Applied Data Science: Midterm Project

Subject Matter

The project will focus on an image recognition problem. You will construct a variety of machine learning models with the goal of generating predictive classifications.

Details

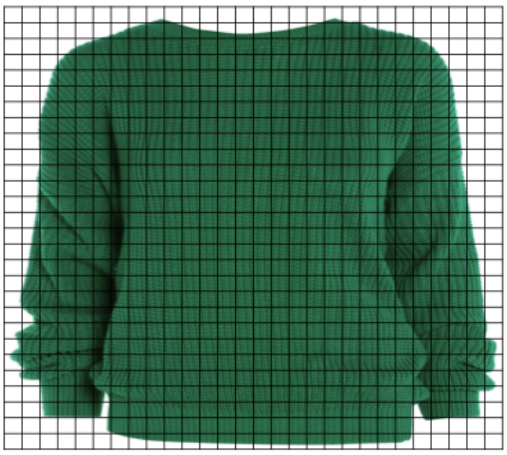
- **Group Project**: This project will be completed in groups. You may use any publicly available written resources to help, but please cite your sources.
- **Materials to Submit**: You must write a report in RMarkdown format. Please provide the code file (in Rmd format) and the output file (HTML preferred). We are supplying a template file to help you get started. The final report should be of moderate length (roughly 2000-3000 words, not strictly enforced). Include all of the code you used, and display any relevant tables, plots, or other supplementary materials.

In your report, you need to give a detailed explanation of each step you take to arrive at your solution. Please give a justification or explanation of the results you obtain. The reports should also include the lessons you may have learned from doing the project.

The Data

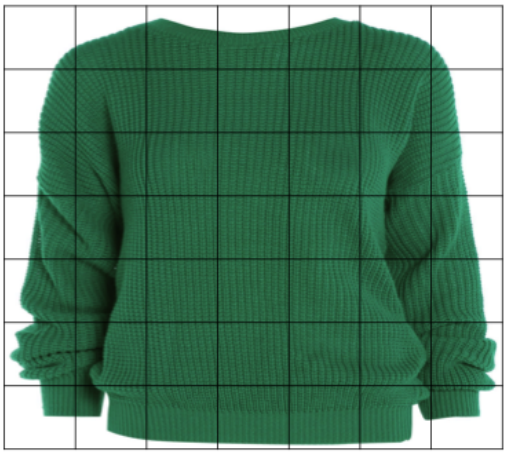
The MNIST Fashion database (https://github.com/zalandoresearch/fashion-mnist) collected a large number of images for different types of apparel. Each image is divided into small squares called **pixels** of equal area. Within each pixel, a brightness measurement was recorded in grayscale. The brightness values range from 0 (white) to 255 (black). The original data set divided each image into 784 (28 by 28) pixels. As an example, one such image would be divided as follows:

A 28-by-28 Pixel Image



To facilitate more tractable computations, we have condensed these data into 49 pixels (7 by 7) per image. An example of dividing an image is shown below:

A 7-by-7 Pixel Image



For each picture, the first 7 pixels represent the top row, the next 7 pixels form the second row, etc.

The assignment provides the following files:

- **Training Set:** MNIST-fashion training set-49.csv. This file contains 60,000 rows of data.
- **Testing Set:** MNIST-fashion testing set-49.csv. This file contains 10,000 rows of data.

Each file includes the following columns:

- **label**: This provides the type of fashionable product shown in the image.

- **pixel1, pixel2, ..., pixel49**: These columns provide the grayscale measurement for the 49 pixels of the image.

A Practical Machine Learning Challenge

What are the best machine learning models for classifying the labels of the testing set based upon the data of the training set? How small of a sample size do you need to generate the "best" predictions? How long does the computer need to run to obtain good results? To balance these competing goals, we will introduce an overall scoring function for the quality of a classification.

Points = 0.25 * A + 0.25 * B + 0.5 * C

where

A is the proportion of the training rows that is utilized in the model. For instance, if you use 30,000 of the 60,000 rows, then A = 30,000 / 60,000 = 0.5;

B = $\min\left(1, \frac{X}{60}\right)$, where X is the running time of the selected algorithm in seconds. Algorithms that take at least 1 minute to run will have the value **B = 1**, which incurs the full run-time penalty.

C is the proportion of the predictions on the testing set that are incorrectly classified. For instance, if 1000 of the 10000 rows are incorrectly classified, then C = 1000 / 10000 = 0.1.

You will create and evaluate different machine learning models on different sample sizes. The quality of different combinations of the models and the sample sizes can be compared based on their Points. The overall goal is to build a classification method that **minimizes the value** of **Points**. In this setting, the ideal algorithm would use as little data as possible, implement the computation as quickly as possible, and accurately classify as many items in the testing set as possible. In practice, there are likely to be trade-offs. Part of the challenge will be adapting to the rules of the game to improve your score!

The Rules

- You may select **3 different sample sizes** to work with.
- For each selected sample size, you will generate 3 separate **model development** sets by sampling from the rows of the overall training data randomly without replacement. (You may use the **sample** function in R.) If the full sample size of the training data is selected, then please select your three model development sets by drawing randomly with replacement from the full training data set. As an example, you may consider using the following table names to create the 9 model development data sets:

Show 10 entries Search:

Sample Size	First Random Sample	Second Random Sample	Third Random Sample
500	dat_500_1	dat_500_2	dat_500_3
1000	dat_1000_1	dat_1000_2	dat_1000_3
2000	dat_2000_1	dat_2000_2	dat_2000_3

Showing 1 to 3 of 3 entries

Then, on the 9 model development data sets, you will conduct the following work:

- You will build 10 different classification models using machine learning techniques **on each of the 9 model development data sets**. The selected methods must include at least 5 of the following techniques:

1. Multinomial logistic regression (Package: nnet; Function: multinom)
2. K-Nearest Neighbors (Package: class; Function: knn)
3. Classification Tree (Package: rpart; Function: rpart)
4. Random Forest (Package: randomForest; Function: randomForest)
5. Ridge Regression (Package: glmnet; Function: glmnet with alpha = 0)
6. Lasso Regression (Package: glmnet; Function: glmnet with alpha = 1)
7. Support Vector Machines (Package: e1071; Function: svm)
8. Generalized Boosted Regression Models (Package: gbm; Function: gbm or Package: xgboost; Function: xgboost)
9. Neural Networks (Package: nnet; Function: nnet)

Many of these techniques may also be implemented using other libraries such as **caret**. The requirement of 5 separate modeling techniques applies to the choice of methods; it does not constrain your choice of which packages are used to implement the methods.

- Multiple variants of the same technique (e.g. K-Nearest Neighbors with k = 5 and k = 10) would each count separately toward your 10 models. The variants must have different parameter settings to be considered as distinct models.
- Any other machine learning technique is also allowed.
- At least one model must be an ensembling model that combines the results of some of the other models you have built. For this model, it may not be constructed from the original data. Instead, the **predictions from the other selected models** will be the inputs. This can be as simple as averaging the predictions from two or more other algorithms. You are also welcome to use these inputs in building a more complex model.
- For each of the 9 model development sets, you will fit all 10 of the selected classification models. This means that a total of **90 separate models** will be considered.
- For each of the 90 fitted models, the sample size proportion **A**, the running time score **B** and the misclassification rate **C** will be computed and recorded.

As an example, with selected sample sizes of 500, 1000, and 2000, the results of the 90 models may be recorded as follows:

Show 10 entries Search:

Model	Sample Size	Data	A	B	C	Points
Model 1	500	dat_500_1				
Model 1	500	dat_500_2				
Model 1	500	dat_500_3				

Model	Sample Size	Data	A	B	C	Points
Model 1	1000	dat_1000_1				
Model 1	1000	dat_1000_2				
Model 1	1000	dat_1000_3				
Model 1	2000	dat_2000_1				
Model 1	2000	dat_2000_2				
Model 1	2000	dat_2000_3				
Model 2	500	dat_500_1				

Showing 1 to 10 of 81 entries Previous 1 2 3 4 5 ... 9 Next

- We will evaluate the results of a **model at a selected sample size** by averaging the values of **A, B, C, and Points** across the 3 randomly sampled data sets. To do this, compute the mean of these quantities grouped by the Model and Sample Size.
- Then you will report an overall scoreboard of your average results for the 30 combinations of Model and Sample Size:

Show 10 entries Search:

Model	Sample Size	A	B	C	Points
Model 1	500				
Model 1	1000				
Model 1	2000				
Model 2	500				
Model 2	1000				
Model 2	2000				
Model 3	500				
Model 3	1000				
Model 3	2000				
Model 4	500				

Showing 1 to 10 of 30 entries Previous 1 2 3 Next

- The values of A, B, C, and Points should all be rounded to **4** decimal places.
- The values on the scoreboard should be **sorted in increasing order of the Points** scored.

The Report

Each of the 10 categories of models you construct should be summarized in a separate section. The overview should include a few paragraphs of written content along with the programming code that was used. Show the code that you use to fit the model to the training data, generate predictions on the testing set, and evaluate the accuracy of the results. Provide any details about the selected parameters. Explain why you selected this technique, a few details about how it works, and a short discussion of its advantages and disadvantages.

The report should be written in the style of a conference paper. It should include an Introduction section and conclude with a Discussion section. Briefly explain the results of the models and what conclusions you can draw from the findings. You might also explore some related questions: How did the choice of the Points function impact the results? What would happen if we gave more weight to the sample size component **A** or the running time factor **B** instead of more weight to the accuracy **C**? What would you do if you had the computing resources to explore a wider variety of models and sample sizes? Are you concerned about the possibility of fitting too many models? Or feel free to propose and address some other questions.

Grading

The report will be graded based on a variety of factors:

- **The Quality of the Implementation**: Generating code that builds conceptually sound models without errors.
- **Coding Practices**: Writing code with a good style.
- **Reproducibility**: Generating results in a manner that can be easily understood and reproduced by others.
- **Playing by the Rules**: Creating solutions that fit the specified requirements.
- **Quality of Communication**: How well the methods and results are explained.

We will not specifically grade the reports based on the Points scored by the methods. However, some reasonable standard of quality should be met.

It is more important to obtain high-quality results within the rules we specified and to provide clear descriptions of your work. However, we will not assign additional credit for exhaustive exploration, extra pages, or a larger number of models. You may feel free to consider these possibilities in your own time, but the report should focus only on the 10 selected models and the 3 selected sample sizes you choose.

Because this is a group project, we will also implement the following requirements:

- Each team must submit a **Division of Responsibilities** form. For each section of the report, the team will state in percentage terms how each member contributed to the development of the section.
- Each team member must have the primary responsibility (at least a 50% stake) for the development of **at least 2 of the 10 models**.
- Each individual must submit a **Summary of Collaboration** form. Each student will separately provide a short report on the team's collaboration and the work of each other team member. This form will be kept confidential.

Computational Hints

The sample size and number of features in the data can create computational challenges. It is not unreasonable for some models to run for a few minutes or more. Some techniques require greater computational resources than others. For these circumstances, it can be a good practice to gradually build up your models. Start with a relatively small sample size (e.g. 100-500 randomly selected rows). Test out the methodology to make sure that the smaller model is working properly. You can use the **Sys.time()** function to record the time before and after a model is run. See how much time elapses at different sample sizes. Try to get an idea of how long the model will take to run when you provide the intended sample size. It is also fair to select models that are easier to implement if the computations prove to be burdensome – or simply to work with smaller sample sizes. Discuss how the practical challenges of implementing these methods can create a barrier to using the techniques that generate the best predictions – and how you might decide on the best compromise for the circumstances.

We recommend that you organize your data with meaningful names that will help streamline your efforts. The 9 selected training sets include the combinations of 3 selected sample sizes and 3 iterations of randomized sampling. Using a naming structure like **dat_n_k** to represet the training set at sample size **n** with the **k**th sample will help you stay organized and present your work. Another option would be to create one aggregated data set by binding the rows of the 9 separate training sets. With this option, it would help to include columns **n** and **k** that would identify the sample size and iteration. As an example, if sample sizes of **500, 1000, and 2000** were selected, then the aggregated data set would have **3 * (500 + 1000 + 2000) = 10500** rows. Then, for each combination of **n** and **k**, a model can be fit on the subset of this aggregated data set corresponding to a unique pair of **n** and **k**.

One tricky aspect of generating predictions is understanding the workings of R's **predict** function. Each modeling technique has its own associated prediction method. For instance, a Random Forest model from the **randomForest** package has a prediction method called **predict.randomForest**. It is worthwhile to study the help files for these methods, as sometimes the parameters to supply have some variations in their names or forms. Then, it is also worthwhile to examine the output of the predictions to ensure that they match your expectations. Usually supplying the parameter **type = "response"** in the prediction will lead to the correct output. Other types could lead to output that is not of the correct form to evaluate the method's predictions.

Because this is a group project, we recommend that you divide the work in reasonable ways. Each member of the team can take responsibility for generating a number of models. The overall steps of iteration, scoring, summarization, and reporting can also be divided up. Before fitting the models, it would make sense to build a plan for how the computations for each model can feed into a unified structure.

Because you will be fitting the same model on 9 separate data sets, we recommend that you develop functions that will simplify your work. This will also improve the reproducibility and readability of your code. This could include the following steps:

- For each of the 10 models, write a customized function that will fit that model on a specific data set (e.g. **dat_n_k** as described above), generate predictions on the testing set, and report a summary of information to be included on the scoreboard.
- Then, for any single one of the 10 models, it would help to write a function that fits that model to **each of the 9 training sets**. This could be as simple as using a for loop to iterate through the 9 training sets and calling the customized function for that model described in the previous point.

- For each model, each of the 9 training sets will lead to its own summary of information for the scoreboard. The results of these trials can then be aggregated into a table.
- This table of 9 scoring results will then be further summarized into 3 rows of information, one for each selected sample size, to describe the overall performance of the model.

With all of this in mind, it should be possible to have all of the work proceed in a relatively small number of processing functions:

- **Sampling**: This could be performed in 1 function.
- **Modeling Functions**: This could up to 1 per model for a total of 10. (It could be fewer if the same procedure is fit with different parameters.)
- **Iteration Function**: This would be 1 function that runs through the 9 iterations of a single model.
- **Scoring Function**: This could be 1 function to compute the scoreboard results for a model at a given sample size.
- **Scoring Summary Function**: This could be 1 function that aggregates the 90 modeling results into 30 rows of information, one for each unique pair of a model and a sample size.
- **Reporting Function**: This would appropriately format and display the scoreboard.

Meanwhile, it should be emphasized that running 90 models is **likely to be time consuming**. Re-running a report in RMarkdown could require substantial time. For this reason, we recommend **saving** your models using one code chunk and then separately **loading** the results in a second code chunk. By doing this, you can set **eval = FALSE** in the completed chunks once you are satisfied with the development of the models. Individual models can be saved in a separate file using either the **saveRDS** command (recommended) or with the **save** command. Then, in the second chunk, the models can be loaded using either **readRDS** (recommended) or with **load**.

Collaboration Policy

For this project, the same collaboration policy that is used for the homework assignments will apply. You are welcome to work with the teaching staff and other students (including those outside of your group) to better understand the material and the associated challenges. However, your team must produce its own original work, which means writing your own code and report. Any outside materials should be cited as references.