[CS 412. Final project report]

Students’ academic performance prediction

Hyeonjeong Park; Seyedeh Fatemeh Ahmadi; Parisa Salehi; Muhammad Talha Adnan Khan

1. Introduction

Throughout this course, we've been introduced to the fundamental concepts of Machine Learning, including supervised and unsupervised learning, neural networks, and model evaluation. Together, we've explored how these techniques are applied in diverse fields.

In supervised learning, we've explored how algorithms can be trained on labeled data to make predictions or classify new instances. For example, in the realm of healthcare, supervised learning techniques could be utilized to predict the likelihood of a patient developing a certain disease based on their medical history. By using ground truth information, supervised learning-based algorithms have shown superior performance in many fields, including image classification, detection, and segmentation.

On the other hand, unsupervised learning has intrigued us with its ability to identify hidden patterns or structures in unlabeled data. An example of this is in customer segmentation for marketing purposes, where unsupervised algorithms can group customers (called clustering, such as agglomerative clustering algorithms) based on similarities in their purchasing behavior without prior labels. Due to the absence of ground truth, unsupervised learning algorithms have shown lower performance than those of supervised learning-based methods, however, it has gotten much attention

since it does not require labeling costs. Hence, recent unsupervised models with a huge amount of training data have surpassed supervised models.

As a group, we're eager to deepen our understanding of Machine Learning by exploring advanced topics such as natural language processing and computer vision, and their integration with domains like healthcare and finance. We believe that Machine Learning holds immense potential for addressing real-world challenges, and we're committed to continuing our exploration and contributing to this field.

2. Project: students' academic performance prediction

This project aims to predict students' academic performance in three levels: low, middle, and high. Predicting academic performance in advance or during a semester is vastly crucial to enhancing learning quality and providing a personalized education system. For this reason, the demand for accurate academic performance prediction systems is gradually increasing.

Technically, this task is a kind of classification to classify three categories. To this end, diverse features, including environmental and innate factors such as nationality, education level, grade level, and gender, are given for training a machine learning model. From these given inputs, machine learning models will learn decision boundaries between the categories in different ways. To demonstrate the classification ability of machine learning models, we adopt the three representatives for our project: Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), and Neural Network (NN). Then, we construct a general pipeline to train and evaluate the models. Finally, we show and compare experimental results to verify the models' abilities.

3. Methods

1. Dataset

This dataset, Students' academic performance dataset, is constructed by the University of Jordan using a learner activity tracker tool, called experience API (xAPI). xAPI is a component of the training and learning architecture (TLA) that monitors learning progress and learners' actions such as solving a problem or watching a lesson video.

The dataset includes academic records of 480 students. The academic records are composed of 16 features and performance categories, which are 'low', 'middle', and 'high'. The features are divided into three major categories: 1) Demographic features such as gender and nationality. 2) Academic background features, for example, educational stage, grade level, and class section. 3) Behavioral features such as raising a hand in a class, the number of absences, and accessing resources. Technically, the dataset includes the features in terms of parents, which could be a crucial factor for education, such as answering a survey by parents and the level of parents' school satisfaction. All features are illustrated in Table X.

Table 1. 16 features of Students' academic performance dataset

Gender	GradeID	Relation	Discussion
Nationality	SectionID	RaiseHands	ParentAnsweringSurvey
Place of Birth	Topic	VisitResources	ParentSchoolSatisfaction
StageID	Semester	AnnouncementView	StudentAbsenceDays

Specifically, the dataset includes 305 males and 175 females from diverse origins, such as Kuwait, Jordan, Palestine, Venezuela, and the US. In terms of

nationality, the dataset is highly imbalanced. For instance, the number of students from Kuwait is 179, while that of students from Venezuela is only 1.

2. Our designs

a. Data preprocessing

Firstly, we need to divide the whole dataset into a training, a validation, and a testing set. However, we do not split the validation set since our dataset is small-scale. Therefore, we divide the data into training and testing sets with a ratio of 8:2. Then, if we need to use a validation set to search for the best hyper-parameter, we divide the training set into a temporary training set and a validation set with the same ratio. The statistics of the divided data sets are shown below:

	General set	Train set	Test set	features
qty	480	384	96	16

Figure 1. Training and test set split

Then, we convert the raw data to a trainable format to train or fit the machine learning models. In other words, non-numerical features such as nationality and gender should be transformed into numerical features. For example, ‘M’, indicating male, is converted to 1 while ‘F’ is converted to 2 like below:

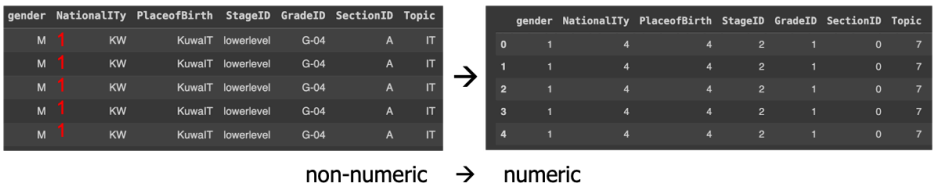

Figure 2. An example of data preprocessing

Next, we need to normalize each column since the features have different ranges, which could hinder representation learning. For instance, ‘gender’ column has 0 and 1 as values while the ‘VisitedResources’ column has 50, which is significantly larger than the value of ‘gender’. In this case, ‘VisitedResources’ column might dominate the whole training process as it has a larger value. Moreover, this huge gap could make the training unstable. To prevent this, we utilize the ‘keras.utils.normalize’ function or ‘StandardScalar’ function provided by scikit-learn library. Specifically, the former normalizes each data vector to have a length of 1 while the latter standardizes the data to have a mean of 0 and a standard deviation of 1 for each vector.

After performing all these steps, we could train the ML models.

b. Related libraries and modules

The ML code extensively employs several critical libraries and modules for data handling, model building, and performance evaluation. The primary library used is Scikit-learn, which provides a robust framework for machine learning, offering tools for data preprocessing, model selection, and model evaluation. Additionally, Pandas is utilized for advanced manipulation and analysis, facilitating operations on numerical tables and time series. NumPy supports efficient numerical computations, which is essential for manipulating large arrays and matrices. Matplotlib is used to visualize detailed graphs and plots to assess model performance. Threading and resource module is incorporated for monitoring system resource usage during

computation-heavy tasks. Together, these libraries create a comprehensive environment for developing, tuning, and evaluating effective machine learning models.

c. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm used for both classification and regression tasks, though it is most commonly used for classification. At its core, SVM seeks to find the best hyperplane that separates different classes in the feature spaces with the maximum margin. The margin is defined as the distance between the separating hyperplane and the nearest data points from each class, known as support vectors.

Initializing and Training the Classifier: The SVM Classifier is initialized using Scikit-learn's `SVC` module, a powerful tool for implementing support vector machines. This classifier is configured with various parameters such as kernel type (liner, RBF, polynomial, sigmoid), C(regularization parameter), and gamma(kernel coefficient), which are optimized through systematic grid search using `GridSearchCV`. This process not only tests multiple combinations of parameters but also cross-validates (since we do not split the validation set explicitly, the training set will be divided into a new subset of the training set and a validation set here) to determine the set that yields the best performance, refining the model's ability to generalize to new data. Once optimal parameters are identified, the classifier is trained with these settings on the training dataset (X_train and y_train). The

training is executed within a separate thread to enhance the processing efficiency, utilizing parallel computation capabilities. The `fit` method is employed, where the model learns from the training data, adjusting itself to minimize the errors. During training, system resources such as CPU time and memory usage are monitored using the resource module, ensuring the process is resource-efficient.

d. K-Nearest Neighbor (KNN)

We explore the implementation of the k-Nearest Neighbors (k-NN) algorithm using Python's Scikit-Learn library. The k-NN algorithm is a simple yet powerful machine learning technique used for both classification and regression tasks. It classifies a data point based on how its neighbors are classified.

Initializing and Training the Classifier. The k-NN classifier is initialized with a specific number of neighbors. Here, knn = KNeighborsClassifier (n_neighbors=3) initializes a k-NN classifier with 3 neighbors. The classifier is then trained using the fit method on the training dataset X_train and labels y_train. However, the best number of k should change depending on the dataset and its scale. Therefore, we need to perform cross-validation to find the best k for this task. To this end, we divide the training set into a subset of the training set and a validation set. After finding the best k on the validation set, we use the original whole training set to train a model with the best k. Specifically, inside the loop for each value of k, the k-NN classifier is

initialized with knn = KNeighborsClassifier(n_neighbors=i) and trained on X_train and y_train.

e. **Neural Network (NN)**

Libraries and Modules: The NN code makes use of several essential libraries and modules for building, training, and evaluating classification models. The primary library employed is TensorFlow, accessed through its Keras interface. TensorFlow provides a comprehensive framework for developing deep learning models, offering high-level APIs for defining neural network architectures, optimizing performance, and evaluating results. Additionally, NumPy is utilized for efficient numerical computations, facilitating data manipulation and preprocessing tasks essential for machine learning workflows. Importantly, TensorFlow's keras.callbacks module is leveraged to implement the ModelCheckpoint callback, enabling the automatic saving of the best-performing model during training for future use.

Initializing and Training the Classifier: Each NN model implementation initializes a neural network classifier using TensorFlow's Keras API, specifically the Sequential model, a fundamental architecture for building sequential neural networks. These classifiers are composed of multiple layers, including densely connected layers with Rectified Linear Unit (ReLU) activation functions, in some cases, dropout layers for regularization to mitigate overfitting, and, in some other cases, batch normalization layers for improved training stability and convergence. Subsequently, the models are

compiled with specific loss functions, optimizers, and evaluation metrics tailored to the classification task at hand. The fit method is then employed to train the models, utilizing training data (X_train and y_train), validation data (X_val and y_val), batch size, number of epochs, and callback functions for monitoring and saving the best model during the training process.

Here, we train a simple MLP consisting of 5 layers. Since the dimension of the input is 16, we set the layer dimension to 8. Our MLP model is illustrated in Figure 3. Then, we train our model using the Adam optimizer for 300 epochs. We set the initial learning rate and mini-batch size as 0.01 and 64, respectively.

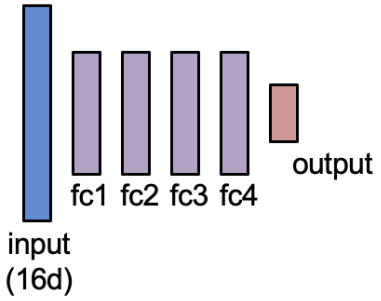

Figure 3. Our MLP model for students' academic performance prediction

4. Experimental results and analysis

1. Evaluation

After the model concludes, the best-performing classifier, identified and saved during the optimization process, is used to make predictions on the test dataset (X_test). The predicted class labels(y_pred) from this classifier are then compared against the ground truth (y_val) to evaluate the model's performance on unseen data. This step assesses the classifier's effectiveness

in generalizing beyond training data. The evaluation is comprehensive, employing metrics such as accuracy, precision, recall, and F1-score, which are calculated and presented using Scikit-learn's "classification-report" function. These metrics provide a detailed analysis of the model's performance, offering insights into its accuracy and identifying areas where the model excels or may require improvement. Additionally, predictions and accuracy calculations are also conducted on the training dataset (X_train) to assess the model's generalization capabilities and its potential for overfitting. By evaluating the accuracy of the classification model on both the training and validation datasets, we can gain valuable insights into the model's effectiveness and robustness across different data subsets, aiding in informed decision-making and model selection.

2. **Results of SVM** :

The result of the classification report is illustrated in Figure 4.

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
Best Parameters: {'C': 1, 'gamma': 'auto', 'kernel': 'poly'}
Best Cross-Validation Score: 0.7839029391660971
CPU Time Used: 5.1327940000000005 seconds
Memory Used: 1424.0 MB
Accuracy on validation set: 0.7083333333333334
Classification Report on validation set:
              precision    recall  f1-score   support

           0       0.74      0.59      0.65        29
           1       0.83      0.76      0.79        25
           2       0.64      0.76      0.70        42

    accuracy                           0.71        96
   macro avg       0.74      0.70      0.71        96
weighted avg       0.72      0.71      0.71        96
```

Figure 4. Classification report of SVM

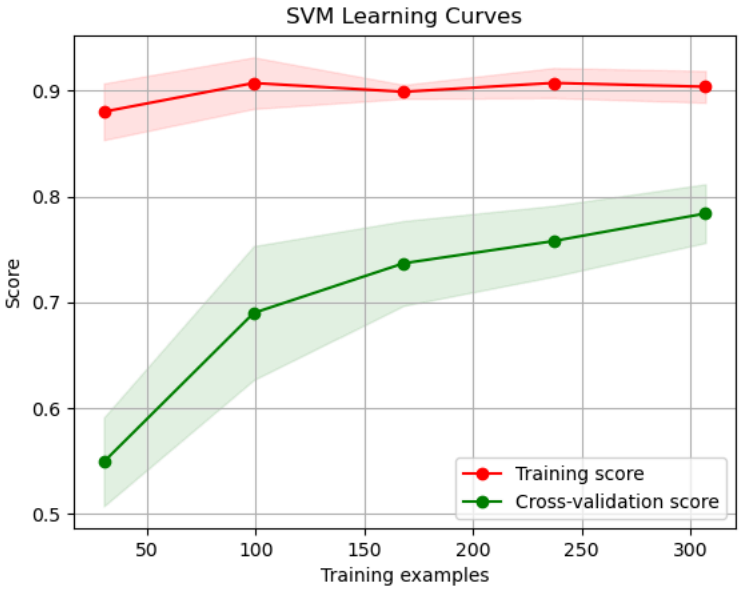

Figure 5. SVM learning curve

After the grid search, the best hyperparameters were found to be ‘C’:1, ‘gamma’:auto, and ‘kernel’:ploy. We obtained a 70.8% accuracy using these hyperparameters. The transparent regions in the graph represent the variability of the scores around the mean score for both the training and testing datasets. These regions illustrate the uncertainty of the results across different runs of the model training and validation.

3. Results of KNN

Firstly, when we simply apply k=3, all the results of each metric are shown in Figure 6. The overall accuracy achieved by the model was 53.125%, indicating the proportion of total predictions that were correct. These results indicate that while the kNN classifier has performed reasonably well in some classes, there is significant room for improvement, particularly in enhancing both precision and recall across all classes to boost overall accuracy.

```
Accuracy: 0.53125
Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.39      0.38        28
           1       0.81      0.65      0.72        26
           2       0.51      0.55      0.53        42

    accuracy                           0.53        96
   macro avg       0.56      0.53      0.54        96
weighted avg       0.55      0.53      0.54        96
```

Figure 6. Classification report of KNN

Since the accuracy was lower than we expected, we experimented with different k values as we wanted to explore the effect of varying the number of neighbors (k) in the k-NN classifier across a wider range. The aim is to understand how the choice of k influences the accuracy of the classifier. Therefore, we consider a range (loop) from 1 to 30. By plotting the accuracies in Figure 7, we can visually identify the optimal k value that maximizes accuracy (k=17), which is 63.5 %. This helps in understanding the balance between too few and too many neighbors, highlighting the importance of parameter tuning in machine learning models.

When we used a smaller number of neighbors (like 1 or 2), the accuracy was around 55.2% and 52.1%, respectively. This shows that using very few neighbors does not give the best results. As we increased *k* to around 10 to 15, the accuracy improved, reaching up to 61.5% with 10 neighbors. This suggests that a moderate number of neighbors helps the model perform better. When we used even more neighbors (20 to 30), the accuracy stabilized around 59.4% to 60.4% so it shows using a high amount for k does not give high accuracy.

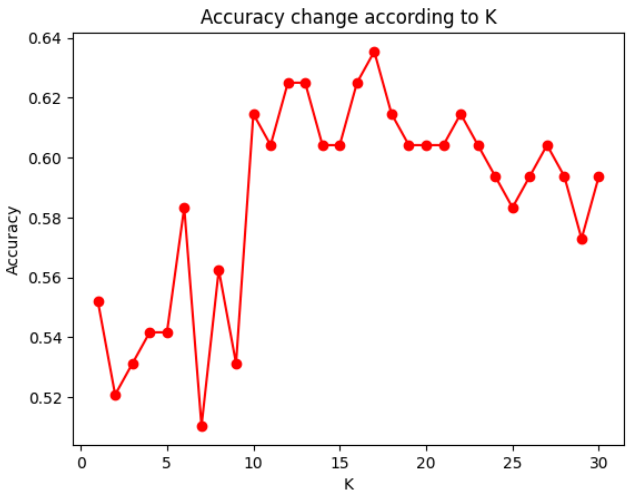

Figure 7. Accuracy change according to k

Because the size of the dataset was too small, we immediately looked for k suitable for the test set, but in reality, it is common to find k using the validation set. Therefore, we tried to find the most effective number of neighbors by cross-validation setting. Each fold tests different values of *k* to find which one gives the highest accuracy using a different validation set. Here's what we found for each fold in Figure 8. This helps ensure that our model is reliable and makes accurate predictions.

```
1 -th cross-validation, best k =  12     best_acc =  0.6103896103896104
2 -th cross-validation, best k =  30     best_acc =  0.5714285714285714
3 -th cross-validation, best k =  27     best_acc =  0.6623376623376623
4 -th cross-validation, best k =  7      best_acc =  0.5064935064935064
5 -th cross-validation, best k =  22     best_acc =  0.5584415584415584
```

Figure 8. The best accuracy of 5 folds

These results show that the optimal number of neighbors, *k*, varies across different subsets of the data. The highest accuracy obtained in any fold was 66.23% with *k* = 27, suggesting this might be a particularly effective setting for our dataset under these test conditions.

4. Results of NN

There are a lot of factors that affect to performance of neural networks such as regularization and normalization. From this perspective, we conducted ablation studies using dropout and batch normalization layer (BN).

As a result, both components increased the accuracy compared to the baseline model which has not both dropout and BN layer. Applying dropout increased by 8.4% while the BN layer increased by 7.3% as shown in Table 2.

Table 2. Performance comparison on the baseline, dropout, and BN layer

Method	Best accuracy (%)
baseline	75.0
baseline + dropout	85.4
baseline + BN layer	82.3

Interestingly, we could observe overfitting, meaning that the training accuracy is increasing while the testing accuracy is decreasing (or the training loss is decreasing while the test loss is increasing), if the model was trained without dropout as shown in Figure 9.

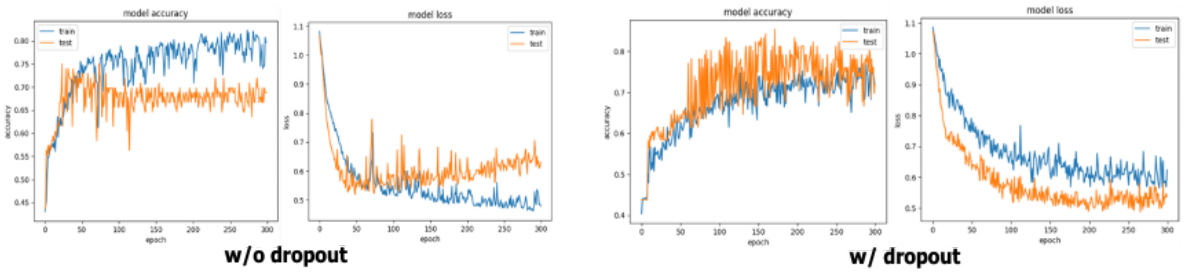

Figure 9. Overfitting (left) and well training (right)

Since the model with dropout showed the best performance, we analyzed the other aspects such as loss, training convergence, and diverse metrics based on it.

As shown in Figure 10 (left), in the initial learning phase (epoch 0-50), the model's accuracy on both training and test datasets showed a significant increase. Initially, the model rapidly learned the underlying patterns in the training data, which also generalized well to the test data. Then, in the epochs 50-200 we see an initial increase, the training accuracy continued to increase slightly, indicating further learning, whereas the test accuracy began to stabilize. This phase suggests that the model started fitting the training data more closely while maintaining a reasonable level of generalization. Finally, in epochs 200-300, both training and testing accuracies fluctuated within a specific range during this extended phase. This behavior could illustrate that the model was exploring the solution space to find a balance between bias and variance, attempting to generalize well without overfitting.

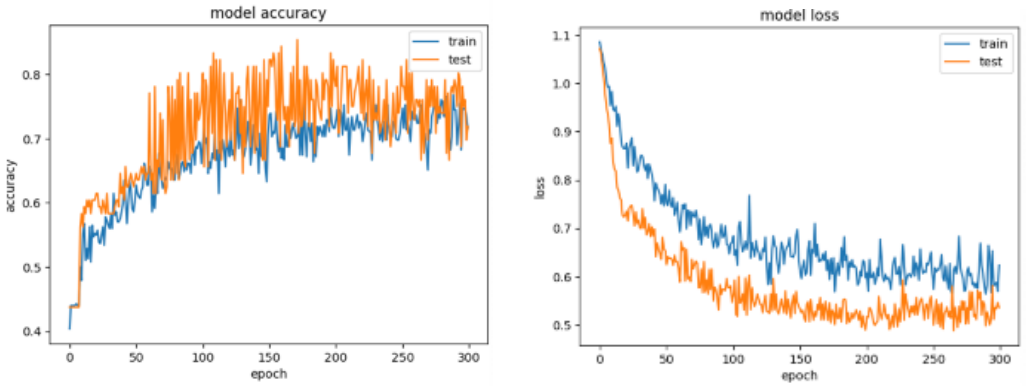

Figure 10. NN model accuracy (left) and loss (right)

As shown in Figure 10 (right), the neural network model provides insight into the optimization process over 300 epochs. The graph tracks the loss for both the training set and the testing set, offering a visual representation of the model's learning efficiency and stability over time. Initially, the model exhibits a sharp decrease in loss for both datasets, a typical indicator of effective initial learning where major errors are rapidly corrected. As training progresses, the rate of loss reduction slows, signaling that the model is entering a phase of fine-tuning its parameters with diminishing returns from each epoch. After the initial steep decline, both training and testing loss lines show a period of gradual decrease before entering a phase where the loss levels off. This steady state occurs despite ongoing training, which generally suggests that the model has reached the limits of what it can learn from the current configuration and dataset. The near-convergence of the training and testing loss values towards the end of the training, accompanied by their parallel progression, also indicates that the model is not overfitting. The close tracking of testing loss with training loss suggests that the model retains its generalization capabilities.

```
confusion matrix for train :
[[ 71   1  42]
 [  0  93   8]
 [ 25  23 121]]
```

```
confusion matrix for validation :
[[18  0 10]
 [ 0 26  0]
 [ 7  2 33]]
```

Figure 11. Confusion matrix for validation and train

Looking at the confusion matrices in Figure 11 shows how accurately NN could classify items into three different classes. It did an excellent job with the

second class during both validation and training, correctly identifying most items without any errors. This indicates that the model is highly reliable for recognizing features specific to the second class. However, the model faced challenges when distinguishing between the first and third classes. During the validation phase, while it did manage to identify 18 items correctly as belonging to the first class, it mistakenly classified 10 items as belonging to the third class. Similarly, for the third class, the model correctly identified 33 items but mistakenly classified 7 items from the first class and 2 from the second as belonging to the third class. These results suggest that while the model excels at identifying the distinct features of the second class, it struggles with the similarities between the first and third classes. To improve the model's overall accuracy, focusing on enhancing its ability to distinguish between these two closely aligned classes could be beneficial. This might involve refining the model's training data, modifying its sensitivity to the features that differentiate these classes, or adjusting its classification algorithms.

Overall, the model is quite effective, but there's still some room to make it even better.

5. Comparison and conclusion

In the context of evaluating different classification algorithms for a specific task, we compared the performance of Support Vector Machines (SVM), K-Nearest Neighbors (K-NN), and Neural Networks (NN) based on accuracy and inference speed. Among these methods, Neural Networks emerged as

the leader, achieving the highest accuracy of 85.40% and the fastest inference speed of 0.003 seconds (as shown in Table 3 below).

Table 3. Performance comparison

Method	Best accuracy (%)	Inference speed (s)
SVM	70.83	0.05
K-NN	63.50	0.01
NN	85.40	0.003

The best-performing model among each method was selected, and the inference speed was calculated for each student. It's important to note that the test set used in this analysis was relatively small (only 96 samples). Therefore, further evaluation with a larger dataset is recommended to solidify the conclusions drawn about inference speed.

5. Remarks and future work

Given the abundance of features in our classification project, we are keen to explore feature engineering as a means of refining our models. Feature engineering involves strategically modifying, selecting, or creating features from our dataset to enhance model performance. This process includes techniques such as feature selection, where we identify the most influential features; feature transformation, where we adjust feature distributions and scales; and feature creation, where we generate new features to capture complex relationships in the data.

Feature engineering holds significant importance in improving the efficacy of our classification models. By selectively choosing informative features, we can mitigate

issues such as overfitting and improve model generalization. Transforming features to adhere to desired distributions aids in better model training while creating new features allows us to capture nuances that might be overlooked otherwise.

Incorporating domain knowledge further enhances the relevance and accuracy of our features, ultimately leading to more robust and effective classifiers. In essence, by leveraging feature engineering techniques, we aim to optimize the performance of our SVM, NN, and KNN models, thereby achieving better classification outcomes.

6. Each team member’s contributions

We all actively participated in this project. We discussed the methods to solve the classification problem, shared the code, and got interesting experimental results. Specifically, Muhammad Talha Adnan Khan mainly conducted the SVM part, while Parisa Salehi took responsibility for K-NN. Seyedeh Fatemeh Ahmadi did most part of the NNs, and Hyeonjeong Park partially participated in both K-NN and NNs parts to correct the comparison setting and ablation study of NNs, respectively. Additionally, Talha gave a presentation as a representative, and Hyeonjeong made a presentation material.