Hong Kong University of Science and Technology
COMP 4211: Machine Learning
Spring 2024

Programming Assignment 1
Due: 6 March 2024, Wednesday, 11:59pm

1 Objectives

The objectives of this programming assignment are:

- To practice data importing and preprocessing skills using the `pandas` library.
- To better understand supervised learning methods by using `scikit-learn`.
- To evaluate the performance of several supervised learning methods by conducting an empirical study on a real-world dataset.

2 Dataset

You will use an anonymous education dataset provided as a ZIP file (data.zip). There are two CSV data files: train.csv (training set) and test.csv (test set). The details of the dataset are unknown except that **the second last column of the CSV files is the regression target class**, while **the last column is the classification target class** and has two labels: success and failure.

3 Major Tasks

The assignment consists of five parts and a written report:

Part 1: Use `pandas`, `scipy`, `matplotlib`, or `seaborn` for data importing and exploration.

Part 2: Use `scikit-learn` for data preprocessing.

Part 3: Use the linear regression model and feedforward neural networks for regression.

Part 4: Use the logistic regression model and feedforward neural networks for classification.

Part 5: Use `scikit-learn` to tune the hyperparameters and validate the data preprocessing.

Written Report: Report the results and answer some questions.

More details will be provided in the following sections. Note that **[Q*n*]** refers to a specific question (the nth question) that you need to answer in the written report. All the coding work should be done using Python 3.

4 Part 1: Data Exploration and Preparation

Load the provided train.csv file using `pandas`. Write code to perform an initial data exploration to understand the basic statistics about all the attributes and target variables. Although you

need to run the code to explore this dataset, you only need to answer the questions [**Q1**] to [**Q6**] in the report without code submission. Your report should include:

[**Q1**] **Dataset Overview.**

- **Size of the Dataset:** Total number of instances and features.
- **Feature Types:** Identify numerical and categorical features.

[**Q2**] **Missing Values.**

- **Identification:** Report any features with missing values and the proportion of missing values for them.
- **Potential Impact:** Briefly discuss how these missing values might affect analysis and model performance.

[**Q3**] **Feature Distribution.**

- **Numerical Features:** Identify which ones are discrete or continuous. Describe the distribution (e.g., mean, median, range, variance) of the first 3 numerical features and visualize their distributions using box plots.
- **Categorical Features:** Identify which ones are binary, nominal, or ordinal. Summarize the categories count of the first 3 categorical features and visualize their distributions using bar plots.

[**Q4**] **Outliers.**

- **Detection:** Identify potential outliers in the first 3 numerical features. You can use `scikit-learn` or `scipy` to help you. Please show the results of at least two outlier detection methods.
- **Consideration:** Briefly discuss how outliers might affect your preprocessing and modeling strategy.

[**Q5**] **Correlation Analysis.**

- **Feature Correlation:** Visualize the correlation between every two features (numerical features only, including the regression targets) with a heatmap. You can use `matplotlib` and `seaborn` to help you.
- **Insights:** Highlight any strong correlations that might influence feature selection or necessitate feature engineering.

[**Q6**] **Initial Thoughts on Preprocessing.**

- Based on the exploration, briefly outline the preprocessing steps that are **necessary** for the regression models or neural networks to run.
- Briefly discuss any specific challenges identified during exploration that **are better addressed** in preprocessing (those that do not affect running models, but might affect model performance).

5 Part 2: Data Preprocessing Techniques

Preprocessing is an indispensable step in the machine learning pipeline, serving as the foundation upon which the performance of your models is built. It encompasses a range of techniques to transform raw data into a more suitable and effective format for modeling. Proper preprocessing can significantly enhance model accuracy, efficiency, and interpretability by addressing missing values, inconsistent data formats, and irrelevant or redundant features.

In this task, you will apply a variety of advanced preprocessing techniques from `scikit-learn` to your dataset. You are expected to understand the rationale behind each technique and its impact on the data to predict the effect on subsequent model performance. In addition to the required APIs, you can additionally use and report on any other preprocessing and normalization modules in `scikit-learn`. You will have a chance to test these conjectures in Part 5.

[**Q7**] **Handling Missing Values:** Apply `SimpleImputer` to handle missing values. Experiment with different imputation strategies (mean, median, mode, and constant) on different types of feature columns with missing values. Briefly discuss their impact on feature distribution and model performance. Briefly judge what imputation should be used and when.

[**Q8**] **Normalization and Standardization:** Normalize/standardize different types of numerical features using `StandardScaler`, `MinMaxScaler`, or `RobustScaler`. You need to report the first numerical feature column of the first 10 samples before and after processing. Briefly discuss the difference between these techniques and when to use each.

[**Q9**] **Encoding Categorical Variables:** Utilize `OneHotEncoder` and `OrdinalEncoder` to encode different types of categorical variables. You need to report the first categorical feature column of the first 10 samples before and after processing. Briefly explain the scenarios where each encoding technique is preferred.

[**Q10**] **Feature Selection:** Use at least two feature selection modules in `scikit-learn`, such as `VarianceThreshold` and `SelectKBest`. Think about how feature selection impacts the performance of your models. You need to report which features were removed for what reasons.

[**Q11**] **Feature Engineering:** Assuming you plan to use Linear Regression only, please suggest one feature engineering method that could help improve model performance. Justify the creation of this new feature based on your data exploration findings.

6 Part 3: Regression

6.1 Linear Regression

In this task, you will build six linear regression models in the first step, where each model uses only one feature (not necessary to be those selected in [**Q10**]) to find whether it is correlated with the regression target (the second last column). (Hints: It is easier to work with binary categorical or continuous numerical features as independent variables)

Then, in the second step, you will build another linear regression model to explore the relationship between linear combinations of the six selected features and the regression target.

You are required to use the `train_test_split` submodule in `scikit-learn` to split the data in `train.csv`, with 80% for training and 20% for validation. You should set `random_state` = 4211 for reproducibility.

[**Q12**] After training the seven models described above using the training set, use them to make predictions on the validation set. Report the validation R^2 score of each of the models to evaluate the relationship between different features and the regression target.

[**Q13**] Report the mean squared error of each of the seven models in the validation set and compare them based on the two performance metrics, i.e., R^2 score and mean squared error.

[**Q14**] Discuss the mathematical meaning of the weight (in brief) for a binary categorical independent variable. How can a categorical feature with more than 2 possible values be formulated as the independent variable of a linear model? Pick a categorical feature with more than 2 possible values for the linear regression model and repeat [**Q12-13**].

6.2 Feedforward Neural Networks

In this part, you are asked to use the six features selected in Section 6.1 to train feedforward neural networks.

You need to try different numbers of hidden units $H \in \{1, 8, 32, 128\}$ to build different three-hidden-layer neural networks. The hyperparameter `early_stopping` can be set to 'True' to avoid overfitting (default is 'False'). The other hyperparameters may just take their default values. For each hidden layer in a specific neural network model, the number of hidden units should be kept the same for simplicity. During training, you are expected to record the training time of each model. After training, evaluate your models by reporting the R^2 scores on the validation set. You have to report the R^2 score for *each value* of H by plotting them using `matplotlib`.

[**Q15**] Report the model setting, training time, and performance of the neural network model for each value of H. You are also expected to repeat each setting three times for the same hyperparameter setting and report the mean and standard deviation of the training time and R^2 score for each setting.

[**Q16**] Compare the training time and R^2 score of the linear regression model and the best neural network model.

7 Part 4: Classification

In this task, you will build a logistic regression model as well as neural network classifiers to predict whether or not the classification target is 'success'. You are also required to use the `train_test_split` submodule in `scikit-learn` to split the data, with 80% for training and 20% for validation. As before, we ask that you set `random_state` = 4211 for reproducibility.

7.1 Logistic Regression

Learning of the logistic regression model should use a gradient-descent algorithm by minimizing the cross-entropy loss. It requires that the step size parameter η be specified. Try out a few values (<1) and choose one that leads to stable convergence. You may also decrease η gradually during the learning process to enhance convergence. When set properly, this can be done automatically in `scikit-learn`. Use the features selected in [**Q10**] to train the model. During training, record the training time for the logistic regression model. After training, you are required to evaluate your model using accuracy and the F1 score on the validation set.

[**Q17**] Report the model setting, training time, and performance of the logistic regression model.

Since the solution found may depend on the initial weight values, you are expected to repeat each setting three times and report the corresponding mean and standard deviation of the training time, accuracy, and the F1 score for each setting.

[**Q18**] Plot the ROC curve calculated on the validation set with the last model in [**Q17**] and report the AUC value. Give one reason why we need to examine the ROC curve as well.

[**Q19**] Train the model on the training set and report the accuracy and F1 score on the validation set using different learning rates. The `random_state` hyperparameter is again set to 4211. Other settings are the same as those mentioned in [**Q17**].

7.2 Feedforward Neural Networks

Neural network classifiers generalize logistic regression by introducing one or more hidden layers. Their learning algorithm is similar to that for logistic regression, as described above. Remember to standardize the features before training and validation.

You need to try different numbers of hidden units $H \in \{1, 8, 32, 128\}$ to build different three-hidden-layer neural networks. The hyperparameter `early_stopping` can be set to 'True' to avoid overfitting (default is 'False'). The other hyperparameters may just take their default values. During training, you are expected to record the training time of each model. After training, evaluate your models using accuracy and the F1 score on the validation set. You have to report the accuracy and F1 score for *each value* of H by plotting them using `matplotlib`.

[**Q20**] Report the model setting, training time, and performance of the neural networks for each value of H. You are also expected to repeat each setting three times and report the mean and standard deviation of the training time, accuracy, and the F1 score for each setting.

[**Q21**] Plot the accuracy and F1 score for each value of H. Suggest a possible reason for the gap between the accuracy and F1 score.

[**Q22**] Compare the training time, accuracy, and the F1 score of the logistic regression model and the best neural network model.

[**Q23**] Do you notice any trend when you increase the hidden layer size from 1 to 128? If so, please describe what the trend is and suggest a reason for your observation.

8 Part 5: Performance Enhancement

8.1 Preprocessing Validation

This section aims to demonstrate the impact of different preprocessing techniques on model performance by constructing and evaluating multiple pipelines. This task is designed to provide hands-on experience with `sklearn`'s `Pipeline` and `ColumnTransformer` utilities, enabling efficient experimentation with various preprocessing strategies. You could use `Pipeline` to create a sequence of preprocessing steps for numerical and categorical data separately. Then, `ColumnTransformer` could be used to apply the respective pipelines to your dataset's numerical and categorical columns. Your task is to evaluate how different preprocessing combinations affect the performance of three-hidden-layer neural networks to predict whether or not the target is a success.

[**Q24**] **Combination A:** For numerical features, apply `StandardScaler` and `SimpleImputer`

with mean strategy for imputation. For categorical features, use `OneHotEncoder` for encoding. Do not apply feature selection and feature engineering. Validate this combination with a neural network model.

[Q25] Combination B: For numerical features, use `MinMaxScaler` for normalization and `SimpleImputer` with zero strategy for imputation. For categorical features, encode ordinal features with `OrdinalEncoder` and remaining with `OneHotEncoder`. Do not apply feature selection and feature engineering. Validate this combination with a neural network model.

[Q26] Combination C: Create a custom combination where you choose a different preprocessing technique for numerical and categorical features based on your hypothesis of what might work best. This could also involve custom encoders, any appropriate feature selection, or engineering techniques. Validate this combination with a neural network model.

8.2 Hyperparameter Tuning

In this task, you need to use grid search to tune the hyperparameters of a three-hidden-layer feedforward neural network model to predict whether or not the target is a success. All the hyperparameters defined in the `MLPClassifier` class in `scikit-learn` except the number of hidden layers can be tuned. Use the best preprocessing techniques in **[Q26]** and features selected in **[Q10]** for training and testing.

You are required to use the `model_selection` submodule in `scikit-learn` to facilitate performing grid search cross-validation for hyperparameter tuning. This is done by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%. Five such random data splits are performed and the average over these five trials is used to estimate the generalization performance. You are expected to try at least 10 combinations of the hyperparameter setting. Set the `random_state` hyperparameter of the neural network model to 4211 for reproducibility and `early_stopping` to 'True' to avoid overfitting.

[Q27] Report five combinations of the hyperparameter setting and highlight the best hyperparameter setting in terms of accuracy. You also need to report the mean and standard deviation of the validation accuracy of the five random data splits for each hyperparameter setting.

9 Report Writing

Answer **[Q1]** to **[Q27]** in the report.

10 Some Programming Tips

As is always the case, good programming practices should be applied when coding your program. Below are some common ones but they are by no means complete:

- Using functions to structure your code clearly
- Using meaningful variable and function names to improve readability
- Using consistent styles
- Including concise but informative comments

You are recommended to take full advantage of the built-in classes of `scikit-learn` to keep your code both short and efficient. Proper use of implementation tricks often leads to speedup by orders of magnitude. Also, please be careful to choose the built-in models that are suitable

for your tasks, e.g., `sklearn.linear_model.LogisticRegression` is *not* a correct choice for our logistic regression model since it does not use gradient descent.

11 Assignment Submission

Assignment submission should only be done electronically on the Canvas course site.

There should be two files in your submission with the following naming convention required:

1. **Report** (with filename `report_{student_id}.pdf`): in PDF form.
2. **Source code** (with filename `code_{student_id}.zip`): all necessary code, written in one or more Jupyter notebooks, compressed into a single ZIP file. The data should not be submitted to keep the file size small.

When multiple versions with the same filename are submitted, only the latest version according to the timestamp will be used for grading. Files not adhering to the naming convention above will be ignored.

12 Grading Scheme

This programming assignment will be counted towards 10% of your final course grade. Note that the plus sign (+) in the last column of the table below indicates that reporting without providing the corresponding code will get zero point. The maximum scores for different tasks are shown below:

Grading scheme	Code (39)	Report (61)
Part 1		
- Report number of instances [Q1]		1
- Report number of features [Q1]		1
- Identify numerical features [Q1]		1
- Identify categorical features [Q1]		1
- Report features with missing values [Q2]		1
- Report the proportion of missing values [Q2]		1
- Briefly discuss the impact [Q2]		1
- Identify which ones are discrete or continuous [Q3]		1
- Describe the distribution [Q3]		1
- Visualize their distributions [Q3]		1
- Identify which ones are binary, nominal, or ordinal [Q3]		1
- Summarize the categories count [Q3]		1
- Visualize their distributions [Q3]		1
- Identify potential outliers [Q4]		2
- Briefly discuss how outliers' impact [Q4]		1
- Visualize the correlation [Q5]		1
- Highlight and discuss any strong correlations [Q5]		1
- Briefly outline the necessary preprocessing steps [Q6]		1
- Briefly discuss any specific challenges [Q6]		1
Part 2		
- Perform missing value imputation with four imputation strategies [Q7]	2	
- Discuss the impact on feature distribution and model performance [Q7]		1
- Briefly judge what imputation should be used and when [Q7]		1
- Perform normalization/standardization using three scalars [Q8]	2	
- Report one feature column before and after processing [Q8]		+1
- Briefly discuss the difference between these techniques and when to use [Q8]		1
- Encode categorical features using two encoders [Q9]	2	
- Report one feature column before and after processing [Q9]		+1
- Briefly explain the scenarios where each encoding technique is preferred [Q9]		1
- Try two feature selection modules [Q10]	2	
- Report how feature selection impacts the performance of your models [Q10]		1
- Report which features were removed for what reasons [Q10]		+1
- Suggest one suitable feature engineering [Q11]		1
- Justify your idea [Q11]		1
Part 3		
- Build the linear regression model	2	
- Compute the R^2 scores of the seven linear regression models + [Q12]	1	+1
- Make prediction on the validation set	2	
- Compute the MSE scores of the seven linear regression models + [Q13]	1	+1
- Build the model and make prediction on the validation set + [Q14]	1	+2
- Build the feedforward neural network model	2	
- Compute the training time and R^2 score for each value of H in the feedforward neural network model + [Q15]	2	+2
- Plot the R^2 score with different values of H for the feedforward neural network model + [Q16]	1	+2

Grading scheme	Code (39)	Report (61)
Part 4		
- Build the logistic regression model by adopting the gradient descent optimization algorithm	2	
- Compute the training time, accuracy, and F1 score of the logistic regression model + [Q17]	1	+1
- Calculate the ROC curve and AUC value on the validation set + [Q18]	1	+1
- Logistic regression with different learning rates + [Q19]	2	+1
- Build the feedforward neural network model	2	
- Compute the training time, accuracy, and F1 score for each value of H in the feedforward neural network model + [Q20]	2	+1
- Plot the accuracy and F1 score with different values of H for the feedforward neural network model + [Q21]	1	+1
- [Q22]		+2
- [Q23]		+2
Part 5		
- Run the pipeline of combination A [Q24]	2	
- Report the model performance [Q24]		+2
- Run the pipeline of combination B [Q25]	2	
- Report the model performance [Q25]		+2
- Run the pipeline of combination C [Q26]	2	
- Report the model performance [Q26]		+2
- Grid search on the feedforward neural network model [Q27]	3	
- Report five combinations of the hyperparameter setting [Q27]		+1
- Report their performances [Q27]		+1

Late submission will be accepted but with penalty.

The late penalty is deduction of one point (out of a maximum of 100 points) for every hour late after 11:59pm with no more than two days (48 hours). Being late for a fraction of an hour is considered a full hour. For example, two points will be deducted if the submission time is 01:23:34.

13 Academic Integrity

Please refer to the regulations for student conduct and academic integrity on this webpage: `https://registry.hkust.edu.hk/resource-library/academic-standards`.

While you may discuss with your classmates on general ideas about the assignment, your submission should be based on your own independent effort. In case you seek help from any person or reference source, you should state it clearly in your submission. Failure to do so is considered plagiarism which will lead to appropriate disciplinary actions.