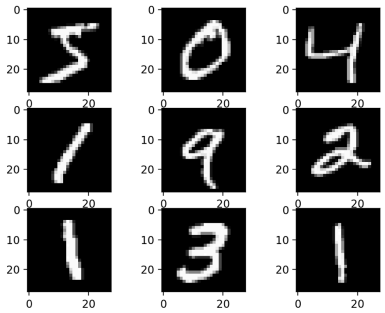
Assignment #5

Due date: 5월 21일 (화)

Handwritten Digit Recognition

MNIST 데이터셋은 image classification model의 성능을 평가하는 데 주로 활용되는 데이터셋으로, 아래 예와 같이 손으로 쓰여진 숫자들의 이미지 70,000개로 구성되어 있다. 이 중에서 60,000개는 training set으로 활용되며 10,000개는 test set으로 활용된다. 각 데이터는 28 * 28 = 784개의 픽셀의 명암을 0~255 사이의 값으로 표현한 784개의 feature와 0~9 사이의 숫자로 표현되는 target을 포함한다. 본 과제에서는 tree를 활용하여 숫자를 분류하기 위한 classification model을 만들어본다.



1. 아래의 순서에 따라 data preprocessing을 수행하자.

   A. **dslabs** 패키지를 설치하고, 다음 코드를 실행하면 mnist 변수에 아래 설명과 같이 데이터가 저장된다.

   ```
   mnist <- read_mnist()
   ```

   - **mnist$train$images**: Training set의 feature 데이터 (행렬)
   - **mnist$train$labels**: Training set의 Target 데이터 (벡터)
   - **mnist$test$images**: Test set의 feature 데이터 (행렬)
   - **mnist$test$labels**: Test set의 Target 데이터 (벡터)

   B. Training set의 데이터 사이즈가 매우 크기 때문에 60,000개의 데이터 중에 처음 2,000개만 사용하자. 이때 feature 데이터는 변수 **train_x**에 저장하고, target 데이터는 변수 **train_y**에 저장한다. **train_y**의 분포를 확인해 보자.

   C. **train_x**의 column의 이름을 V1, V2, V3 … 순서대로 설정하자. **colnames**() 함수를 사용하여 column의 이름을 수정할 수 있다.

   D. 784개의 픽셀 중에서 숫자와 관련없는 가장자리 부분과 같은 경우는 많은 데이터들에 대해서 같은 색을 가진다. 이러한 픽셀은 숫자를 분류하는 데 크게 영향을 미치지 않으므로 feature에서 제외시키는 것이 합리적이다. caret 패키지의 **nearZeroVar**(train_x) 함수를 실행하면 **train_x**의 column들 중에서 variance가 0이거나 0에 가까운 것들의 index를 얻을 수 있다. 이 index에 해당하는 column을 **train_x**에서 제외시키자. 784개의 feature 중에서 몇 개가 제외되었는가?

   E. 최종적으로 **train_x**와 **train_y**를 합쳐서 **train**이라는 이름의 데이터프레임을 만들자.

   F. C~E의 과정을 test set에 대해서 동일하게 수행하여 **test**라는 이름의 데이터프레임을 만들자. 이때 D에서 제외한 feature와 동일한 feature들을 test set에서도 제외시켜야 한다.

2. 아래의 코드는 test set의 첫번째 데이터를 화면에 이미지로 출력해준다. 이를 활용하여 test set의 image 행렬의 행 번호를 입력받아 숫자 이미지를 출력하는 함수 **print_image()**를 만들어보자. 이 함수를 활용하여 test set 중에서 이미지로부터 실제 숫자값을 유추하기 어려운 예를 몇 개 찾아보자.

```
image(1:28, 1:28, matrix(mnist$test$images[1,], nrow=28)[ , 28:1], col =
gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```

3. Decision tree를 만들어보자.

   A. **rpart()** 함수의 default 옵션으로 Tree를 만든 후 cross validation을 활용한 pruning 과정을 수행해보자.

   B. Pruning을 통해 얻은 Tree의 **Test set에 대한 정확도**는 얼마인가?

   C. **randomForest()** 함수를 사용하여 **bagging model**을 만들어보자. mtry를 제외한 옵션은 모두 default 값을 사용한다.

   D. Bagging model의 **Test set에 대한 정확도**는 얼마인가? B번의 Tree model에 비해서 성능이 얼마나 향상되었는가?

   E. **randomForest()** 함수의 default 옵션을 사용하여 **random forest model**을 만들어보자. 그리고 Bagging과 random forest 모델의 Tree의 수의 증가에 따른 OOB classification error rate의 변화를 하나의 그래프에 그려보고 두 모델의 성능을 비교해보자.

   F. Random forest model의 **Test set에 대한 예측 정확도**는 얼마인가? Bagging model에 비해서 성능이 얼마나 향상되었는가?

   G. Random forest model의 결과로부터, 분류가 가장 정확한 숫자는 몇인가? 가장 분류가 어려운 숫자는 몇인가?

   H. 실제 값은 7지만 Random forest model에 의해 1로 예측되는 test data를 찾아 이미지를 몇 개 출력해보자. 눈으로 확인했을 때 7와 1의 구별이 어려운가?

* **잘 정리된 report**를 제출해야 하며, 최종적으로 아래의 두 파일만 제출합니다 (압축파일 제출금지)

- R markdown 파일
- 생성된 html 파일 (실행코드, comment, 결과분석 포함)