Week 5 Assignment

Regis MSDS

August 7, 2020

Assignment Overview

In this assignment, we will learn some basics of using machine learning in R. We'll use a few different machine learning algorithms on binary classification data. Note that this is only one of the many ways of using machine learning. Most of the common ways of using machine learning include:

- supervised learning
  - classification: multi-class and binary
  - regression: predicting continuous values, like the temperature
- unsupervised learning
  - clustering: grouping data
  - recommenders: suggesting other items to try
- semi-supervised learning
  - usually used for classification, e.g. website labeling
- reinforcement learning
  - creating an agent to play a game or accomplish a task

Data science process / lifecycle

As was covered in the first week, there are several approaches to the data science lifecycle. This generally follows something like:

- understand the problem to be solved
- collect data
- explore and clean data (exploratory data analysis (EDA), this is often iterative)
- prepare data for modeling (can involve feature engineering and feature selection)
- carry out modeling and/or analysis (statistics)
- loop back to earlier steps as needed to collect more data, clean, and prepare it
- deployment (could be creating a report, deploying a machine learning model, or deploying a stats model like an A/B test)

Here, we will mainly focus on the EDA and machine learning (modeling) steps.

Dataset

We will be using a loan default dataset. This is a good example problem because it's easy to understand the data. Loans are given to people to buy things, and if the person can't pay back the loan, then they have defaulted. The data comes from a data science competition/hackathon.

EDA

First, we want to load the data and do some EDA. An example of loading data and plotting it for some EDA is shown below.

EDA helps us understand which variables to use in our machine learning (ML) algorithms.

EDA example

Our demo dataset is composed of samples from two multinormal distributions, meaning each distribution is a combination of multiple normal distributions (two in this case). We have two feature columns. Features are data columns we use for inputs in machine learning models. The data has one target column. Target columns are what we try to predict. The target column is based on the two multinormal distributions – 1

```
# first import necessary libraries -- remember, you can install them with install.packages(c('data.tabl
# or click the prompt at the top from the Rmd file in rstudio
library(data.table)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(yardstick)  # used for plotting a confusion matrix
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument `event_level = "second"` to alter this as needed.

##
## Attaching package: 'yardstick'

## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity
```

```
set.seed(42)  # sets the random number generator seed for more reproducible results
```

```
filepath <- '/home/nate/github/MSDS600_instructor/Week_5/'
demo.data <- fread(paste0(filepath, 'demo_data.csv'))
colnames(demo.data) <- c('feature1', 'feature2', 'target')
fwrite(demo.data, paste0(filepath, 'demo_data.csv'))

# print first few rows
head(demo.data)
```

```
##     feature1 feature2 target
## 1: 11.901526 13.55048      0
## 2:  8.494258 15.62358      0
## 3: 10.891726 16.01514      0
## 4:  7.970747 16.83320      0
## 5: 10.522082 14.48046      0
## 6: 10.923734 15.51564      0
```

```
# look at the structure of the data.table
str(demo.data)
```

```
## Classes 'data.table' and 'data.frame':   20000 obs. of  3 variables:
##  $ feature1: num  11.9 8.49 10.89 7.97 10.52 ...
##  $ feature2: num  13.6 15.6 16 16.8 14.5 ...
##  $ target  : int  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# look at a stats summary
summary(demo.data)
```

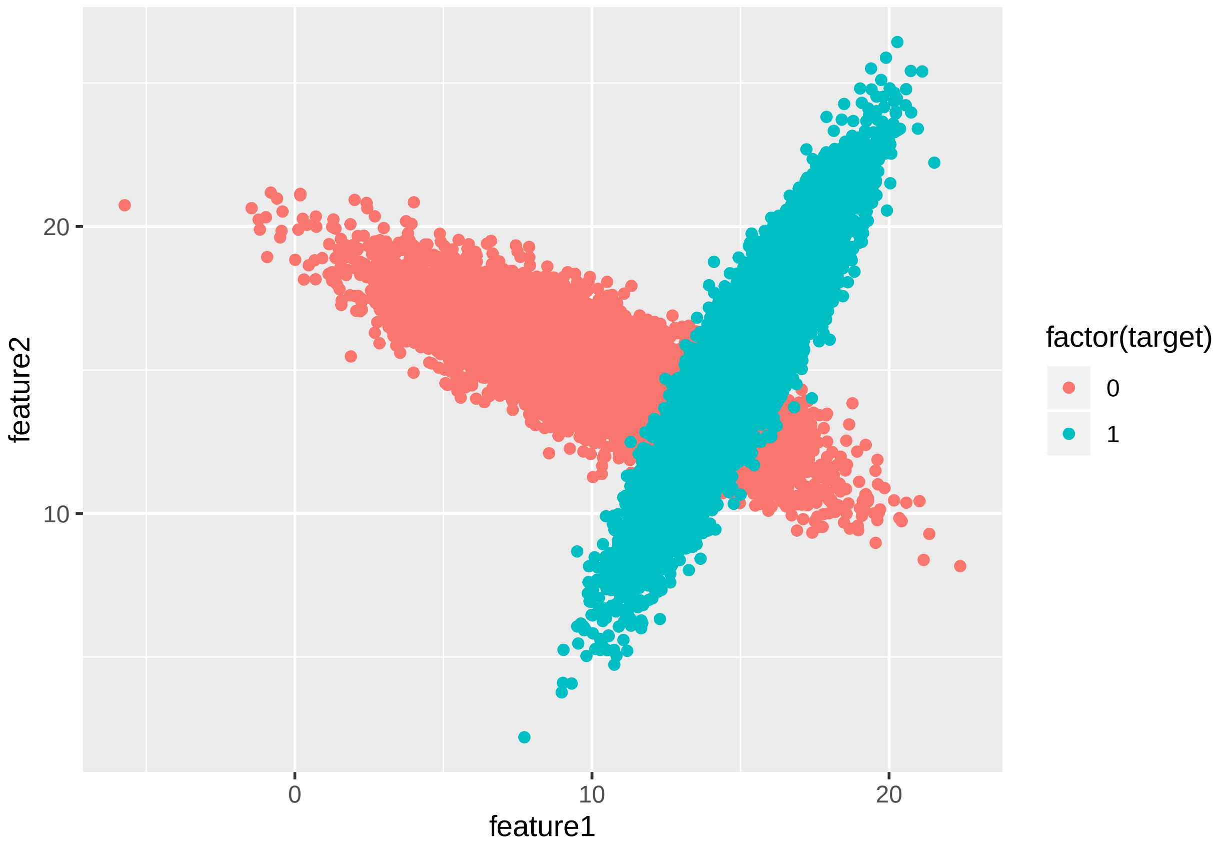
```
##     feature1         feature2          target   
##  Min.   :-5.727   Min.   : 2.205   Min.   :0.0  
##  1st Qu.:10.052   1st Qu.:13.406   1st Qu.:0.0  
##  Median :13.253   Median :14.949   Median :0.5  
##  Mean   :12.522   Mean   :14.951   Mean   :0.5  
##  3rd Qu.:15.201   3rd Qu.:16.498   3rd Qu.:1.0  
##  Max.   :22.395   Max.   :26.428   Max.   :1.0
```

```r
# get number of rows/cols
dim(demo.data)
```

```
## [1] 20000     3
```

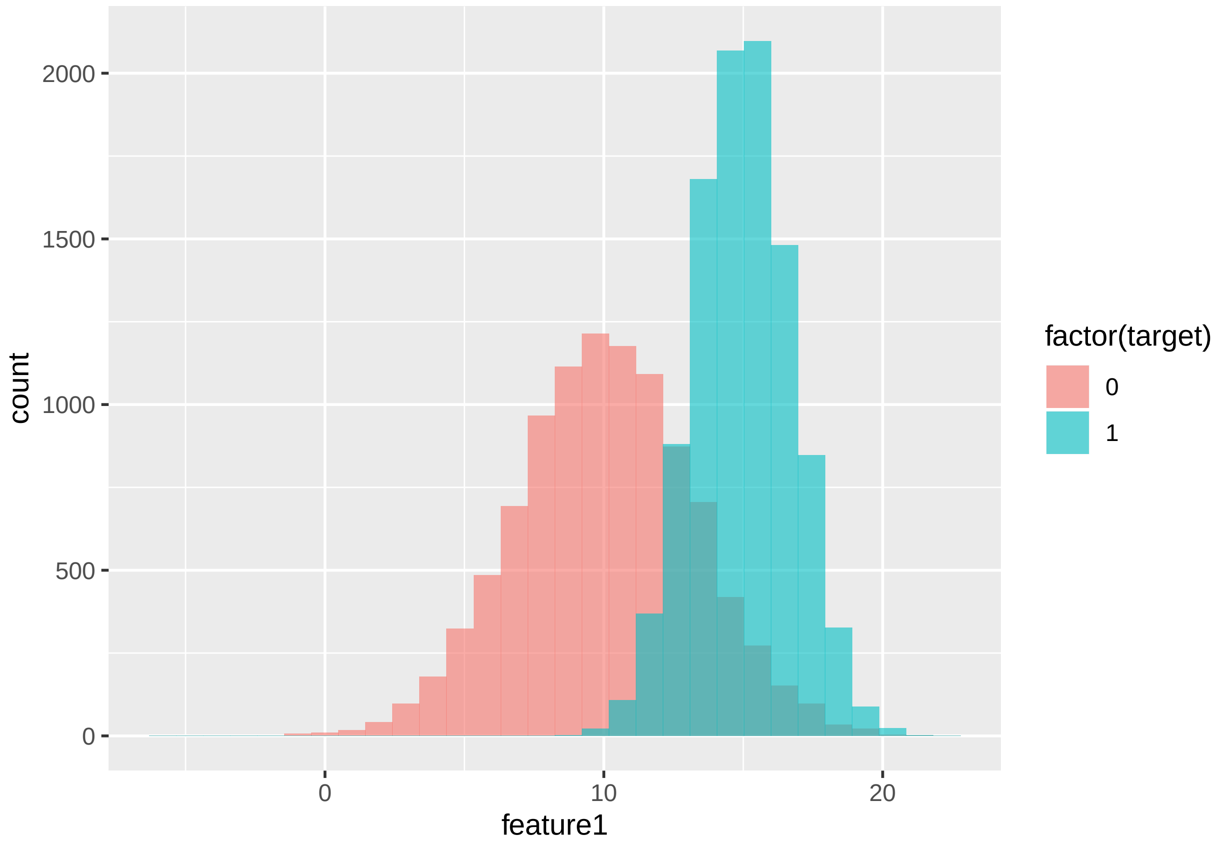
Here is one way to look at the data – a scatter plot of the two features with the targets represented as color. There are other ways to accomplish the same thing in ggplot2, but this is one of the most compact and readable code-wise. The second figure shows a histogram of one of the features, colored by the two target values. Finally, the correlation plot helps us understand the relationship between the feature and target columns.

```r
ggplot(data = demo.data, mapping = aes(feature1, feature2)) +
  geom_point(mapping = aes(color = factor(target)))
```



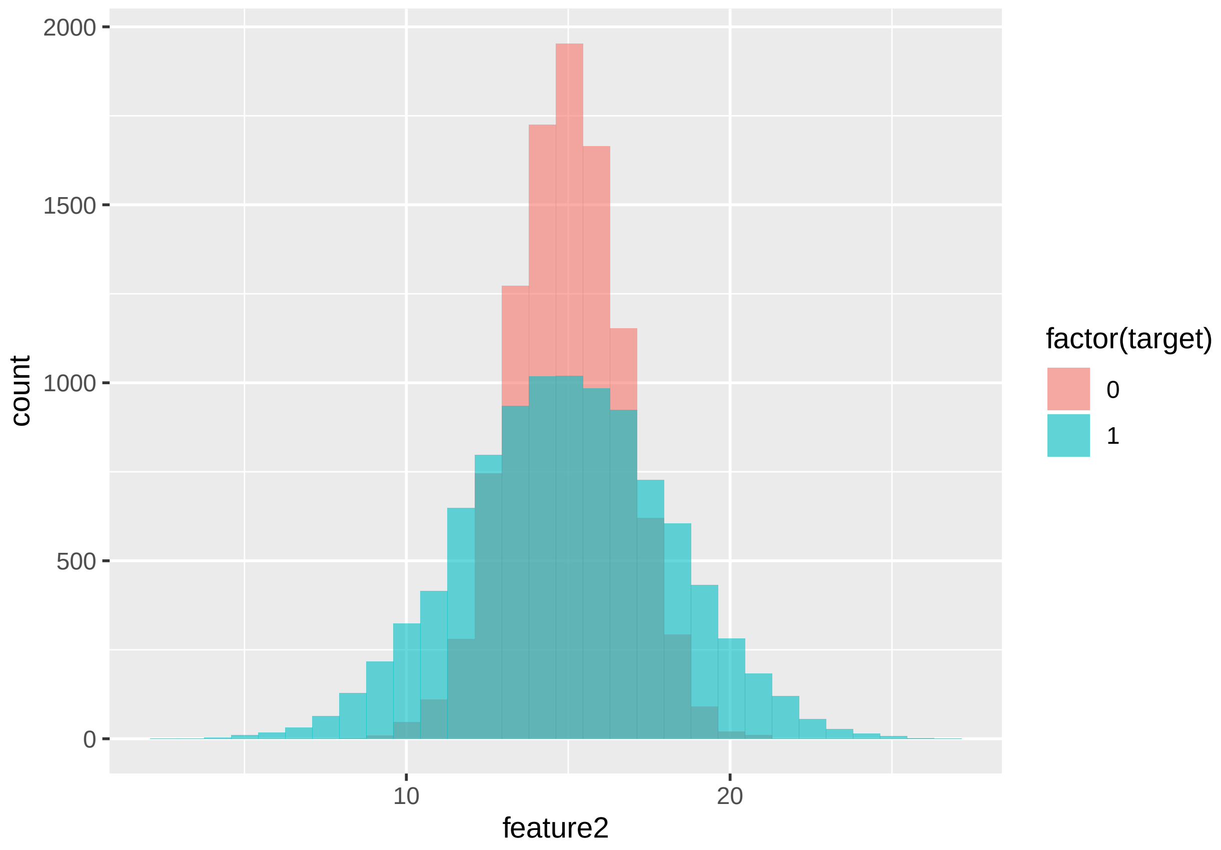
```r
# you might use r-graph-gallery for more ideas of good-looking plots you can make:
# https://www.r-graph-gallery.com/histogram_several_group.html
ggplot(data = demo.data, mapping = aes(feature1, fill=factor(target))) +
  geom_histogram(alpha = 0.6, position = 'identity')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

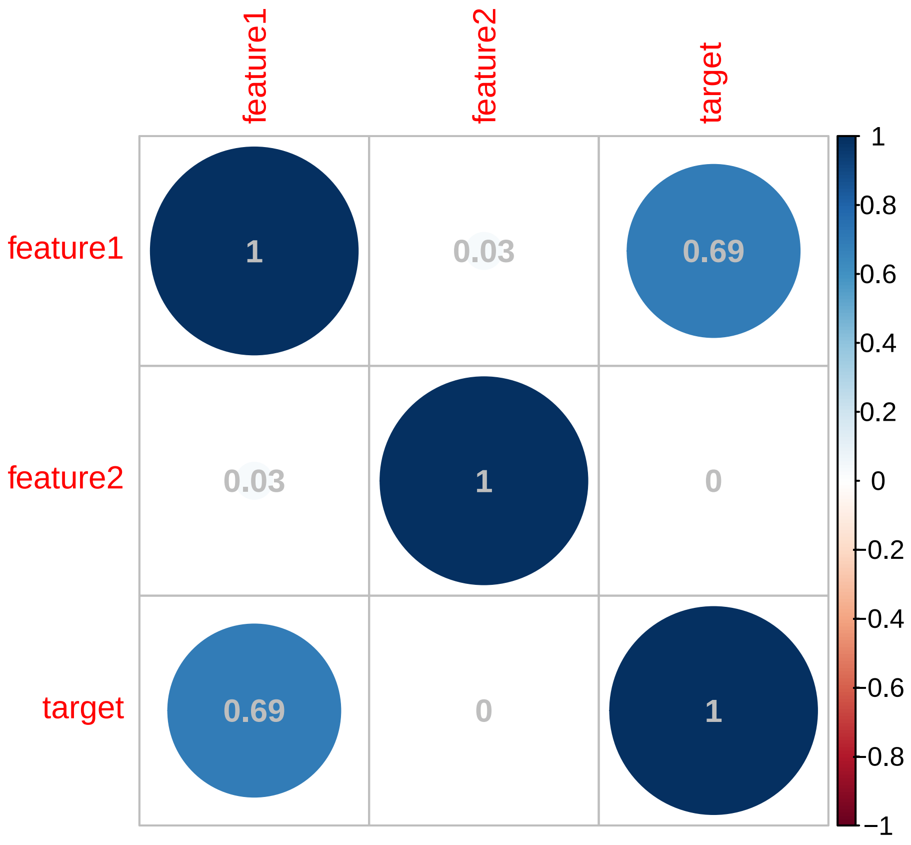


```r
ggplot(data = demo.data, mapping = aes(feature2, fill=factor(target))) +
  geom_histogram(alpha = 0.6, position = 'identity')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
corrplot(cor(demo.data), addCoef.col = "grey")
```



TO DO – EDA on the loans dataset

Carry out your EDA on the dataset here. Create at least 3 EDA plots, including a correlation plot. You might ask your instructor and classmates in the discussions for advice on which columns seem most important, but two columns you can start with are LTV (loan to value, the amount of the loan relative to the value of the vehicle purchased) and PERFORM_CNS_SCORE (credit score).

```
# load the loan_data.csv file here and utilize EDA
```

Feature selection

When using machine learning, feature selection and engineering is key. Imagine if we were trying to predict loan defaults based on a person's hair color – it probably wouldn't work very well. Instead, we look for features that correlate strongly to the target. These relationships to the target have to be measured in different ways depending on the type of data we're using.

If we have two continuous variables, like feature1 and target from the example, then we can easily use classic Pearson correlation. If the features and targets don't relate linearly (as is needed for Pearson to work), then we have to use a different correlation method like Spearman or Kendall-Tau. This is explored more in MSDS660. If the data is categorical, we might use something like mutual information score. We can also use feature importance (also called variable importance) from different machine learning methods like random forests or decision trees to understand which features most strongly relate to the target. Lastly, if two features are strongly correlated to each other, we probably only need one of them.

For now, we will only use Pearson correlation for feature selection in the example and assignment, but you are free to go above and beyond by using more advanced methods like mutual information and feature importance.

We saw from our correlation plot above that feature2 has zero Pearson correlation to the target, so we will only use feature1 in our modeling. We are also doing classification and our output variable is binary and a numeric data type, so we need to convert that to a factor. Although many R functions do a lot of 'magic' behind the scenes where these conversions happen, not all of them do.

```
demo.data[, target:=as.factor(target)]
demo.data.for.model <- demo.data[, c('feature1', 'target')]
head(demo.data.for.model)
```

```
##     feature1 target
## 1: 11.901526      0
## 2:  8.494258      0
## 3: 10.891726      0
## 4:  7.970747      0
## 5: 10.522082      0
## 6: 10.923734      0
```

TO DO: Feature selection for loans data

Machine learning

Now that we have our data prepared, we can finally get to machine learning (ML). There are dozens of ML libraries in R, but one important library is caret. This library allows us to easily use several different common ML algorithms and methods. Lets start by seeing how easy it is to run a model with caret:

```
model <- train(target ~ .,
              data = demo.data.for.model,
              method = "rpart")

model
```

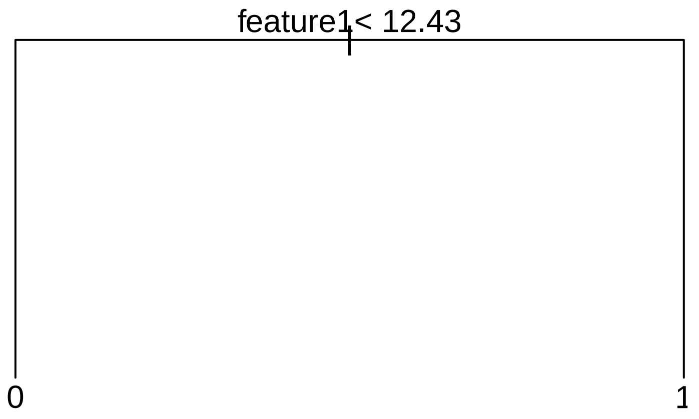
```
## CART
##
## 20000 samples
##     1 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 20000, 20000, 20000, 20000, 20000, 20000, ...
## Resampling results across tuning parameters:
##
##   cp        Accuracy   Kappa
##   0.000350  0.8486506  0.6973046
##   0.000375  0.8492545  0.6985010
##   0.701400  0.6091746  0.2225860
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.000375.
```

This is a decision tree model, called a Classification And Regression Tree (CART) from the rpart library.

The first line, `target ~ .`, defines the features and targets for the model. Remember that features are inputs to the model, and targets are outputs (what we want to predict). In our case, the target column is called 'target', and using a period simply means all other columns are used as features.

We can plot the decision tree like so (run this whole cell at once by clicking the green button on the far right of the cell). This shows us the splits in the decision tree which were learned based on the data. For a more complex problem with more variables, the tree will be deeper.

```
plot(model$finalModel, margin = 0.2)
text(model$finalModel)
```



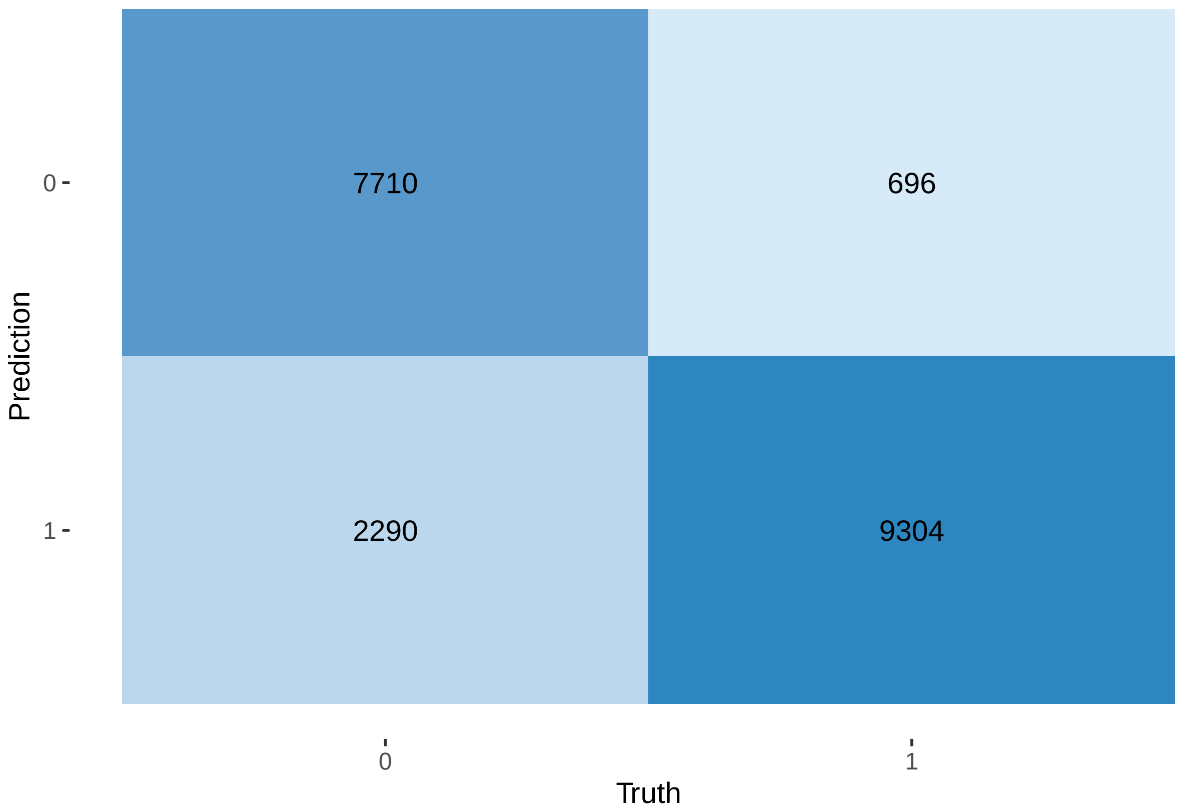
We can also get the predictions and plot them. For binary classification, one good way to plot the results is a confusion matrix (https://stackoverflow.com/a/60164777/4549682). Read more about the confusion matrix here: https://en.wikipedia.org/wiki/Confusion_matrix

```
predictions <- predict(model, demo.data.for.model[, !'target'])
# select target column as a data.table
prediction.dt <- demo.data.for.model[, c('target')]
# use in-line data.table assignment operator to make new column
prediction.dt <- prediction.dt[, predictions:=as.factor(predictions)]

cm <- conf_mat(data = prediction.dt, truth = target, estimate = predictions)

autoplot(cm, type = "heatmap") +
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1")
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



The available models in caret are searchable here: https://topepo.github.io/caret/available-models.html

Some common ML models you may want to try include: - a decision tree (CART from rpart); use `rpart` in the train() function's 'method' argument - random forests (many decision trees); `rf` - naive Bayes; `nb` - k-nearest neighbors; `knn` - support vectior machines; `svmRadial` - logistic regression; `regLogistic` - and many more…

The above list shows the type of model and the value for the 'method' argument in the train() function in backticks (').

TO DO: Machine learning on loan data

Try at least two different machine learning models on the loans data and compare the results. Simply use accuracy as a metric for comparison now, unless you want to go above and beyond and use other metrics (see the section at the end for more).

TO DO: Write a summary and analysis your work

Now that we've taken a loan dataset and predicted if people will default on their loan, write a summary and analysis of the work we performed. Aim to tell a story with the data and results.

Going above and beyond

If you want to improve your machine learning skills and make your assignment stand out, here are some other techniques and methods you can use. Or, you can just read about them to learn more about ML.

Cleaning data

Cleaning data is very important for ML models to work well. Outliers can effect some models strongly, or at least make models perform worse. Some cleaning steps we did not take, but are common include:

- replacing missing data; sometimes missing data is already represented by a 0, -1, etc, and there are no NaNs in the dataset
  - imputation can be used to replace missing data, such as mean imputation, median imputation, or ML imputation such as KNN imputation
- clipping or removing outliers; IQR can be used to detect outliers
- throwing out erroneous data; these are usually detected by outliers, but can also be data that doesn't make sense

Feature engineeing

Feature engineering can have a huge impact on model performance. Some feature engineering can include:

- mathematical transformations of data; raising data to a power (squaring or cubing it), multiplying two datasets, etc
- extracting information from datetimes, such as date spans, year, month, day, hour, etc
- simplifying features with several categories, such as only using the top 3 categories and labeling all others as 'other'

The possibities for feature engineering are huge; there are entire books dedicated to feature engineering.

Train/test splits

In order to compare models and estimate how they will perform on new data, we can break up our data into 2 pieces: train and test sets. This allows us to train our model on the train set, and evaluate performance on the test set. It's relatively easy to do in caret and the createDataPartition() function automatically keeps the proportions of the target the same in train/test, which is called 'stratification'. Similarly, the `splitstackshape` package also has a function 'stratified' which allows us to take stratified random samples of data (useful for scaling down data).

Cross-validation

Cross-validation (CV) creates train/test splits several times (k times in k-fold CV), then trains the data on the train set and evaluates on the test set. It does this many times and averages the metrics. This helps avoid any potential biases in the data that may happen from train/test splits, and results in better estimates of the best model. This is relatively easy to do it caret.

Tuning hyperparameters

Hyperparameters are the settings for ML algorithms that effect performance. Hyperparmeters are independent of the data and cannot be learned from the data (unless we are doing autoML or automatic hyperparameter optimization). For a decision tree, we can set things like the maximum number of splits (height) to limit how complex it gets. Each ML algorithm has different hyperparameters. Parameters, on the other hand, are values that are learned by the ML algorithm from the data. In a linear fit to data, these are the coefficients (the m in y = mx + b)

The caret documentation has information on hyperparameter tuning here. In the caret docs, Max Kuhn (caret's creator) decided to call hyperparameters 'parameters'. He also has some other strange design choices such as model reports labeling '0' the 'positive' class when the target is composed of 1s and 0s. The data science and statistics community generally defines parameters vs hyperparameters as we did above, so we recommend you stick with those definitions.

Other binary classification metrics

Different machine learning methods have different metrics for choosing the best model and comparing models. For binary classification, which is what we're doing in this assignment, here are some other metrics you could look into using:

- Cohen's Kappa
- ROC curves and AUC score
- a confusion matrix
- F1 score
- plot prediction probabilities by class (by setting type = 'prob' in the predict() function)

Using the full dataset

We have only used a small sample of the loans dataset. The full dataset can be found here, which has a lot more data points and features.