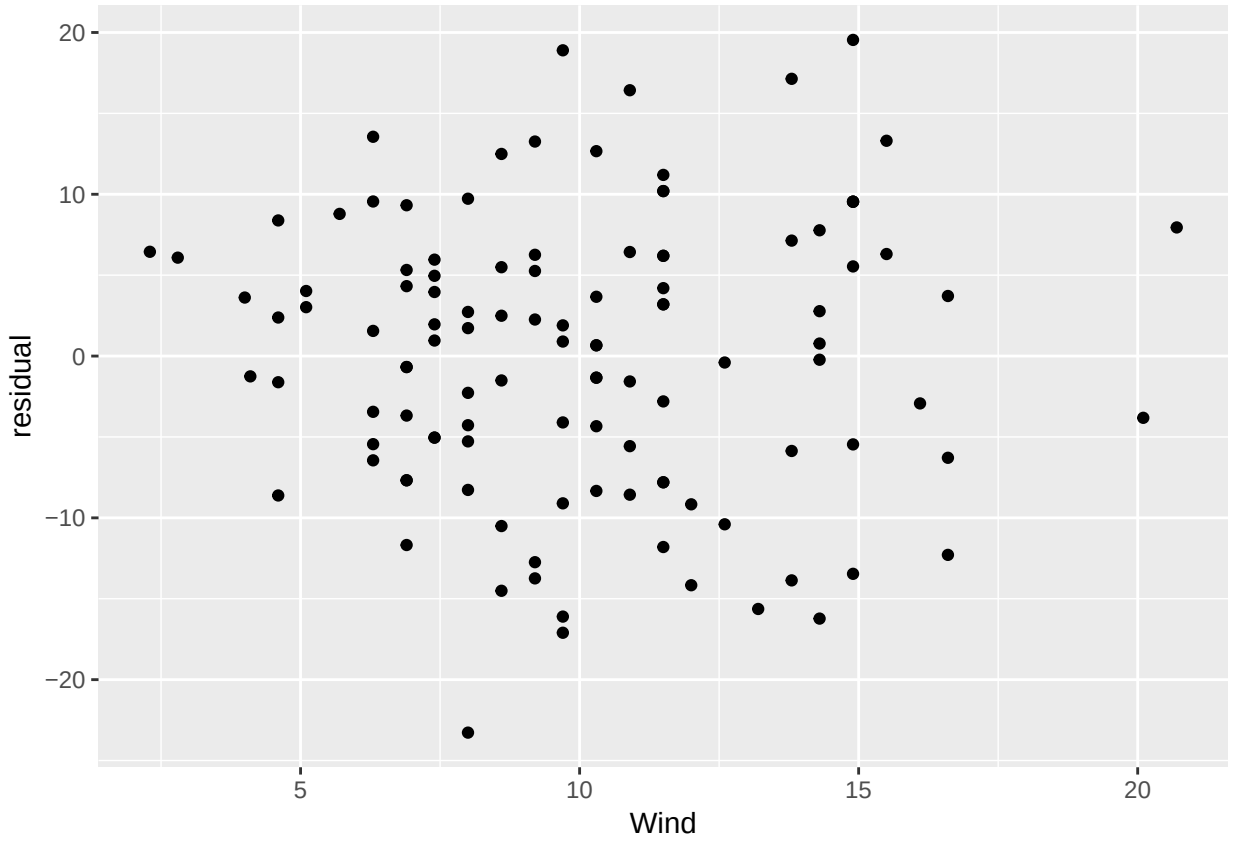



```
#look at the median residual value. Close to zero is best
summary(linear_regression)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.2740  -5.7910   0.8357   6.1686  19.5403
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  90.4935     2.4296   37.25  < 2e-16 ***
## Wind         -1.2774     0.2281   -5.60 1.55e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.703 on 112 degrees of freedom
## Multiple R-squared:  0.2188, Adjusted R-squared:  0.2118
## F-statistic: 31.36 on 1 and 112 DF,  p-value: 1.546e-07
```

Residuals should be normally distributed. Plot the residuals against the fitted values.

```
residuals_lin <- residuals(linear_regression)
residvpredict <- data.frame(residual = residuals_lin, Wind = train_regression$Wind)
ggplot(data=residvpredict)+
  geom_point( aes(x=Wind, y = residual))
```



Independent variables and residuals should not be correlated

```
cor.test(train_regression$Wind, resid(linear_regression))
```

```
##
##  Pearson's product-moment correlation
##
## data:  train_regression$Wind and resid(linear_regression)
## t = -1.5625e-15, df = 112, p-value = 1
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1839148  0.1839148
## sample estimates:
##          cor
## -1.47641e-16
```

Ridge Regression

- Assumes a linear relationship.
- Observations may outnumber samples
- Independent variables may be correlated

$$RidgeRegression = \sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1}^{p} w_j x_{ij})^2 + \lambda \sum_{j=1}^{p}(w_j)^2$$

2. Create and train model

```
ctrl =  trainControl(method = "boot", 15)
#when doing ridge regression, you should center your data

#Ridge_regression <-
```

```
#Ridge_regression
```

Examine the residuals

```
#ridge_test_pred <-

#plot the predicted values vs the observed values
#plot_ridge_test_pred <- data.frame(Temp_test_pred = ridge_test_pred, Observed_Temp = test_reg
#ggplot(data = plot_ridge_test_pred) +
#  geom_point(aes(x=Observed_Temp, y = Temp_test_pred)) +
#  ggtitle("True Temp Value vs Predicted Temp Value Ridge Regression")

#median residual value should be close to zero
#median(resid(Ridge_regression))
```

Classification

1. Split into training and test set

```
data(iris)

#split into training and test set
train_size <- floor(0.75 * nrow(iris))
set.seed(543)
train_pos <- sample(seq_len(nrow(iris)), size = train_size)
train_classifier <- iris[train_pos,]
test_classifier <- iris[-train_pos,]


dim(train_classifier)
```

```
## [1] 112   5
```

```
dim(test_classifier)
```

```
## [1] 38  5
```

Logistic Regression

- Y=1 is the probability of the event occuring.
- Independent variables should not be correlated.
- Log odds and independent variables should be linearly correlated.

2. Train and fit model

```
#only look at two classes
train_classifier_log <- train_classifier[c(which(train_classifier$Species == "setosa"), which(t
test_classifier_log <- test_classifier[c(which(test_classifier$Species == "setosa"), which(tes

train_classifier_log$Species <- factor(train_classifier_log$Species)
test_classifier_log$Species <- factor(test_classifier_log$Species)

ctrl <- trainControl(method = "repeatedcv", repeats = 15,classProbs = T, savePredictions = T)

#create model. logistic regression is a bionomial general linear model.
#predict species based on sepal length
logistic_regression <- train(Species~ Sepal.Length, data = train_classifier_log, method = "glm"
```

```
logistic_regression
```

```
## Generalized Linear Model
##
## 74 samples
##  1 predictor
##  2 classes: 'setosa', 'versicolor'
```

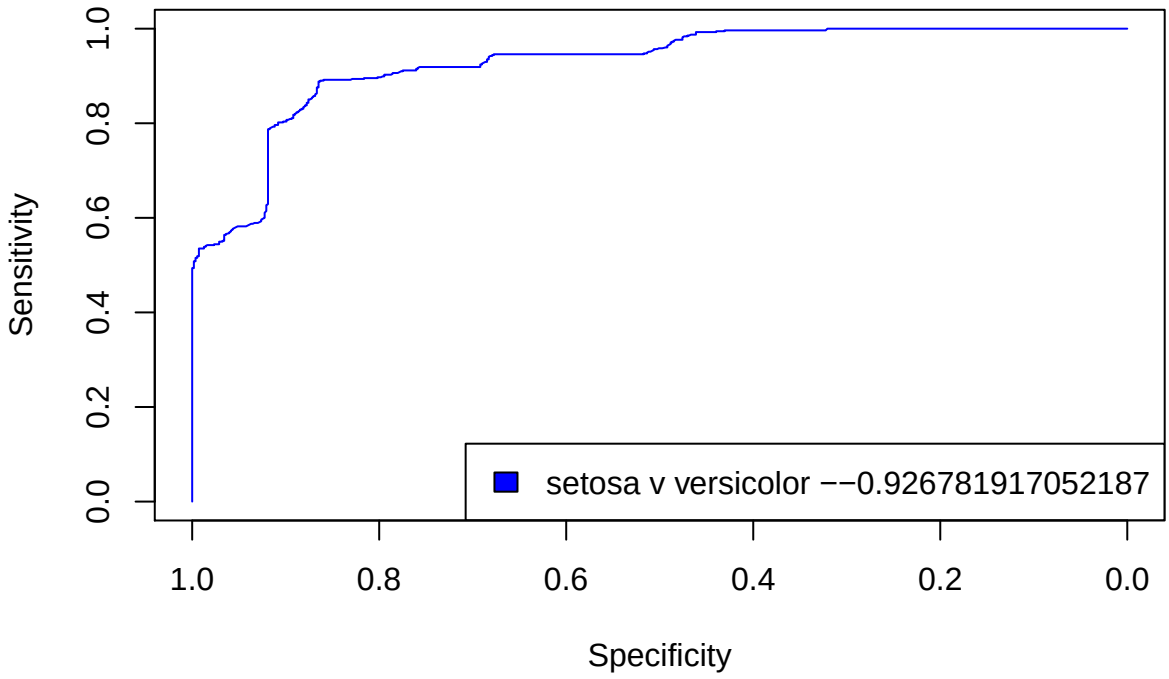
```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 15 times)
## Summary of sample sizes: 67, 67, 67, 67, 66, 66, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8751984  0.7483604
```

```
summary(logistic_regression)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.98465  -0.43322  -0.02248   0.40932   2.14846
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -27.343      6.362  -4.298 1.73e-05 ***
## Sepal.Length    5.028      1.168   4.305 1.67e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 102.586  on 73  degrees of freedom
## Residual deviance:  47.527  on 72  degrees of freedom
## AIC: 51.527
##
## Number of Fisher Scoring iterations: 6
```

3. Visualize ROC curve

```
plot(x = roc(predictor = logistic_regression$pred$setosa, response = logistic_regression$pred$
legend("bottomright", legend = paste("setosa v versicolor --", roc(predictor = logistic_regres
, sep = ""), col = c("blue"), fill = c("blue"))
```



4. Test on an independent set

```
#predict iris species using Sepal legth
logistic_regression_predict_class <- predict(logistic_regression, newdata = test_classifier_log

#confusion matrix
confusionMatrix(logistic_regression_predict_class, reference = test_classifier_log$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor
##   setosa         13          2
##   versicolor      0         11
##
##                Accuracy : 0.9231
##                  95% CI : (0.7487, 0.9905)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 5.245e-06
##
##                   Kappa : 0.8462
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.8462
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
```

```
##    Detection Prevalence : 0.5769
##       Balanced Accuracy : 0.9231
##
##        'Positive' Class : setosa
##
```

Check if log odds and independent variables are linearly correlated

```
logistic_regression_predict <- predict(logistic_regression, newdata = test_classifier_log, type
cor.test(logistic_regression_predict[,1], test_classifier_log$Sepal.Length)
```

```
##
##  Pearson's product-moment correlation
##
## data:  logistic_regression_predict[, 1] and test_classifier_log$Sepal.Length
## t = -12.572, df = 24, p-value = 4.736e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9692801 -0.8518601
## sample estimates:
##       cor
## -0.9317596
```

```
cor.test(logistic_regression_predict[,2], test_classifier_log$Sepal.Length)
```

```
##
##  Pearson's product-moment correlation
##
## data:  logistic_regression_predict[, 2] and test_classifier_log$Sepal.Length
## t = 12.572, df = 24, p-value = 4.736e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8518601 0.9692801
## sample estimates:
##       cor
## 0.9317596
```

Look deeper at the logistic regression

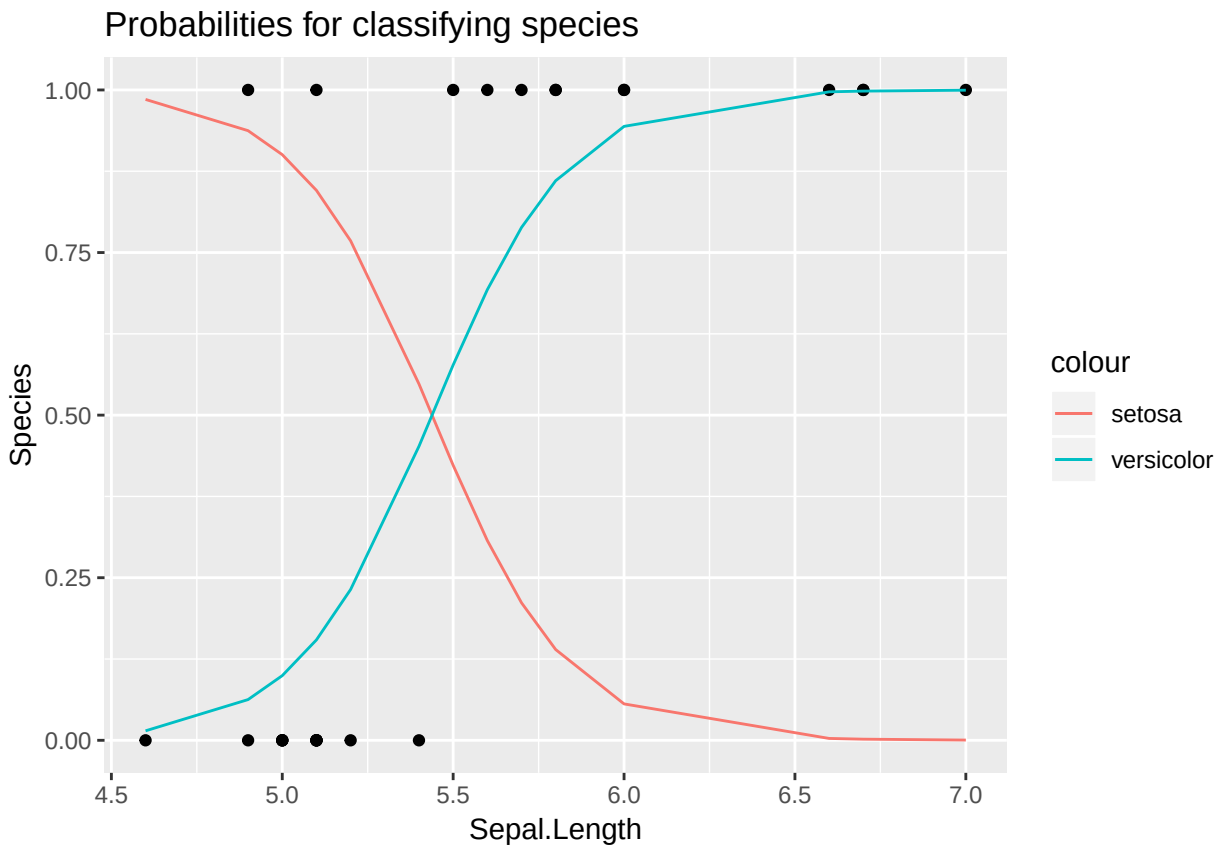
```
logistic_predict_prob <- predict(logistic_regression, newdata = test_classifier_log, type="prob

logistic_pred_prob_plot <- data.frame(Species_pred = logistic_predict_prob, Sepal.Length  = tes

test_classifier_log$Species <- as.numeric(test_classifier_log$Species) -1

ggplot(data = test_classifier_log) +
  geom_point(aes(x=Sepal.Length, y = Species)) +
  geom_line(data = logistic_pred_prob_plot, aes(x = Sepal.Length, y = Species_pred.setosa, col
  geom_line(data = logistic_pred_prob_plot, aes(x = Sepal.Length, y = Species_pred.versicolor,
```

```
ggtitle("Probabilities for classifying species")
```



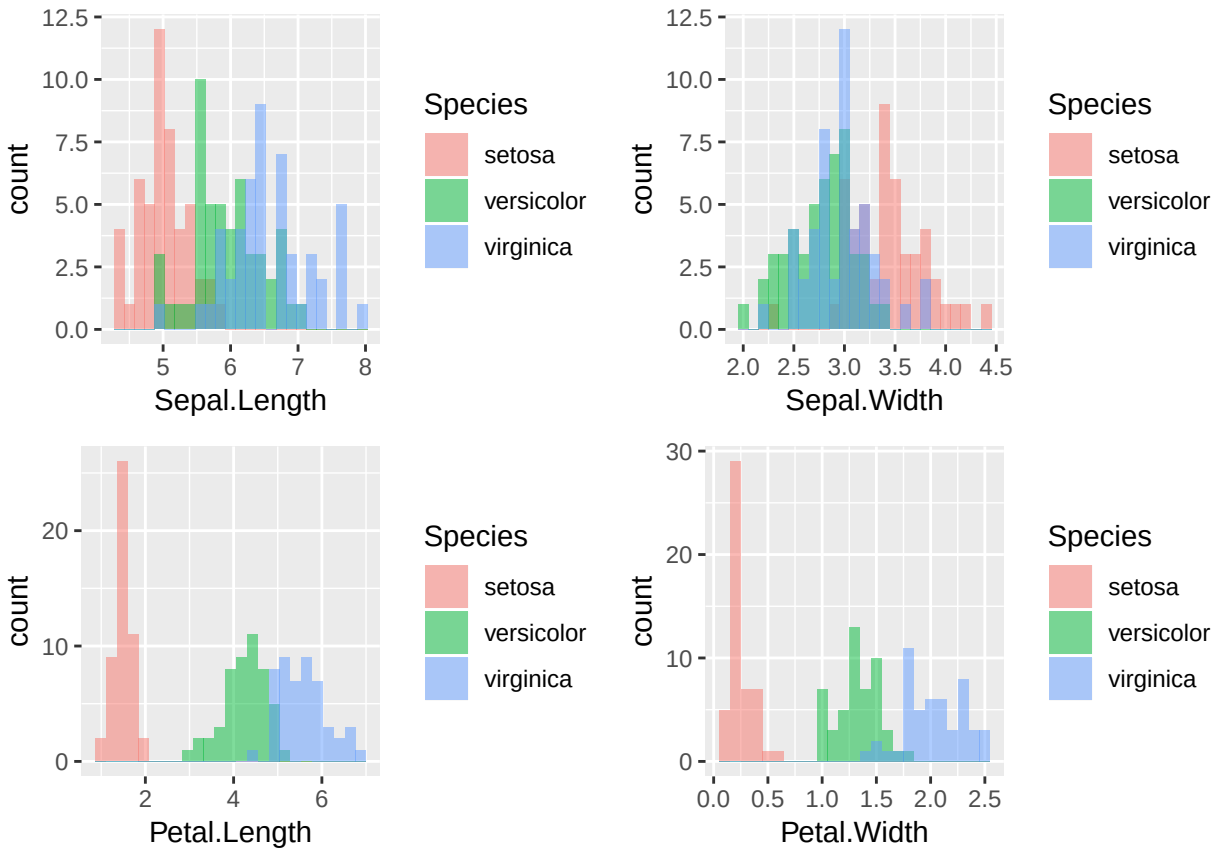
Linear Discriminant analysis

- Good for well separated classes, more stable with small n than logistic regression, and good for more than 2 response classes.
- LDA assumes a normal distribution with a class specific mean and common variance.

Let's see if our data follows the assumptions of LDA.

```
slength <- ggplot(data = iris, aes(x = Sepal.Length, fill = Species)) + geom_histogram(positio
swidth <- ggplot(data = iris, aes(x = Sepal.Width, fill = Species)) + geom_histogram(position=
plength <- ggplot(data = iris, aes(x = Petal.Length, fill = Species)) + geom_histogram(positio
pwidth <- ggplot(data = iris, aes(x = Petal.Width, fill = Species)) + geom_histogram(position=

grid.arrange(slength, swidth, plength, pwidth)
```




```
LDA <- lda(Species~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data= train_classi
```

```
LDA
```

```
## Call:
## lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
##     data = train_classifier, cv = T)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3303571  0.3303571  0.3392857
```

```
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.991892    3.427027     1.467568   0.2513514
## versicolor     5.929730    2.775676     4.316216   1.3486486
## virginica      6.631579    2.963158     5.584211   1.9921053
##
## Coefficients of linear discriminants:
##                    LD1        LD2
## Sepal.Length  1.053740 -0.3504079
## Sepal.Width   1.164761 -1.9499297
## Petal.Length -2.338670  1.0990348
## Petal.Width  -2.729158 -2.9432294
##
## Proportion of trace:
##    LD1    LD2
## 0.9928 0.0072
```

4. Test model on test set

```
#predict the species of the test data
LDA_predict <- predict(LDA, newdata=test_classifier)
confusionMatrix(LDA_predict$class, reference = test_classifier$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         13          0         0
##   versicolor      0         13         1
##   virginica       0          0        11
##
## Overall Statistics
##
##                Accuracy : 0.9737
##                  95% CI : (0.8619, 0.9993)
##     No Information Rate : 0.3421
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9605
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9167
## Specificity                 1.0000            0.9600           1.0000
## Pos Pred Value              1.0000            0.9286           1.0000
## Neg Pred Value              1.0000            1.0000           0.9630
```

```
## Prevalence                0.3421          0.3421          0.3158
## Detection Rate            0.3421          0.3421          0.2895
## Detection Prevalence      0.3421          0.3684          0.2895
## Balanced Accuracy         1.0000          0.9800          0.9583
```

Naive Bayes Classifier

- Independent variables should not be correlated

2. Train model

```
#ctrl <-
#naive_bayes <-
```

```
#naive_bayes
```

3. Visualize ROC curve

```
#versivvirg <- roc(predictor = naive_bayes$pred$virginica, response = naive_bayes$pred$obs, le
#setosvvirg <- roc(predictor = naive_bayes$pred$virginica, response = naive_bayes$pred$obs, le
#versivsetosa <- roc(predictor = naive_bayes$pred$virginica, response = naive_bayes$pred$obs,

#plot(x = roc(predictor = naive_bayes$pred$virginica, response = naive_bayes$pred$obs, levels=

#lines(x = roc(predictor = naive_bayes$pred$setosa, response = naive_bayes$pred$obs, levels= c

#lines(x = roc(predictor = naive_bayes$pred$versicolor, response = naive_bayes$pred$obs, level

#legend("bottomright", legend = c(paste("versicolor v virginica --", versivvirg, sep = ""), pa
```

4. Test model

```
#naive_bayes_pred <- predict(naive_bayes, newdata=test_classifier)
#confusionMatrix(naive_bayes, reference = test_classifier$Species)
```

Homework:

1. Use the Breast Cancer dataset from the mlbench package, and predict whether the cancer is malignant or benign using Logistic Regression, Linear Discriminate Analysis and Naive Bayes. Plot ROC curves, and confusion matrices. Evaluate which model is the best for this dataset.

```
#library(mlbench)
#data(BreastCancer)
#train
#test

##ctrl <- trainControl()
## <- train(, data = , method = "", trControl = ctrl)
## predict(, newdata="")
```

References: https://sebastianraschka.com/Articles/2014_python_lda.html

https://towardsdatascience.com/building-a-multiple-linear-regression-model-and-assumptions-of-linear-regression-a

http://www.statisticssolutions.com/wp-content/uploads/wp-post-to-pdf-enhanced-cache/1/assumptions-of-logistic-regression.pdf

https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/ , https://sebastianraschka.com/Articles/2014_python_lda.html