11. The Craft 3: Feature Engineering

In large, real-world data sets you may have hundreds of features for hundreds of thousands of observations. The majority of your time on such a project will be spent in data cleaning and feature engineering. The term **feature engineering** refers to tasks such as selecting features from the set of all features and also creating features from raw data, sometimes called **feature extraction**. We want a reasonably small set of predictors so that our algorithms do not bog down. As we have discussed, some algorithms like knn suffer from the curse of dimensionality and others like regression can handle several predictors. Nonetheless, we want to limit the data to features that are useful for our tasks.

11.1 Feature Selection

Many of the algorithms we have used have embedded methods to determine the value of predictors. For example, linear and logistic regression give us p-values so that we can gauge confidence in the predictive value of variables. Others do not so we need to use external methods.

11.1.1 Feature Selection with caret

The caret package has several useful functions for feature selection, we will explore a couple of them here. First, the findCorrelation() function which returns a list of columns that can be removed due to their correlation with other variables. We will use the PimaIndiansDiabetes2 data from package mlbench. First we remove rows that have NAs, then compute a correlation matrix with cor() and input this into the findCorrelation() function. The function lets you specify the cutoff point. In this case we had a cutoff of .5 correlation. We also set versose to TRUE to get fuller information from the function.

Code 11.1.1 — Find Correated Variables. PimaIndiansDiabetes2.

```
library(caret)
library(mlbench)
data("PimaIndiansDiabetes2")
df <- PimaIndiansDiabetes2[complete.cases(PimaIndiansDiabetes2[]),]
corMatrix <- cor(df[,1:7])
findCorrelation(corMatrix, cutoff=0.5, verbose=TRUE)
```

```
Compare row 6  and column  4 with corr  0.664
  Means:  0.265 vs 0.187 so flagging column 6
Compare row 2  and column  5 with corr  0.581
  Means:  0.266 vs 0.161 so flagging column 2
All correlations <= 0.5
[1] 6 2
```

The output of findCorrelation() recommended that we remove column 6, mass, because it correlates with triceps. It also recommended that we remove column 2, glucose, because it correlates with insulin. This is easily done with: `df <- df[,-c(2,6)]` Recall that in this data set we had a lot of NAs for triceps. Therfore, knowing that mass correlates with triceps but mass has very few NAs might cause us to remove triceps instead of mass on the full data set.

11.1.2 Ranking Feature Importance

Of the variables we have left, which ones are most important? The caret package was used in the code below to train a model and extract variable importance. We show the output below the code and the plot in Figure 11.1.

Code 11.1.2 — Ranking Feature Importance. PimaIndiansDiabetes2.

```
ctrl <- trainControl(method="repeatedcv", repeats=5)
model <- train(diabetes~., data=df, method="knn",
     preProcess="scale", trControl=ctrl)
importance <- varImp(model, scale=FALSE)
importance
plot(importance)
```

```
ROC curve variable importance

         Importance
age          0.7432
insulin      0.7299
triceps      0.6594
pedigree     0.6215
pregnant     0.6214
pressure     0.6213
```

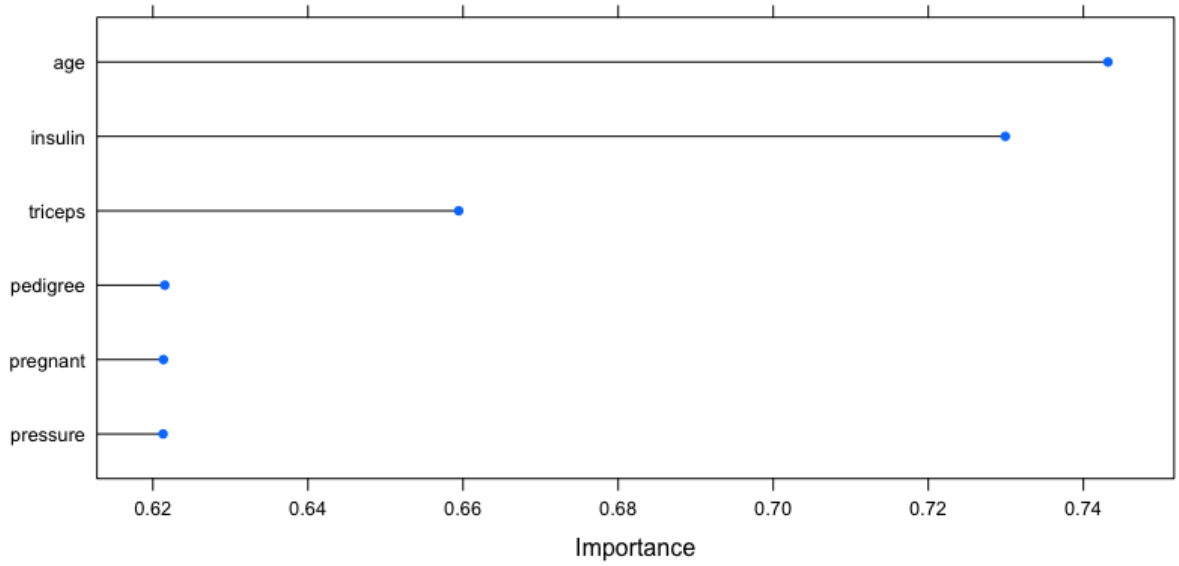

Figure 11.1: Plot of Variable Importance

11.1.3 Recursive Feature Selection

Another option in caret is recursive feature selection. This recursively eleminates features to find a subset of predictors that perform well. We start will all the original features. This gave us different advice than the findCorrelation() function above. Here we are told we can eliminate mass and pedigree but it left in both glucose and insulin which we know are highly correlated.

Code 11.1.3 — Recursive Feature Selection. PimaIndiansDiabetes2.

```
df <- PimaIndiansDiabetes2[complete.cases(PimaIndiansDiabetes2[]),]
ctrl <- rfeControl(functions=rfFuncs, method="cv", number=10)
rfe_out <- rfe(df[,1:7], df[,8], sizes=c(1:7), rfeControl=ctrl)
rfe_out
```

```
Recursive feature selection
Outer resampling method: Cross-Validated (10 fold)
Resampling performance over subset size:
 Variables  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD Selected
         1 7.177   0.5144 5.176  1.768     0.1737 1.0658
         2 7.602   0.4676 5.433  1.738     0.1501 0.9455
         3 7.358   0.4916 5.350  1.754     0.1532 0.9457
         4 7.363   0.4949 5.372  1.760     0.1562 1.0373
         5 7.416   0.4909 5.432  1.650     0.1367 0.9637
         6 7.221   0.5120 5.276  1.658     0.1413 1.0005
         7 6.999   0.5459 5.126  1.725     0.1463 0.9692        *
The top 5 variables (out of 7):
   pregnant, glucose, insulin, triceps, pressure
```

11.2 Feature Engineering

Feature engineering involves changing the data in some way to improve performance on the algorithm. Let's say we have a date field and we know that our target variable has different values for weekdays versus weekends. Some algorithms may be able to work with such data but other algorithms may perform better if we change the date field as a binary weekday/weekend factor. If we have columns in the data that are highly correlated with each other we might consider deleting the correlated variables to reduce the total number of columns fed into the algorithm.

11.2.1 Principal Components Analysis

Principal Components Analysis (PCA) is a data reduction technique that can help you reduce the dimensions of your data. PCA transforms the data into a new coordinate space while reducing the number of axes. Each axis of the reduced space represents a principal component. The first PC represents the dimension of greatest variance, and the other PCs represent decreasing variance. An intuitive explanation is to visualize n-dimensional space reduced to a k-dimensional elliptical space where k is smaller than n. This is a data reduction technique, we are losing data and may correspondingly lose accuracy in any models.

We are going to take a quick look at PCA on the iris data. In the notebook in github, we first divide the iris data into train and test, 100 and 50 observations respectively. Then the preProcess() function from caret is used to find the principal components.

Code 11.2.1 — PCA. Iris data.

```
pca_out <- preProcess(train[,1:4],method=c("center","scale","pca"))
pca_out
```

```
> pca_out
Created from 100 samples and 4 variables

Pre-processing:
  - centered (4)
  - ignored (0)
  - principal component signal extraction (4)
  - scaled (4)

PCA needed 2 components to capture 95 percent of the variance
```

We see that PCA reduced the 4 variables to 2 principal components. These two components capture 95% of the variance in the data. Let's plot the two components PC1 and PC2 and color the points according to the true class. We see that we have one class that is separate from the other two, shown in red. And we see that the blue and green are fairly well seperated but that there is some overlap. This suggests that classification will be possible but challenging.

In the online notebook, we tried kNN and got a mean accuracy of .94. Although this is lower than the 98% accuracy we got in Chapter 8 using knn on all 4 predictors, it is a confirmation that PCA is capturing something important in the data. We also ran a decision tree on the PCA data and got a little lower accuracy of .92. The tree is shown below.

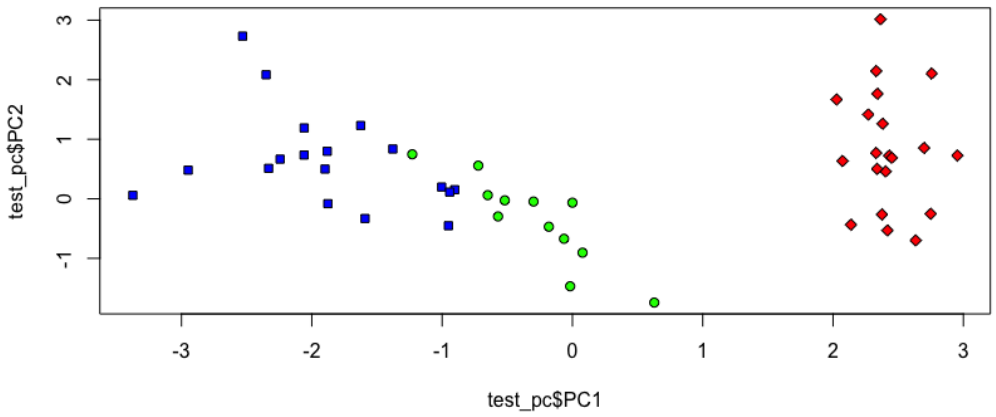

Figure 11.2: Principal Components for the Iris Data

11.2.2 Linear Discriminant Analysis

PCA is applied to data without regard to class and such is used like other unsupervised methods for data analysis. Linear Discriminant Analysis, LDA, does consider the class. LDA seeks to find a linear combination of the predictors that will maximize the separation of the classes while minimizing the within-class standard deviation. There is an lda() function in package MASS which we use on the same data as PCA above.

Code 11.2.2 — Linear Discriminant Analysis. Iris Data.

```
library(MASS)
lda1 <- lda(Species~., data=train)
lda1$means
```

```
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa         4.943333    3.350000     1.470000   0.2633333
versicolor     5.973684    2.752632     4.268421   1.3315789
virginica      6.515625    2.925000     5.493750   1.9750000
```

We see that the LDA analysis has identified means for all variables by class. When we used the lda1 model for prediction in the online notebook, we got an accuracy of 98%. Further when we plot the lda predictions by class using the two linear discriminants returned by the model, we see a nice separation. The color of the points indicates the true class while the shape indicates the predicted class.

In comparing PCA and LDA the most significant difference is that PCA is unsupervised and LDA is supervised. Both can be used for dimensionality reduction. In this example, both reduced the 4 iris predictors to two predictors. LDA will have an advantage over PCA when the class is known.

lda

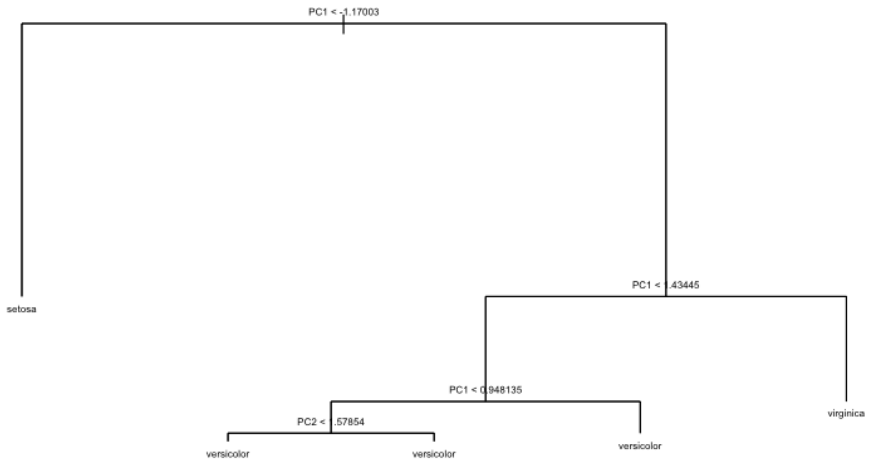

Figure 11.3: Principal Components Decision Tree

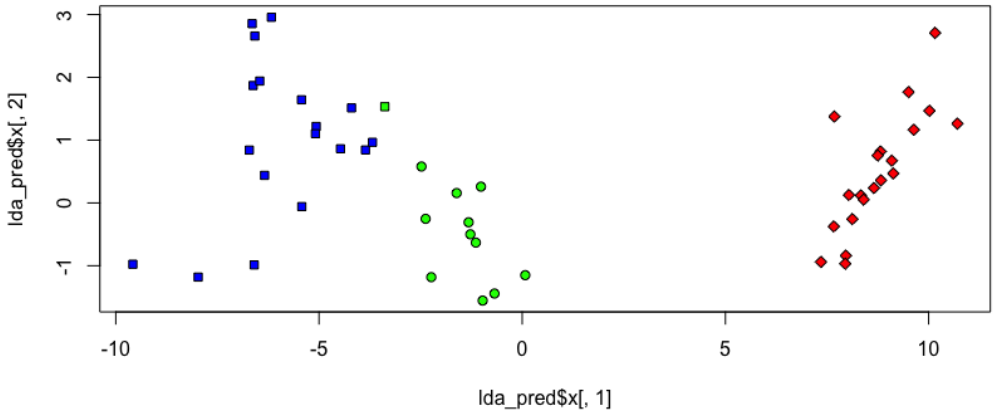

Figure 11.4: Linear Discriminant Analysis, Iris Data