COMP4318 & 5318 - Machine Learning and Data Mining: Assignment 1

Due: Sunday Week 7 - Sep 15th, 2024 11:59PM

1. Summary

In this assignment, you are tasked with the challenge of developing machine learning (ML) classifiers capable of categorizing grayscale images into predefined classes. Your task involves employing various classification algorithms to identify which is most effective and efficient in processing image data. Additionally, you are required to document your methodologies and findings in a detailed report. The total score for this assignment is allocated as follows:

1. Code: max 65 points
2. Report: max 35 points

Detailed about assignment specifcations and scoring criteria can be found in the assignment page on Canvas (Assignments $\rightarrow$ Assignment 1 - Specification). The sections below provide comprehensive information on the assignment tasks and guidelines for submission.

2. Dataset description

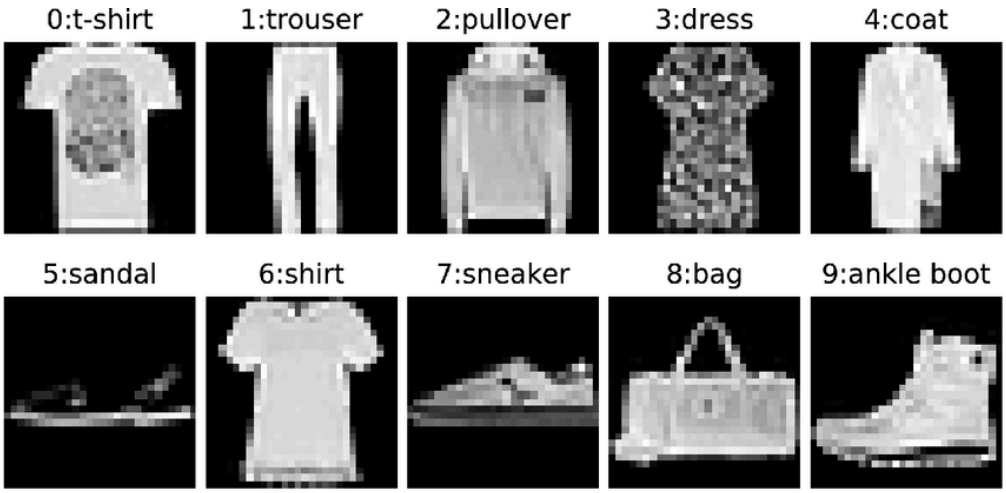
The dataset for this assignment is derived from the Fashion-MNIST dataset, a collection of fashion articles represented as grayscale images. This dataset can be downloaded from Canvas. The dataset consists of a training set of 30,000 examples and a test set of 5,000 examples. They belong to 10 different categories. The validation set is not provided, but you can randomly pick a subset of the training set for validation. Your prediction over the test set must be submitted to Kaggle to receive the public accuracy over first 2,000 examples, you will use this accuracy to analyse the performance of your proposed method. It is NOT allowed to use any examples from the test set for training; or it will be considered as cheating. The rest 3,000 labels of the test set are reserved for marking purpose.

Data samples are categorized into the following ten classes:

- 0: T-shirt/Top
- 1: Trouser
- 2: Pullover
- 3: Dress

- 4: Coat
- 5: Sandal
- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

Below are visual examples of the dataset, showcasing samples from each category:



The dataset can be downloaded from the Assignment 1 page on Canvas. Note that only a subset of the original Fashion-MNIST dataset is provided for this assignment. You must use the specific files supplied in the assignment materials for training and testing.

3. Data Preparation

The required `data` files are in the data folder, downloadable as a zip from the Assignment 1 - Specification page on Canvas. Extract the files into your working directory. The folder includes:

- `train.csv` : 30,000 labeled samples for training, evaluation, and model selection.
- `test1.csv` : 2,000 labeled samples for model efficiency testing (Canvas submission).
- `test2.csv` : 5,000 unlabeled samples for Kaggle evaluation.
- `sample.csv` : A sample prediction file format for Kaggle submission (`test_output.csv`).

Use Python's pandas library to load these CSV files into DataFrames, ensuring they are under the `./data/` directory.

3.1 Loading data

Use the following Python code to load the training data:

```
import pandas as pd
import os
print(os.listdir("./data"))
pd.set_option('display.max_columns', 10)
```

```
['sample.csv', 'test.csv', 'train.csv']
```

```
# train.csv including feature and label using for training model.
data_train_df = pd.read_csv('./data/train.csv')
```

```
# print out the first 5 rows of the training dataframe
data_train_df.head()
```

	id	v1	v2	v3	v4	...	v781	v782	v783	v784	label
0	1	0	0	0	0	...	0	0	0	0	6
1	2	0	0	0	0	...	0	0	0	0	7
2	3	0	0	0	0	...	0	0	0	0	4
3	4	0	0	0	0	...	0	0	0	0	7
4	5	0	0	0	0	...	0	0	0	0	3

5 rows × 786 columns

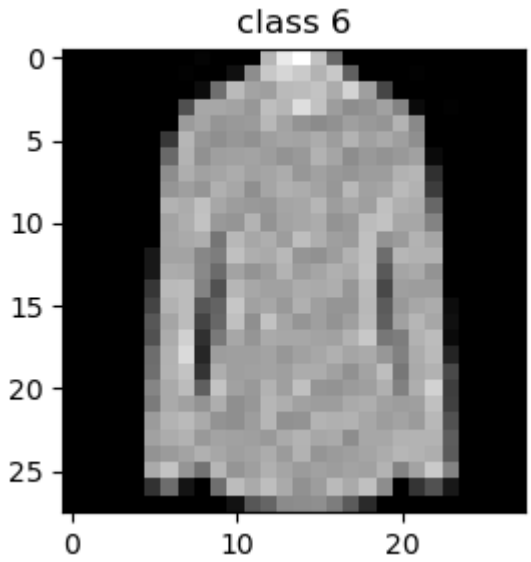
Then data would be a dataframe with 30,000 samples including 784 features (from v1 to v784) and its label.

```
# Selecting input feature
data_train_feature = data_train_df.loc[:, "v1":"v784"].to_numpy()

# Selecting output lable
data_train_label = data_train_df.label.to_numpy()
```

Showing a sample data. The first example belongs to class 2: Pullover

```
import matplotlib.pyplot as plt
data_train_feature = data_train_feature.reshape((data_train_feature.shape[0
plt.figure(figsize=(3,3))
plt.imshow(data_train_feature[0], cmap=plt.get_cmap('gray'))
plt.title("class " + str(data_train_label[0]))
plt.show()
```



4. Task Description

4.1 Code

You will now explore and build diffferent ML models for the given dataset. You are required to implement at least **FOUR** models, which should include THREE from the following methods:

1. Nearest Neighbor
2. Logistic Regression
3. Naïve Bayes
4. Decision Tree
5. SVM

and ONE of these ensemble methods:

1. Bagging
2. Boosting
3. Random forest

For these implementations, you may use established packages and libraries that have been introduced in the tutorials such as sklearn or create your own custom solutions from scratch.

Your code must be easily readable and well commented. The following are expected to be satisfied:

- **Readability & Consistency**: Easy to read, and consistent in style
- **Coding Descriptions & Comments**: Descriptions and comments clarify meaning where needed

- **Robustness**: Handles erroneous or unexpected input

It should follow the structure below.

4.1.1 Environment Setup

Install and import necessary packages and libraries used in your coding environment. It is recommended to specify their versions to ensure reproducibility.

```
# TODO: Install and import necessary libraries
```

Define any necessary utility or helper functions (e.g., for plotting, optimization, etc.) if applicable.

```
# TODO: Define helper function (e.g. plotting) if applicable
```

4.1.2 Data Preprocessing

Implement at least ONE preprocessing technique on the dataset before model training. Possible methods include **Normalization**, **Dimensionality Reduction**, etc.

```
# TODO: Implement Preprocessing Techniques
```

4.1.3 Model 1

Implementation

Implement the initial version of your model using a set of predefined hyperparameters. This will establish a baseline from which improvements can be made.

```
# TODO: Implement model 1
```

Hyper-parameters Tuning

Enhance your model by fine-tuning its hyperparameters. Use techniques such as grid search combined with k-fold cross-validation to systematically identify the optimal parameter set.

```
# TODO: Fine-tune the hyperparameters of model 1
```

4.1.4 Model 2

Implement and fine-tune the hyperparameters for Model 2 (using the same approach as Model 1).

```
# TODO: Implement model 2
```

```
# TODO: Fine-tune the hyperparameters for model 2
```

4.1.5 Model 3

Implement and fine-tune the hyperparameters for Model 3.

```
# TODO: Implement model 3
```

```
# TODO: Fine-tune the hyperparameters for model 3
```

4.1.6 Model 4

Implement and fine-tune the hyperparameters for Model 4.

```
# TODO: Implement model 4
```

```
# TODO: Fine-tune the hyperparameters for model 4
```

4.1.7 Evaluation

Evaluate the best version of each model using appropriate classification performance metrics on the validation set and test on `test1.csv`. Ensure that the results are visualized using high-quality plots, figures, or tables to clearly demonstrate model performance.

```
# TODO: Evaluate each model
```

4.1.8 Comparison

Compare all classifiers with their optimized hyper-parameters, focusing on criteria such as classification performance, training time, and inference time. Visualization of these comparisons is required; use high-quality plots, figures, or tables to facilitate a clear understanding of the differences and strengths of each model.

```
# TODO: Compare performance of all models
```

4.1.9 The Best Classifier

Conclude the best classifier

```
# TODO: Train and test the classifier which has the best performance
```

4.1.10 Loading testing data

Load the testing data for prediction

```
# test2.csv includes 5000 samples used for label prediction. Test samples d
data_test_df = pd.read_csv('./data/test2.csv', index_col=0)
```

```
# print out the first 5 rows of the test dataframe
data_test_df.head()
```

id	v1	v2	v3	v4	v5	...	v780	v781	v782	v783	v784
1	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	...	0	0	0	0	0
4	0	0	1	0	2	...	0	0	0	0	0
5	0	0	0	0	0	...	0	0	0	0	0

5 rows × 784 columns

Use the your best classifier to make predictions for the test data. The predictions should be stored in a vector named `output`, with a length of 5000.

```
# TODO: Use your best classifier to make predictions on unseen data. The ou
```

Save your prediction vector as a `test_output.csv` file, which contains two columns: `id` and `label`. Please refer to the `example_output.csv` for the structure of this output file.

```
output_df = pd.DataFrame(output, columns = ['label'])
output_df.to_csv('./test_output.csv', sep=",", float_format='%d',index_labe
```

4.2 Report

The report must be structured into the following key sections:

1. **Introduction**: Provide a comprehensive overview of the dataset, outline the methods chosen, and summarize the key findings and results.
2. **Methodology**: Describe pre-processing techniques and ML algorithms employed in this assignment. Include a discussion of the theoretical principles underlying each method and explain the rationale behind your choices.
3. **Result and Discussion**: Detail the experimental settings (e.g., implementation strategies, hyperparameter finetuning strategies, etc.). Present the results obtained from the selected algorithms and discuss their implications. Compare the performance of all models, considering factors such as accuracy, model complexity, training time, and inference time. Employ high-quality plots, figures, and tables to visually support and enhance the discussion of these results.
4. **Conclusion**: Summarize your main findings, mention any limitations methods and results and suggest potential directions for future works.
5. **References**: include the references cited in your report in a consistent format.

Important Notes

- The maximum length of the main report is 8 pages (excluding appendix and references).
- You must include an appendix that clearly provides the instructions on how to setup the environment to run your code, especialy the installation guide and version of any external packages and libraries used for implementation. In addition, you should include the hardware configurations used for the coding environment.
- The report must be in PDF format. Make sure the report is well-structured, easy to read, and that it presents your findings in a logical and organized way.

5. Submission Guidelines

5.1 Group Registration

For this assignment, you can work in groups of TWO. Please register your group under *People* → *Group* → *A1.1-Group* or *People* → *Group* → *A1.2-Group* on Canvas (We have created two separate group sets to accommodate the large number of students enrolled in this course).

The group registration should be done by Friday, Aug 30th, 2024.

5.2 Submit your work

5.2.1 Submit to Kaggle

We use the Kaggle leaderboard for evaluating the results predicted by your models on unseen data. Follow the steps below to submit your work to the Kaggle leaderboard.

Kaggle link: https://www.kaggle.com/competitions/comp-4318-5318-2024-s-2-a-1/

1. Use the Kaggle link to join the competition, you need to create a Kaggle account if you don't have one.
2. Go to Team → Use your registered Group ID on Canvas as your team name (e.g., `A1.1-Group 1`). You can create a Kaggle team with up to 2 members.
3. Go to Description → Check the IMPORTANT NOTES for the assignment.
4. Submit Predictions → Follow the submission format and submit your prediction output file (`test_output.csv`).
5. Leaderboard → Check your accuracy score at the Leaderboard.

In summary, go to Kaggle Page → Join Competition → Create a Team → Submit Predictions → Submit file `test_output.csv`

IMPORTANT: This link is only available to the students of COMP4318/5318. All groups need to submit `test_output.csv` to Kaggle for marking puporse. Only 5 submissions are allowed per day for Kaggle. Group ID on Canvas and Kaggle have to be identical otherwise the submission will not be marked for the Accuracy part.

5.2.2 Submit to Canvas

Proceed to the submission box on Canvas and submit 4 files separately as follows:

1. A `.pdf` report file.
2. An `.ipynb` code file: a Jupyter Notebook containing all your implementation. You can reuse the provided `.ipynb` template.
3. A `.pdf` code file: this file is exported from the `.ipynb` file for checking plagiarism.
4. A `test_output.csv` file: contains the predictions made by your best classifier on unseen data. This file must be consistent with the one submitted on Kaggle.

There are two different submission boxes for the different group sets: *Assignment 1 - Submission (for A1.1-Group)* and *Assignment 1 - Submission (for A1.2-Group)*. Please ensure you submit to the correct box corresponding to your group ID.

File Naming Conventions

The submission files should be named with your group ID and all student ID (SID) separated by the underscore (_). For example,

- a1_groupID_SID1_SID2.ipynb (code)
- a1_groupID_SID1_SID2.pdf (pdf version of the code)
- a1_groupID_SID1_SID2_report.pdf (report)

where SID1 and SID2 are the SIDs of the two students.

Important Notes:

- Only one group member needs to submit the assignment on behalf of the group.
- Do NOT submit the dataset or zip files to Canvas. We will copy the `data` folder to the same directory with your `.ipynb` file to run your code. Please make sure your code is able to read the dataset from this folder.
- Both the code and report will be checked for plagiarism.

Other guidelines

1. Please refer to lecture notes, lab materials, and other course resources for different ML methods.
2. Please proceed your own way if we do not specify it in the assignment details.
3. You can use any packages or code which have been introduced in lectures or tutorials. If you use any other packages or code snippets, please put the reference at the bottom of the code. Otherwise, it will be considered as plagiarism and the relevant section will not be marked.

5.3 Late Submission Penalties

A penalty of MINUS 5 percent (-5%) for each day after the due date.

The maximum delay for assignment submission is 5 (five) days, after which assignment will not be accepted.

You should upload your assignment at least half a day or one day prior to the submission deadline to avoid network congestion.

Canvas and Kaggle may not be able to handle a large number of submission happening at the same time. If you submit your assignment at a time close to the deadline, a submission error may occur causing your submission to be considered late. Penalty will be applied to late submission regardless of issues.

All files required for assignment 1 can be downloaded from Canvas → Assignments → Assignment 1 - Specification

5.4 Marking Rubric

Please refer to the rubric, which is available in the submission boxes on Canvas, for detailed marking scheme.

6. Inquiries after releasing the marking

After Assignment 1 marks come out, please submit your inquiries about marking within the 1st week. All inquiries after that will be ignored.

7. Academic honesty

Please read the University policy on Academic Honesty very carefully: https://sydney.edu.au/students/academic-integrity.html

Plagiarism (copying from another student, website or other sources), making your work available to another student to copy, engaging another person to complete the assignments instead of you (for payment or not) are all examples of academic dishonesty. Note that when there is

copying between students, both students are penalised – the student who copies and the student who makes his/her work available for copying. The University penalties are severe and include:

* a permanent record of academic dishonesty on your student file,
* mark deduction, ranging from 0 for the assignment to Fail for the course
* expulsion from the University and cancelling of your student visa.

In addition, the Australian Government passed a new legislation last year (Prohibiting Academic Cheating Services Bill) that makes it a criminal offence to provide or advertise academic cheating services - the provision or undertaking of work for students which forms a substantial part of a student's assessment task. Do not confuse legitimate co-operation and cheating!