Group 120: Machine Project Final Report

Introduction

Determining the difficulty level of a course is often challenging for students, as multiple factors contribute to how demanding a class may be. In this project, we aim to simplify this process by analyzing three primary data points: grade distributions, professor ratings, and class sizes at Georgia Tech. Our objective is to classify classes into difficulty tiers, such as “Easy,” “Medium,” and “Hard,” using machine learning techniques. By making these insights available, we hope to provide students with an accessible resource for better course planning.

Problem Definition

Our approach addresses a common dilemma for students: assessing the effort required for various courses. Instead of relying solely on subjective online reviews or anecdotal feedback, our model will use actual performance data. This structured classification can serve as an informed guide to help students anticipate course rigor and workload.

Data Collection & Preprocessing

Our initial data collection efforts were ambitious, but challenges arose. We were unable to obtain professor ratings from external sources or gather class sizes efficiently without manually reviewing each data point. Despite these limitations, we successfully acquired Georgia Tech grade distribution data from the Office of Institutional Research & Planning. This dataset included various class features, but we selected only the average grade, standard deviations of the grades, instructor name, subject, and course number. Our main preprocessing was encoding our strings in integers for machine learning compatibility. However, this initial encoding did not include semantic meaning, such as associating harder subjects with higher numerical values. We also removed labs and recitations to exclude significant outliers and eliminated classes missing an average grade, as this indicated a 100% withdrawal rate. Four our truth values, we manually labeled courses as “very easy,” “easy,” “medium,” “hard,” and “very hard” based on course numbers and average GPA using a simple algorithm. Finally, we decided to try and normalize our data using a simple method, but while this did give slightly better results, we ultimately decided to use scalarization as it gave even better results than normalization for our models.

Machine Learning Model

K-means actual implementation

After first preprocessing our data, we then used this preprocessed data as the training data for the K-means clustering, which was our first implemented model. We selected kmeans because it's a relatively simple and easy to use starter model for our data, and to act as a baseline for later comparisons. We used scikit-learn for the k-means clustering implementation, then used Streamlit for the visualization of the results. We decided to plot the grade distribution data variables such as course number and average grade as the independent variables and the true labels as the dependent variable for better visualized clusters.

Support Vector Machine

We then moved on to adding super vector machine model to our code. We selected this model due to its ability to easily classify where the data would have a good amount of linear separability, which we believe the data has. This is due to its use of a hyperplane. We also experimented with other types of kernels for the SVM.

Multilayer Perceptron

Next, we added an MLP mode mainly because we wanted to see how a neural network would work with our given data, but also because we believed that an MLP specifically would have the best classification for our model due to its ability to capture complex nature. While we believed that the data was mostly linear, we wanted to see if the MLP would have better classification because it saw some connections we, and other models, didn’t see.

Random Forest

Finally, we also implemented random forests as our last model. We chose this model due to its ability to counter overfitting as we feared that our previous models may only be useful for the training data that we gave it. We wanted to ensure that our code could be used outside of the limited data that we gave it.

Results

Overall

All things considered, our models performed well, except for k-means. The classification was usually correct, and our F1 scores were very high for our other models. In the future, our models would work much better, especially K-means, if we were able to obtain more data relevant to the course difficulty that do not have too much correlation to the strongest feature, average GPA, such as course schedules and the ratings of the professor.

K-Means

K-means under performance could have been a result of a few issues. The first is that the k-means algorithm might have identified patterns in the data that do not match our assumptions about course difficulty. It also could have been that the input variables may not be reliable indicators of difficulty, as evidenced by the wide variation in difficulty among higher course numbers. These problems could have been reduced by better preprocessing, but due to the performance of other models, we are not sure. In the future, we could run other models to find feature importance and limit the k-means to only those features to have a better chance of good clustering in the results.

Suport Vector Machine

SVM performed extremely well for our purposes, and this is most likely due to how we created our truth labels. They allowed for very easy linear separations of all the clusters, which is where SVMs excel. We used what we found to be the two most important features to define the hyperplane, which were the average GPA and the course number. Looking at the results, the SVM was able to clearly define clusters and was best defined by the linear kernel. Compared to the other models, the SVM clearly defined groupings when comparing the course number and the average GPA. This makes sense because the course number and average GPA were some of the strongest features, which is necessary for strong SVM results.

Multilayer Perceptron

The neural net performed well for the limited feature pool given. For our implementation, we fine-tuned our hyperparameters to try to our weighted F1-score and accuracy which was 0.97, which is very good. As seen in the loss plot, the neural network trained off the provided data very well to provide an accurate model using all features we could obtain. Compared to the other models, the multilayer perceptron had a very strong accuracy while being able to use all of the given features.

Random Forests

As seen in the visualizations, our labels do not fully align with the output cluster. A reason could be that our Random Forest model is identifying different patterns that aren't related to our assumptions. Another reason is that some variables are not strong indicators is that the course number and instructors were not as influential as we initially thought.

We originally thought the model would find a clear connection between higher course numbers and increased difficulty. The implementation revealed a different variety of difficulties among higher numbered classes. This meant the course number was not a good indicator of difficulty. As expected, our most influential feature was the average GPA with the standard deviation of the class grade coming in second. When comparing the random forests results with the other results it is shown that random forests best indicated the strengths of each feature more than the other models.

Next Steps

In the future, we would improve our encoding methods by enhancing encoding semantics to align with expected difficulty measures, such as associating lower professor ratings with higher values. We would also implement additional models to explore and implement other models, such as SVM and Random Forest, to compare their performance against K-Means and determine the best algorithm for classifying course difficulty. We also would like to expand the number of features we could obtain. We would develop methods to automatically gather class size and professor ratings data for richer input features.

We would keep our models the same as they worked well for what we wanted from each, except for maybe k-means. However, we would want to tweak parameters of each model a little more, especially for k-means, if we used it again, and random forest.

Contributions Table

- **Hayk Arsenyan**: Report Review, Methods Review
- **Alexandre Abchee**: Report Writing, Methods Review, Report Review
- **John Andrade**: Preprocessing, Methods Review, Data Collection, Report Review
- **Mattias Anderson**: Methods Implementation, Preprocessing, Results, Report Review, Video
- **Juan Macias-Romero**: Methods Implementation, Streamlit, Data Collection, Report Review

Citations

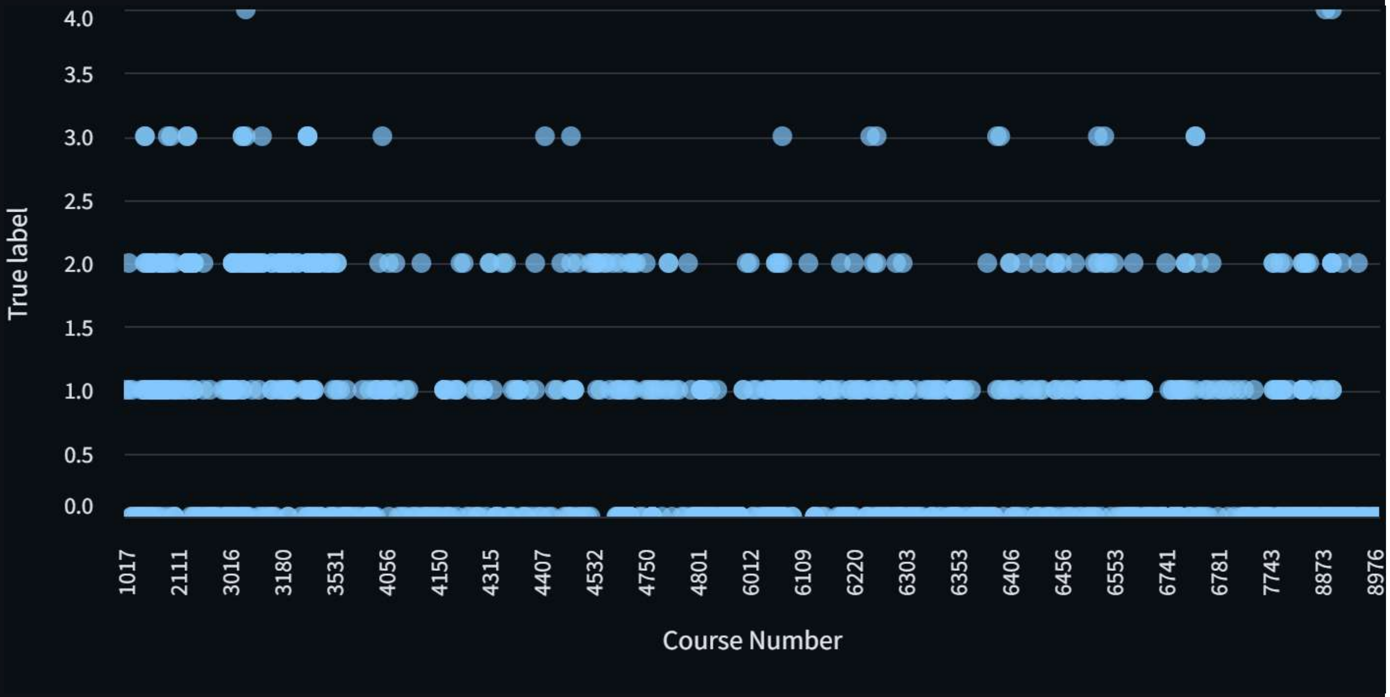
1. Small Classes, Big Possibilities [Accessed Oct. 4, 2024]
2. Grade Distribution - Georgia Tech [Accessed Oct. 4, 2024]

3. Rate My Professors [Accessed Oct. 4, 2024]

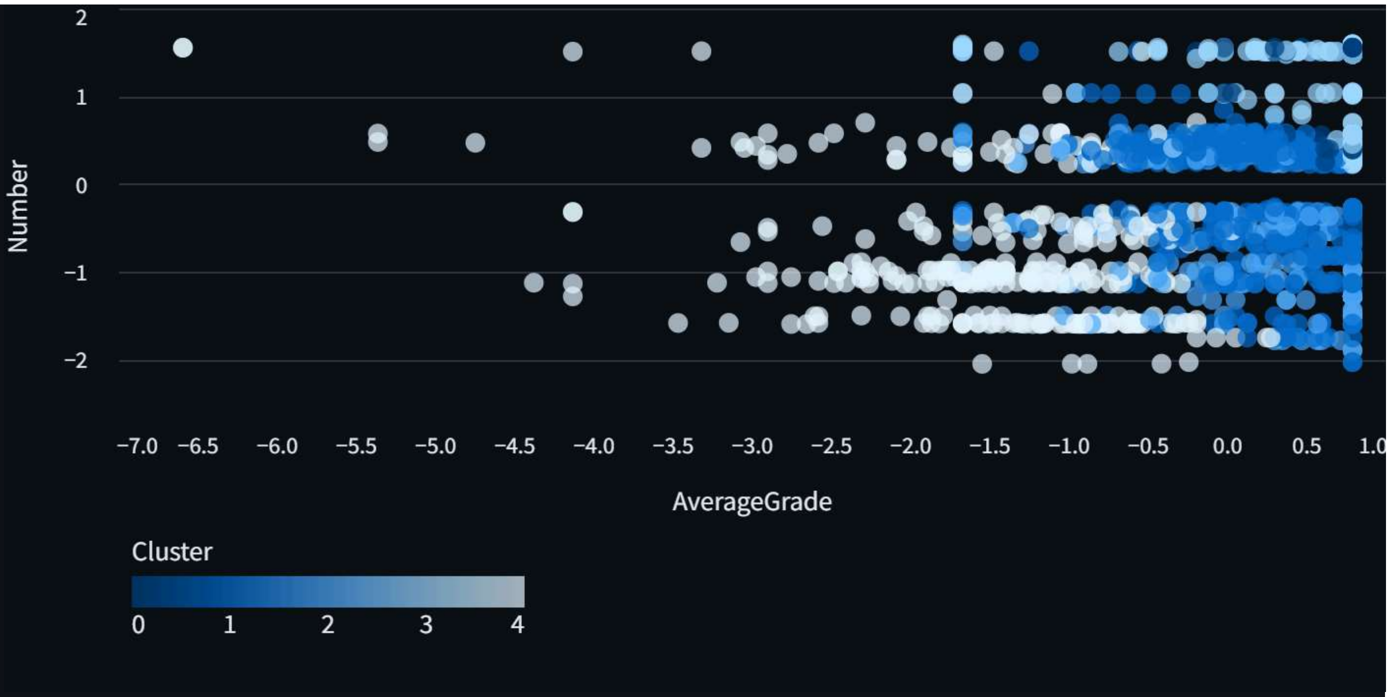
Gantt Chart

You can view our Gantt chart here.

Raw data



Scaled data



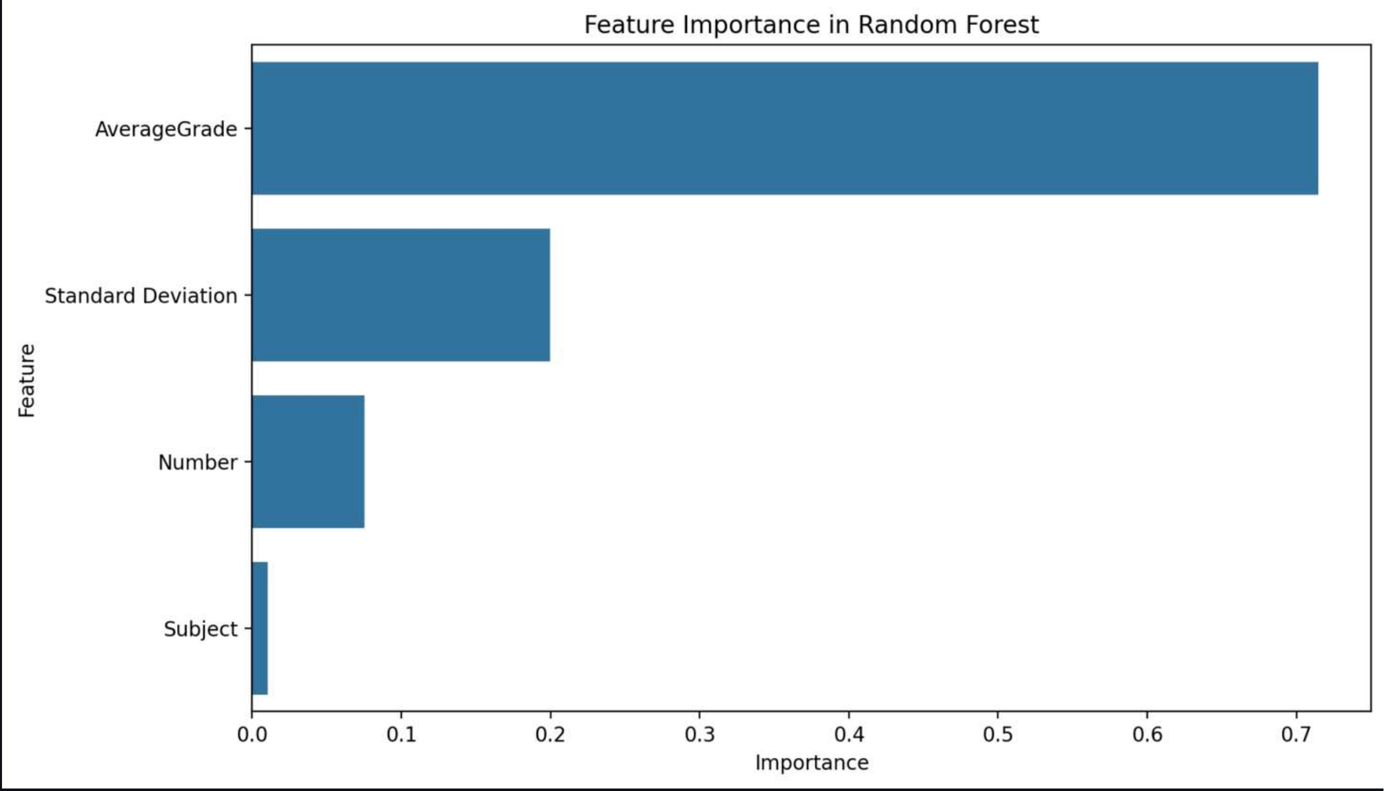
Clustering Evaluation Metrics

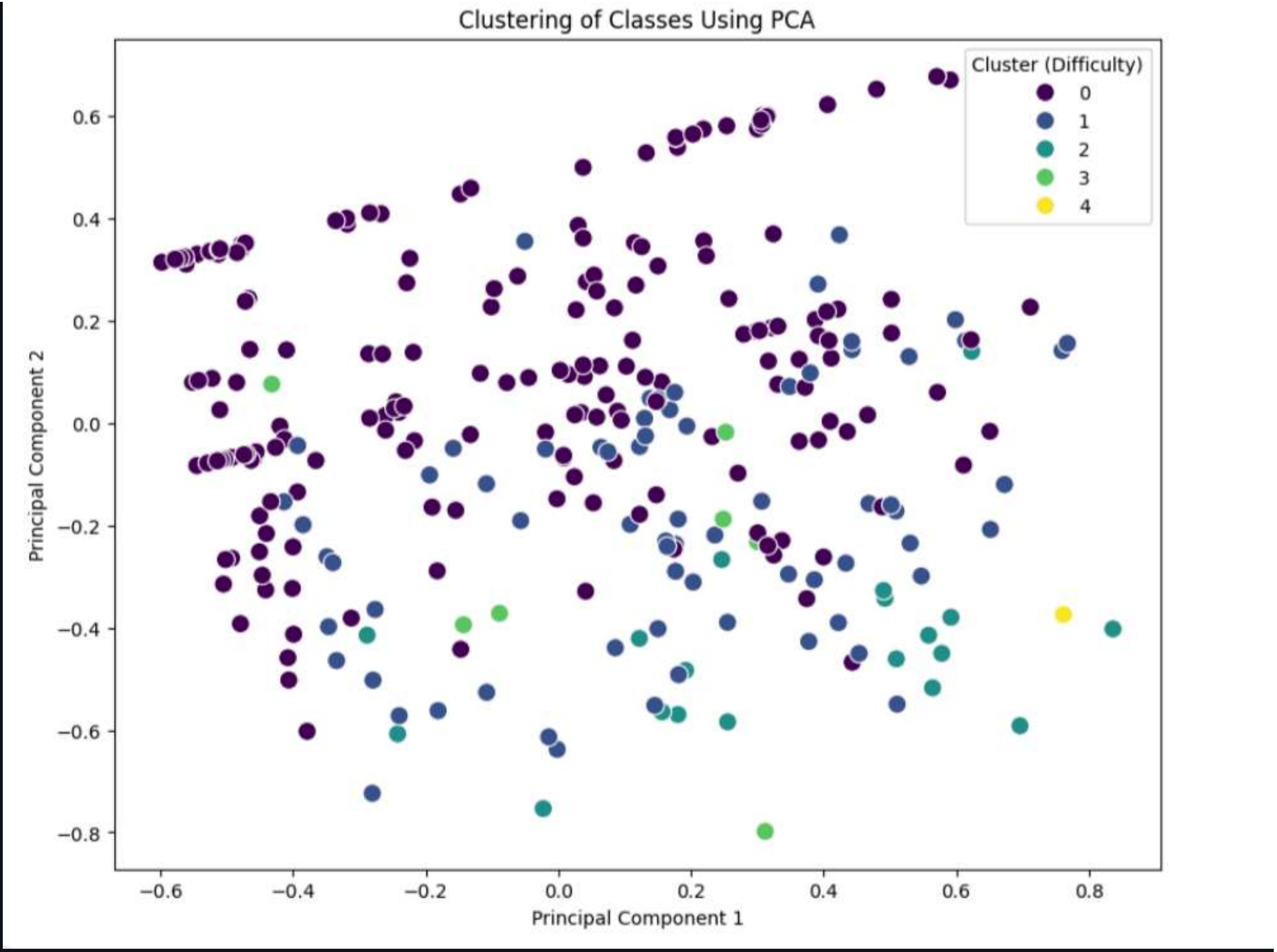
Fowlkes-Mallows score: 0.4170418473592

Silhouette score: -0.04189836447934974

Random Forest

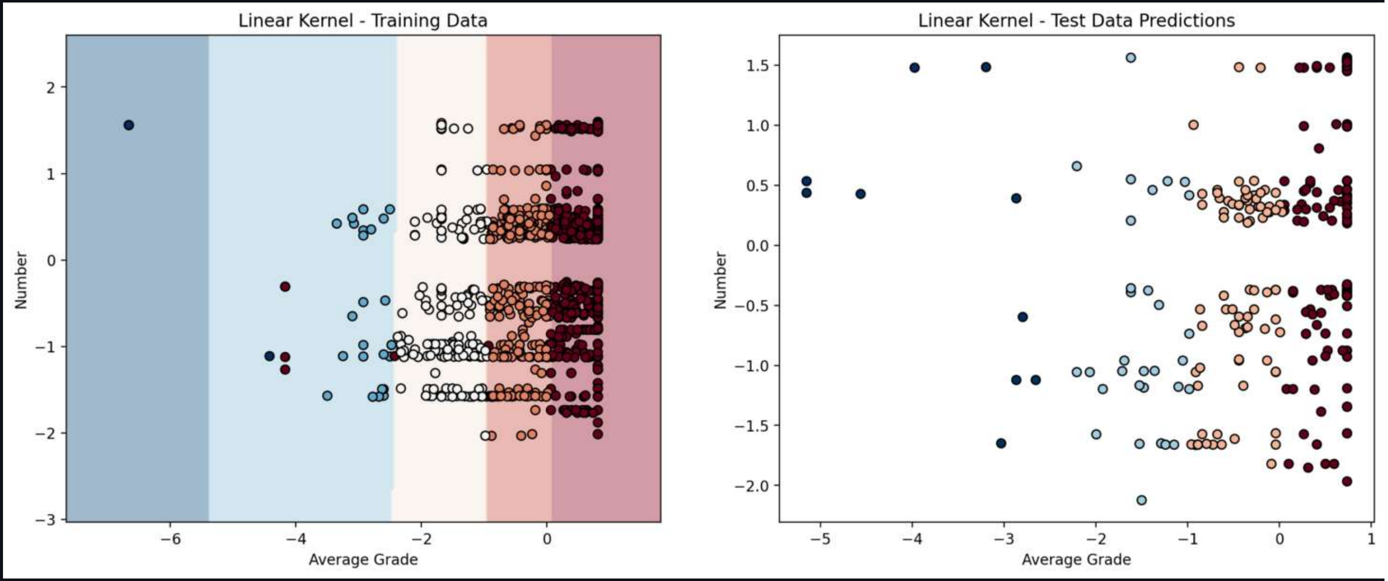
```
Test Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94       242
           1       0.63      0.81      0.71        54
           2       0.42      0.89      0.57         9
           3       0.57      0.80      0.67         5
           4       1.00      0.33      0.50         3

    accuracy                           0.87       313
   macro avg       0.72      0.75      0.68       313
weighted avg       0.91      0.87      0.88       313

Test Weighted F1 Score: 0.88
Feature Importances:
              Feature  Importance
0        AverageGrade    0.640251
1  Standard Deviation    0.167567
3              Number    0.110049
4      Instructor_bin    0.059286
2             Subject    0.022846
```



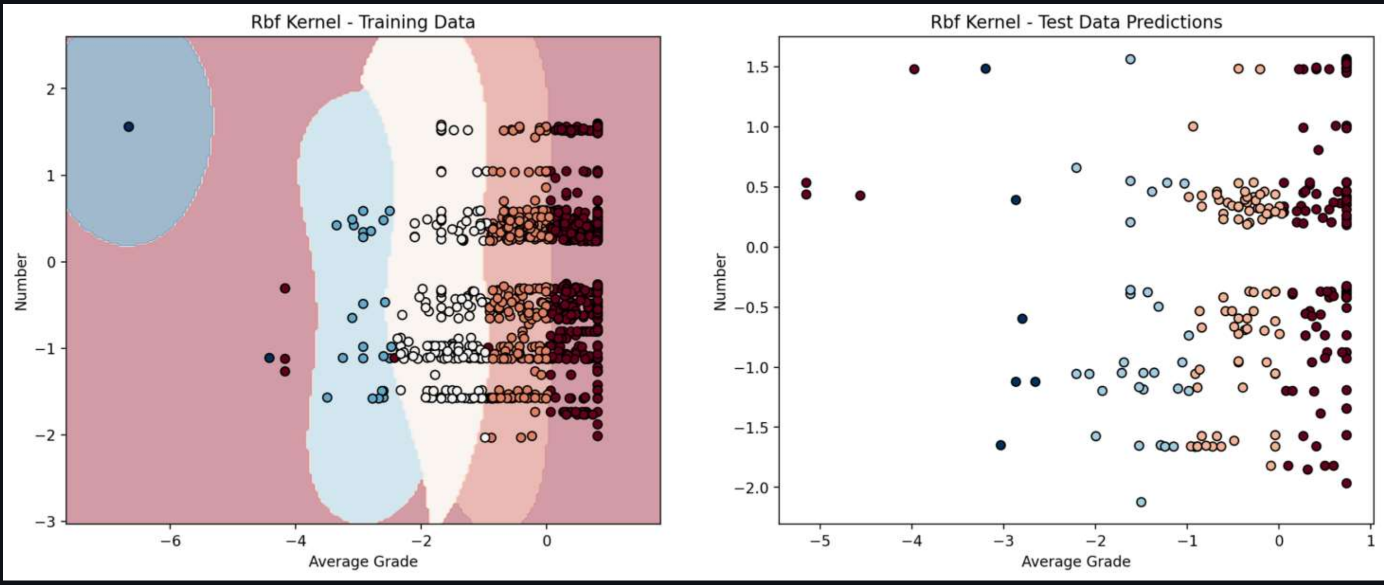


SVM Kernel Comparison

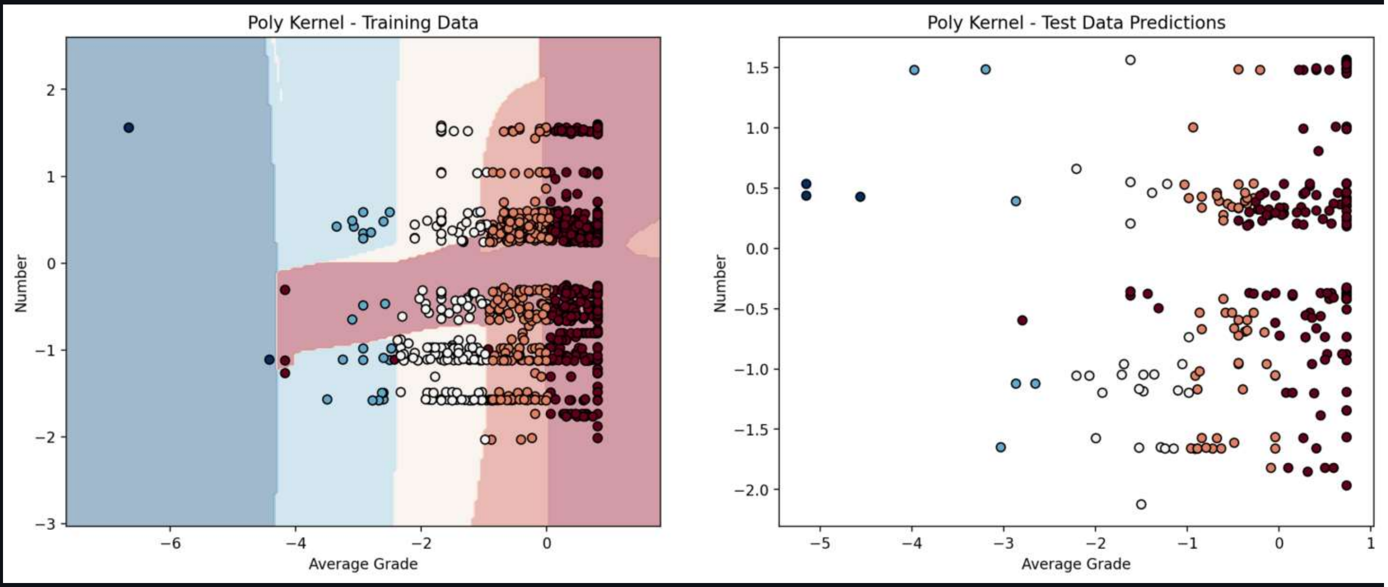
Linear Kernel



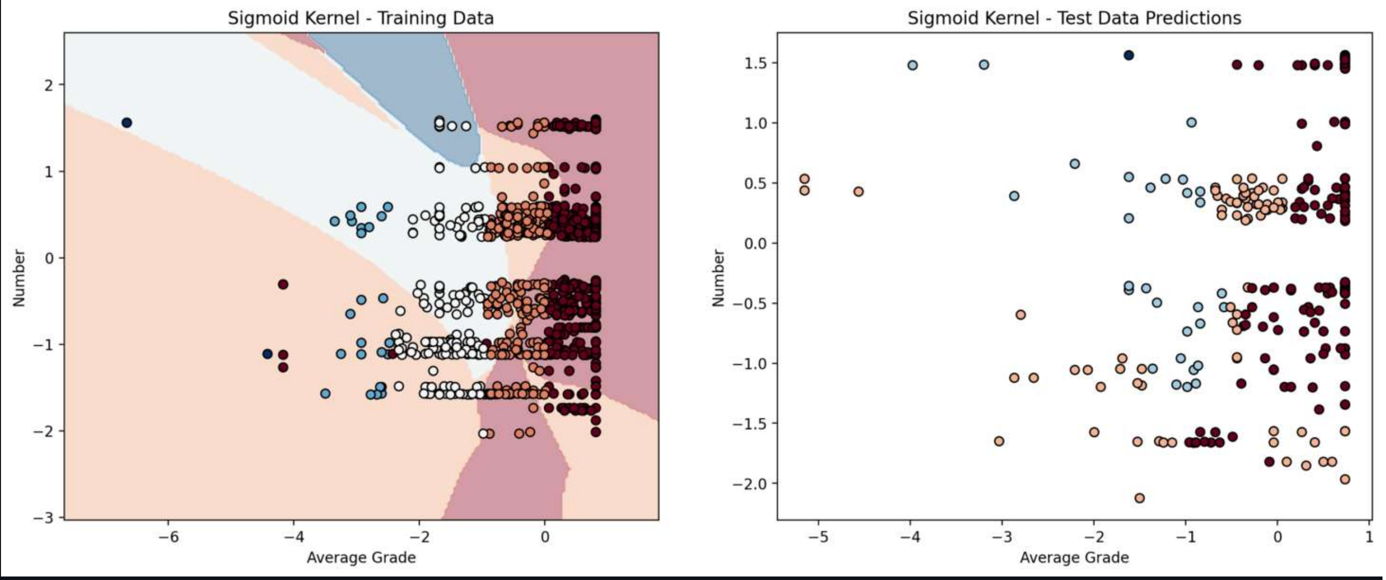
Rbf Kernel



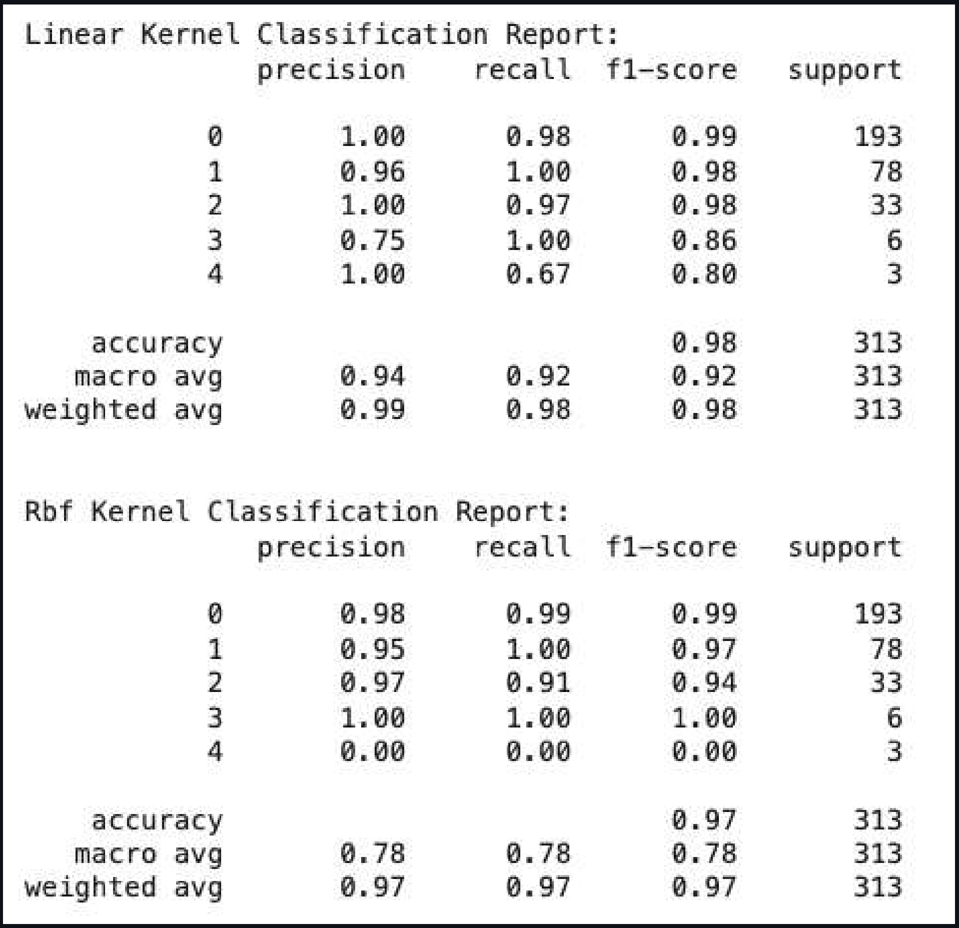
Poly Kernel



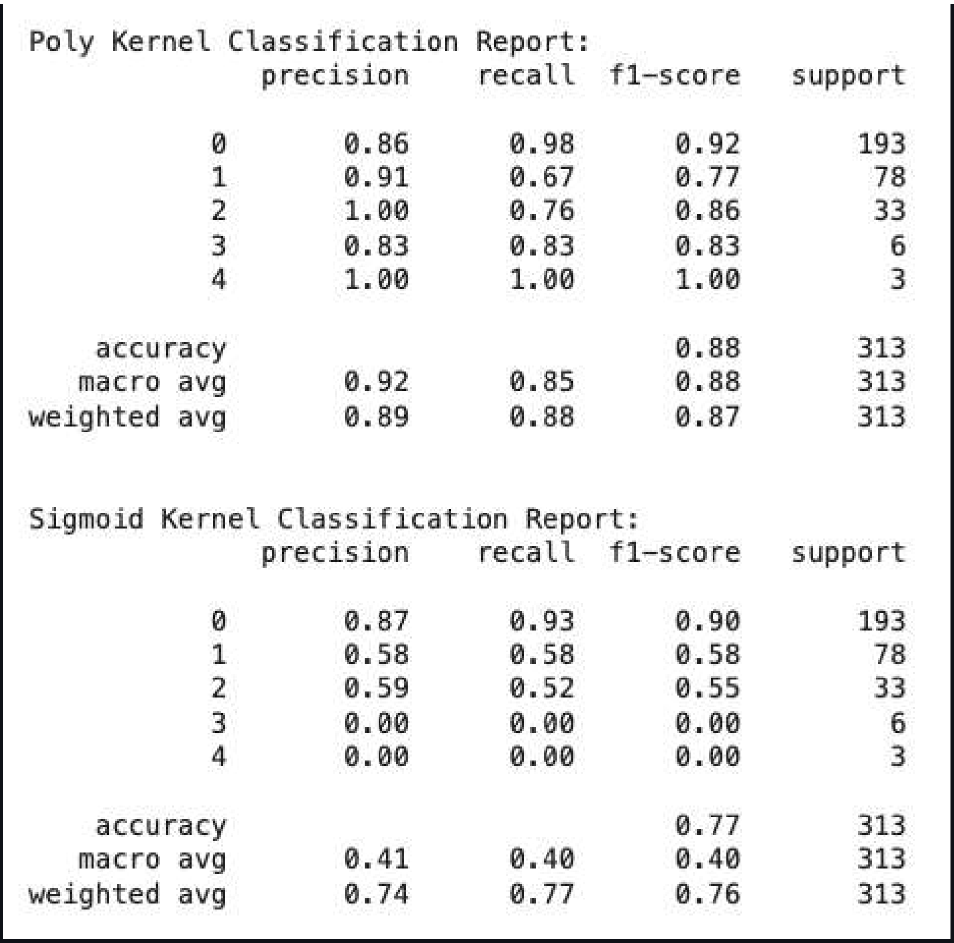
Sigmoid Kernel



Classification reports



```
Linear Kernel Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       193
           1       0.96      1.00      0.98        78
           2       1.00      0.97      0.98        33
           3       0.75      1.00      0.86         6
           4       1.00      0.67      0.80         3

    accuracy                           0.98       313
   macro avg       0.94      0.92      0.92       313
weighted avg       0.99      0.98      0.98       313


Rbf Kernel Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       193
           1       0.95      1.00      0.97        78
           2       0.97      0.91      0.94        33
           3       1.00      1.00      1.00         6
           4       0.00      0.00      0.00         3

    accuracy                           0.97       313
   macro avg       0.78      0.78      0.78       313
weighted avg       0.97      0.97      0.97       313
```



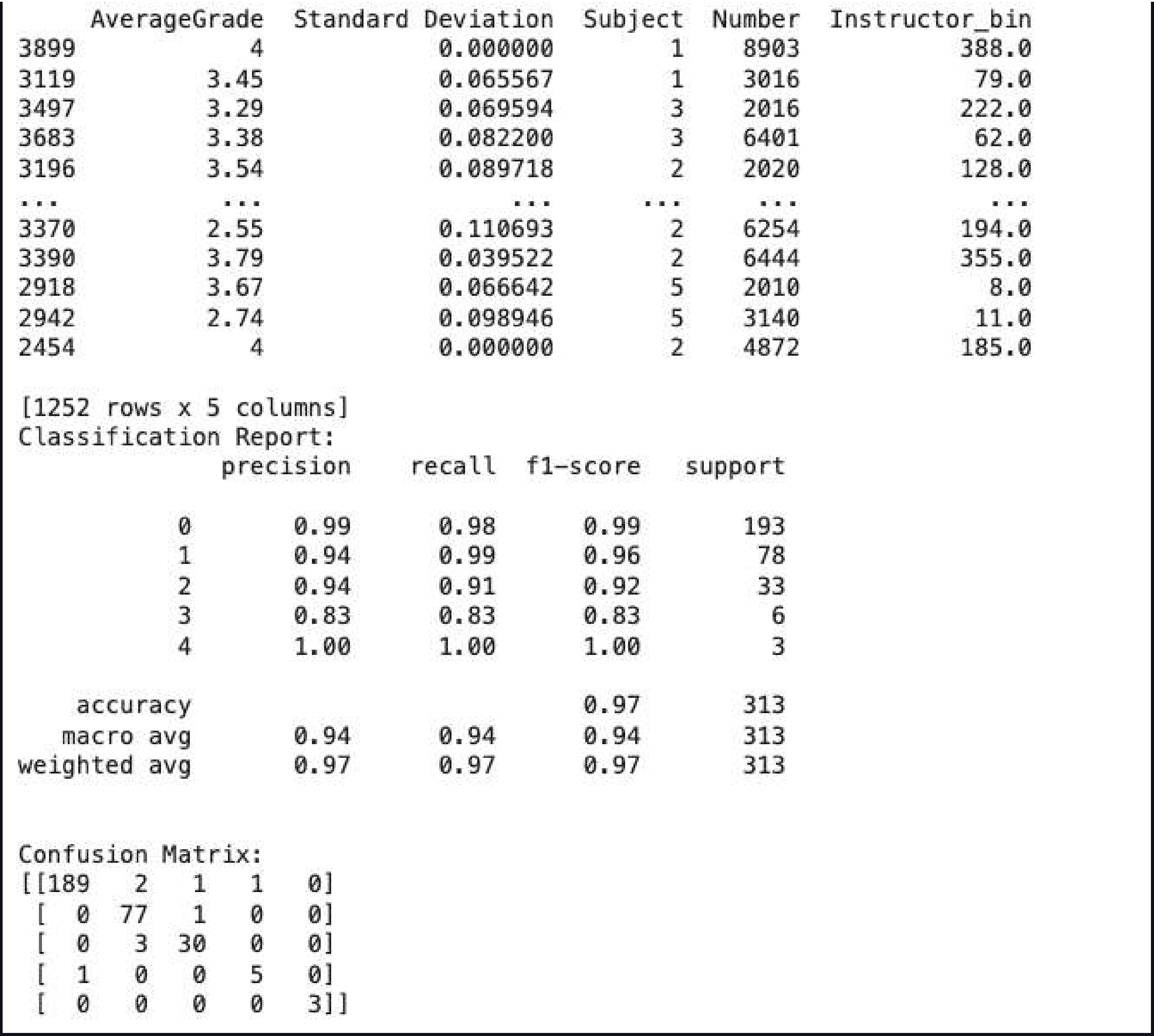
```
Poly Kernel Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.98      0.92       193
           1       0.91      0.67      0.77        78
           2       1.00      0.76      0.86        33
           3       0.83      0.83      0.83         6
           4       1.00      1.00      1.00         3

    accuracy                           0.88       313
   macro avg       0.92      0.85      0.88       313
weighted avg       0.89      0.88      0.87       313


Sigmoid Kernel Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.93      0.90       193
           1       0.58      0.58      0.58        78
           2       0.59      0.52      0.55        33
           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00         3

    accuracy                           0.77       313
   macro avg       0.41      0.40      0.40       313
weighted avg       0.74      0.77      0.76       313
```

Neural Network Classifier

Scaled Training Data:

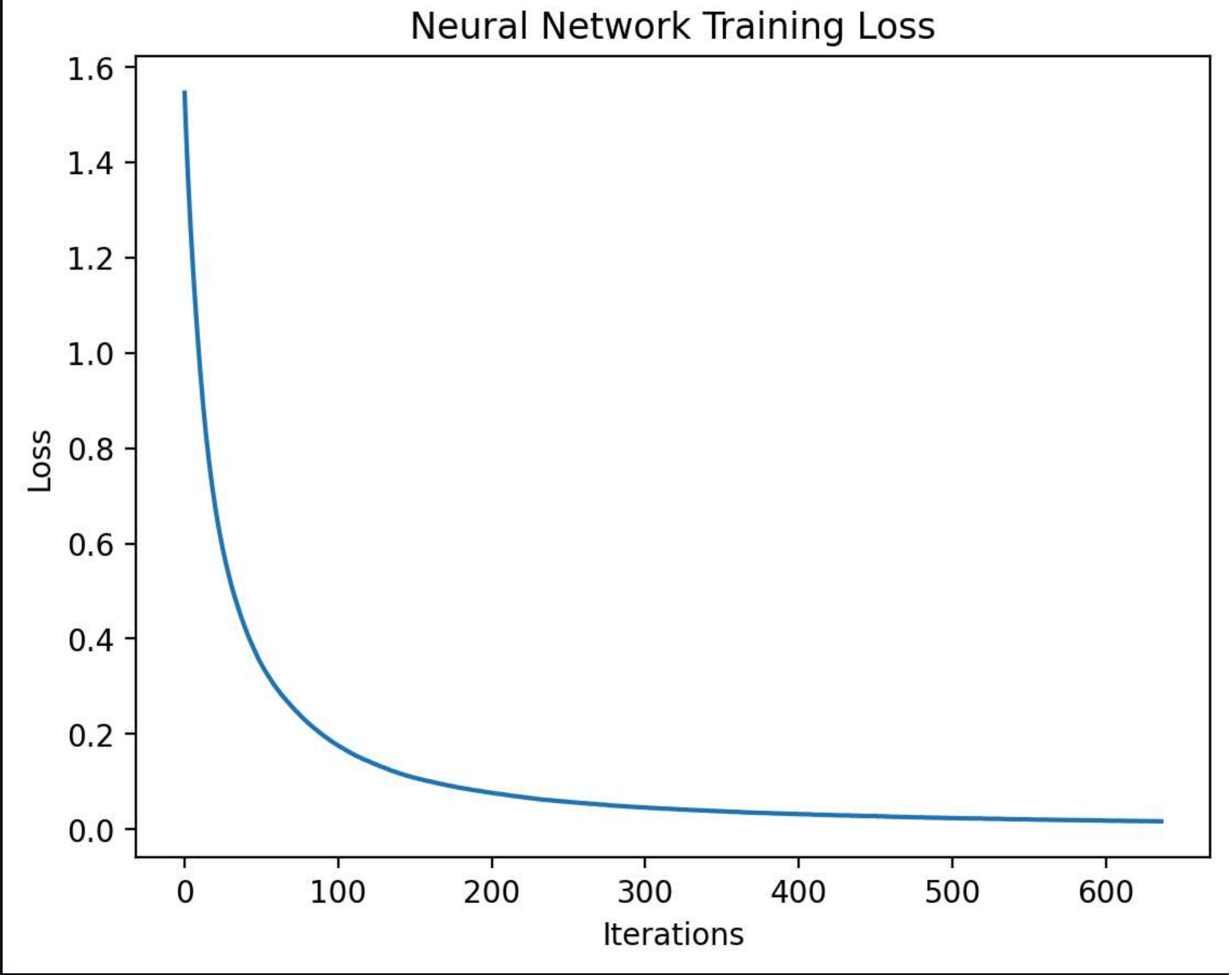
	AverageGrade	Standard Deviation	Subject	Number
0	0.8111	-1.025	-1.4298	1.5605
1	-0.5589	0.7256	-1.4298	-1.1226
2	-0.9575	0.8331	-0.0453	-1.5784
3	-0.7333	1.1697	-0.0453	0.4202
4	-0.3347	1.3704	-0.7376	-1.5765
5	0.4126	-0.0511	-0.7376	0.4471
6	0.8111	-1.025	-1.4298	1.515
7	0.8111	-1.025	-0.0453	1.5195
8	0.3877	0.7323	-0.0453	-1.7361
9	0.014	0.4502	1.3392	0.2602

Classification Report:



```
     AverageGrade  Standard Deviation  Subject  Number  Instructor_bin
3899            4            0.000000        1    8903           388.0
3119         3.45            0.065567        1    3016            79.0
3497         3.29            0.069594        3    2016           222.0
3683         3.38            0.082200        3    6401            62.0
3196         3.54            0.089718        2    2020           128.0
...           ...                 ...      ...     ...             ...
3370         2.55            0.110693        2    6254           194.0
3390         3.79            0.039522        2    6444           355.0
2918         3.67            0.066642        5    2010             8.0
2942         2.74            0.098946        5    3140            11.0
2454            4            0.000000        2    4872           185.0

[1252 rows x 5 columns]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       193
           1       0.94      0.99      0.96        78
           2       0.94      0.91      0.92        33
           3       0.83      0.83      0.83         6
           4       1.00      1.00      1.00         3

    accuracy                           0.97       313
   macro avg       0.94      0.94      0.94       313
weighted avg       0.97      0.97      0.97       313


Confusion Matrix:
[[189   2   1   1   0]
 [  0  77   1   0   0]
 [  0   3  30   0   0]
 [  1   0   0   5   0]
 [  0   0   0   0   3]]
```

Neural Network Training Loss Curve:



Feature Importance (absolute mean of first layer weights):

	Feature	Importance
0	AverageGrade	1.0169
3	Number	0.3725
2	Subject	0.2846
1	Standard Deviation	0.2635