CS 171

Final Project Report

Keerthy Reddy

Katherine Yee

CS 171

Professor Fabio Di Troia

May 19, 2025

Contents

1 Introduction

Students often have many different familial, social, financial, and personal responsibilities and habits while in school, so being successful in the classroom can be a challenge when juggling multiple commitments at once. Additionally, with the influence of technology and social media, students have to resist more distractions than ever in order to set aside time to attend class, study for a test, or complete homework. Therefore, a student's academic success cannot only be determined by their performance in the classroom, but must take a well-rounded view of the student's life. This includes taking their personal habits, mental and physical health, activities inside and outside of the classroom, and at-home conditions into consideration to do so. For this project, our team decided to examine how these common factors of student life can impact the test scores of a student in order to understand how well these factors can predict the student's success or failure on a given exam. This research can inform educators and school faculty on how to provide the appropriate resources and assistance that students require in order to manage their time and get the support they need throughout the school year. This project uses a dataset of 1,000 students with over 15 common factors that affect student life to perform this prediction analysis using three different types of machine learning models (Sequential, Support Vector Classifier, and K-Nearest Neighbors) , each with hyperparameters that we experimented with and refined for increased prediction accuracy. This report will walk the reader through the preprocessing, model architecture decisions, training strategy, and performance evaluation on the experiments conducted and compare the results to determine the most accurate model to use for this dataset.

2 Dataset Analysis

Our team used the "Student Habits vs Academic Performance" dataset (https://www.kaggle.com/datasets/jayaantanaath/student-habits-vs-academic-performance) from Kaggle in order to conduct our experiments. This dataset contains 16 different columns, each representing a different factor with data about the student. These columns include: **student_id** (numerical student ID), **age** (numerical age of student), **gender** (student can identify as "Male", "Female", or "Other"), **study_hours_per_day** (numerical value of average number of hours a student studies per day), **social_media_hours** (numerical value of average number of hours a student spends on social media per day), **netflix_hours** (numerical value of average number of

hours a student watches something on Netflix per day), **part_time_job** ("Yes" or "No" values to represent whether or not a student has a part time job), **attendance_percentage** (numerical value of the percent of total classes that a student has attended in the school year), **sleep_hours** (numerical value of the average number of hours a student sleeps per night), **diet_quality** (student can have a "Poor", "Fair", or "Good" quality of food consumed every day), **exercise_frequency** (numerical value of the average number of times a student exercises per week), **parental_education_level** (student's parents's highest level of education completed can be "High School", "Bachelor", or "Other"), **internet_quality** (student can have access to "Good", "Average", or "Other" internet quality at home), **mental_health_rating** (numerical value of a student's rating of their mental health), **extracurricular_participation** ("Yes" or "No" values to represent whether or not a student participates in extracurricular activities), and **exam_score** (numerical value of the student's final exam score). The dataset captures data for all of these factors for 1,000 students without any missing data.

Since the goal of this project is to examine how different combinations of the factors listed above will impact student test scores, it is important to understand the degree of influence that each singular factor has on the exam scores prior to experimenting on this data.

2.1 Balancing the Data

To begin this analysis, we start by balancing the dataset to ensure that we are analyzing the same or similar number of students who pass the final exam as those who fail the final exam. This prevents the model from becoming skewed by receiving an unproportionate amount of data for one result versus the other and consequently learning how to predict one result much better than the other, instead allowing it to learn and train without bias.

The process of balancing the dataset was done using Python's Pandas library to read the exam_score column in the dataset, while counting the number of students who scored above 70, indicating a passing score, and the number of students who scored below 70, indicating a failing score. These counts were stored in corresponding variables called above_70 and below_70. Based on the values of these variables, we realized that there was a very similar number of students who passed and failed the exam, meaning that the dataset was already sufficiently balanced and we did not remove any more student records from the dataset.

Number of students with passing exam score >= 70: 511

Number of students with failing exam score < 70: 489

2.2 Correlation Heatmap

The heatmap below shows the correlation between various student lifestyle habits and their final exam scores. The strongest positive correlation is between study hours per day and exam score (0.83). This correlation indicates that more study time is closely linked to better academic performance. Mental health rating also shows a positive correlation (0.32), and suggests that students with better mental well-being tend to score higher. However, social media and Netflix usage have slight negative correlations with exam scores (−0.17). This could imply potential distractions but is not strongly predictive.

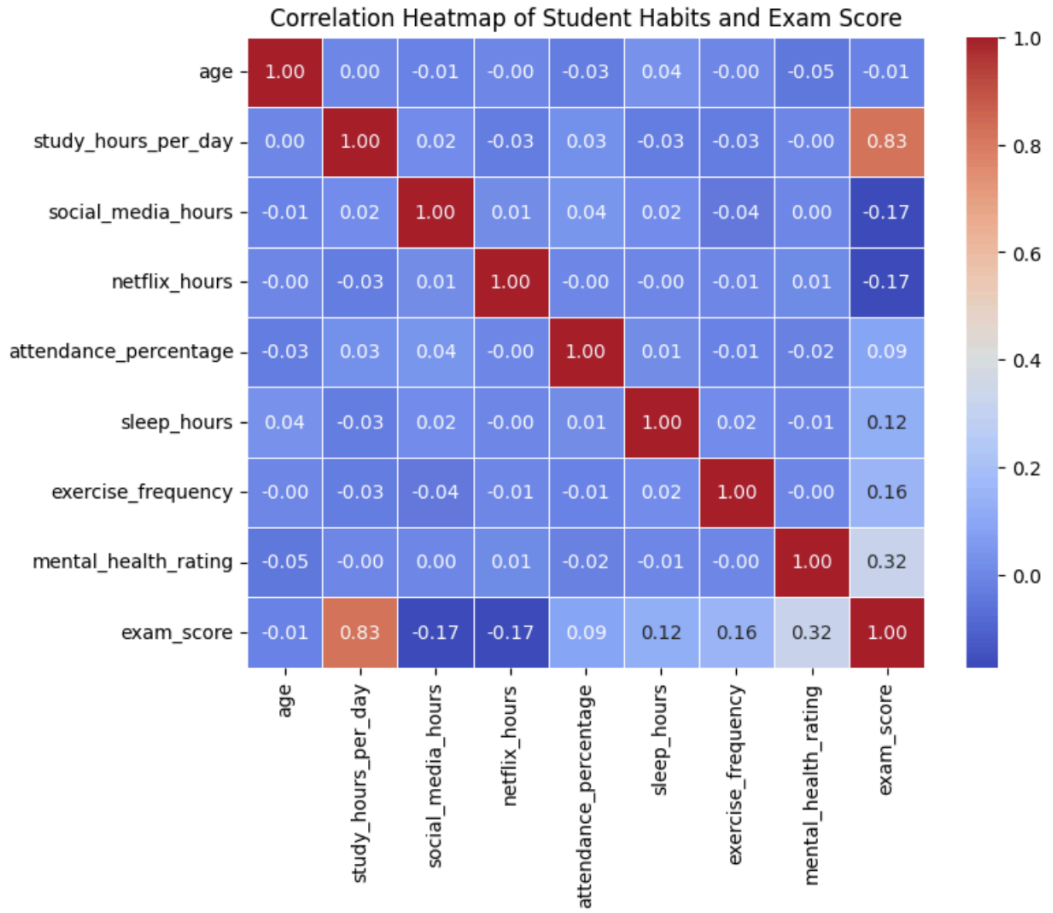

Figure 1: Correlation Heatmap of Student Habits and Exam Score

2.3 Data Preprocessing

Data preprocessing prepares and standardizes the dataset for each use case. For the following experiments in this project, our team decided to preprocess our dataset by first dropping the student_id column since the ID of a student doesn't have any influence or correlation on the student's final score, so leaving this information in the dataset would confuse the model and reduce its accuracy.

Since our dataset also includes non-numerical data in the gender, part_time_job, diet_quality, parental_education_level, internet_quality, and extracuricular_participation columns, and we want the models to only receive numerical data, we decided to use one-hot encoding to convert the form of the data. The process of one-hot encoding converts data to binary values (1 or 0) based on whether or not a certain word is present in each record. For example, since the gender column contains three possible values "Female", "Male", or "Other", it would be split into three different columns called "gender_female", "gender_male", and "gender_other" and if a student identifies as female, the gender_female column would be assigned a "1" and the other two columns would be assigned "0"s.

Since the goal of the experiments is to predict if students pass or fail the final exam rather than predicting their numerical score and we have already determined that an exam score over 70 is passing and a score below 70 is failing, we decided to change the exam_score column so that it had binary targets. This means that students who passed the exam would be assigned a "1" in the exam_score column and students who failed the exam would be assigned a "0" in this column, providing the model with the output format that we expect from the experiments for easier training and testing processes.

Our next preprocessing step was to scale the entire dataset in order to normalize it and ensure that the sizes of the range for each column in the dataset do not impact the model's accuracy. For example, the study_hours_per_day column has a much wider range of possible values than the part_time_job column, which only has two possible values, so the process of scaling would scale down the significance of each possible option in the study_hours_per_day and scale up the significance of each possible option in the part_time_job column so they are considered at equal value by the model during the training phase.

The final step of preprocessing was to shuffle all of the student records in the dataset so that the model did not learn the order of the records in the dataset and understood that it was

irrelevant to the training. This was done by randomizing the indices of the 1,000 records and then matching records to those indices to randomize the records as well.

2.4 Splitting Data

All of the machine learning models used in the following experiments go through three phases: training, validation, and testing. The entire dataset is split into three parts so that each phase works with a different subset of the original dataset. In this case, we decided to split the dataset so that the training set has 800 data points, which is the largest number of data points, in order to ensure that the model has a lot of data to learn from and identify trends and generalizations from. Then, the dataset allocated for use during the validation phase has 100 data points, which are used by the model to repeatedly check its learnings and adjust based on the results from validation. Finally, the dataset allocated for use during the testing phase has 100 data points, which are used by the model to test its learnings for a final time, with entirely new data, to make sure it isn't overfitting to the data in the training set, but is also able to apply its learnings to new data with a similar amount of accuracy.

	Passing Students	Total Targets	Pass Rate
Training	407	800	0.50875
Validation	50	100	0.5
Testing	54	100	0.54

Table 1: Class Distribution Statistics for Training, Validation, and Testing

3 Experiments

3.1 Sequential Model

This Sequential neural network is designed to predict whether a student will pass or fail an exam based on their lifestyle habits and background factors. The model includes two hidden layers, each with 20 neurons, using the ReLU activation function to capture complex patterns in the data. After each hidden layer, a dropout layer with a rate of 0.3 randomly deactivates 30% of the neurons to reduce overfitting. The output layer consists of a single neuron with a sigmoid

activation function, which outputs a probability between 0 and 1 representing the likelihood that a student will pass (a exam score of 70 or higher). The model uses the Adam optimizer with a learning rate of 0.001 and is trained using binary cross-entropy loss, which works well for pass/fail classification problems. Training is done in batches of 50 students over a maximum of 100 epochs. However, the model includes early stopping that monitors validation loss and stops training if no improvement is seen for three consecutive epochs. This helps ensure that the model does not overfit the training data and can make accurate predictions on unseen students.

These are the following baseline hyperparameters in this model:

Hyperparameter	**Value**
Hidden Layers	2
Hidden Layer Size	20
Dropout	0.3
Activation Function	ReLU
Optimizer	Adam
Loss Function	Binary Cross Entropy
Learning Rate	0.001
Batch Size	50
Early Stopping Patience	3

Table 2: Baseline Model Hyperparameters - Sequential Model

Below is the code used to implement the baseline Sequential model to predict whether a student would pass or fail the exam based on their input features. The model is built using TensorFlow's Keras API, and it follows a standard architecture for binary classification tasks.

```python
input_size = train_inputs.shape[1]
output_size = 1
hidden_layer_size = 20

model = Sequential([
    Dense(hidden_layer_size, activation='relu', input_shape=(input_size,)),
    Dropout(0.3),
    Dense(hidden_layer_size, activation='relu', input_shape=(input_size,)),
    Dropout(0.3),
    Dense(output_size, activation='sigmoid')
])
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
batch_size = 50
max_epochs = 100

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3
    restore_best_weights=True,
    verbose=1
)
history = model.fit(
    train_inputs, train_targets,
    epochs=max_epochs,
    batch_size=batch_size,
    validation_data=(validation_inputs, validation_targets),
    callbacks=[early_stopping],
    verbose=2
)
```

After training, the baseline model achieved a test accuracy of 86.50% and a final loss of 0.3189. The loss indicates the average difference between the predicted probabilities and the true labels. The classification results showed strong, balanced performance across both classes (pass and fail), with the model achieving a precision and recall above 0.80 for each. These results confirm that the baseline neural network was able to effectively capture meaningful patterns in the student data and make reliable predictions regarding academic success. To improve the

performance of the baseline Sequential model, several key hyperparameters were tuned and evaluated. Each hyperparameter influences how the neural network learns and generalizes, and adjusting them can lead to better model accuracy, faster convergence, or improved robustness against overfitting. The table below outlines the values tested for each hyperparameter.

Hyperparameter	**Values Tested**	**Results**
Hidden Layers	1, 2, 3	**1** → Total Test Accuracy: 81.57% (Total Test Loss: 0.3253) **2** → Total Test Accuracy: 86.50% (Total Test Loss: 0.3189) **3** → Total Test Accuracy: 84.66% (Total Test Loss: 0.3429)
Hidden Layer Size	10, 20, 50	**1** → Total Test Accuracy: 82.49% (Total Test Loss: 0.3484) **20** → Total Test Accuracy: 86.50% (Total Test Loss: 0.3189) **50** → Total Test Accuracy: 89.57% (Total Test Loss: 0.2918)
Dropout	0, 0.3, 0.5	**0** → Total Test Accuracy: 83.10% (Total Test Loss: 0.3495) **0.3** → Total Test Accuracy: 86.50% (Total Test Loss: 0.3189) **0.5** → Total Test Accuracy: 82.17% (Total Test Loss: 0.3013)
Activation Function	ReLU, tanh	**ReLU** → Total Test Accuracy: 86.50% (Total Test Loss: 0.3189) **tanh** → Total Test Accuracy: 84.95% (Total Test Loss: 0.3164)

Table 3: Model Hyperparameter Testing - Sequential Model

The number of hidden layers determines the depth of the network and its ability to learn hierarchical patterns. When tuning the number of hidden layers, we observed that a single hidden layer resulted in underfitting, with an accuracy of 81.57% and loss of 0.3253, while adding a third layer increased complexity but slightly reduced performance (84.66% accuracy, 0.3429 loss). The best balance was achieved with two hidden layers, which offered a strong trade-off between learning capacity and generalization.

The size of each hidden layer, or the number of neurons per layer, affects the model's learning capacity. Smaller sizes such as 10 and 20 neurons led to lesser performance, with accuracies of 82.49% and 86.50%. Increasing the size to 50 neurons per layer resulted in the best accuracy of 89.57% and the lowest loss of 0.2918. This showed that a larger network was better able to capture the complex relationships between student behaviors and exam results. This configuration allowed the network to model more complex interactions, especially when paired with regularization through dropout.

Dropout is used to prevent overfitting by randomly deactivating a fraction of neurons during training. When testing different dropout rates, we found that no dropout led to overfitting (83.10% accuracy, 0.3495 loss), while a rate of 0.3 provided the best balance between regularization and learning capacity (86.50% accuracy, 0.3189 loss). A higher dropout rate of 0.5 caused the model to underfit slightly (82.17% accuracy) and indicated too much regularization in the learning.

Lastly, the activation function used in the hidden layers was tested between ReLU and tanh. ReLU helped avoid vanishing gradient problems, especially in deeper networks. While tanh did provide smoother outputs, it resulted in slower training and a higher risk of vanishing gradients.

Overall, the hyperparameter tuning process highlighted that the best performing model had two hidden layers with 50 neurons each, a dropout rate of 0.3, and ReLU activation functions with an accuracy of 89.57% and loss of 0.2918. These settings allowed the model to learn complex patterns in the data while maintaining strong generalization on unseen examples. Future work could explore more advanced techniques like learning rate scheduling or batch normalization to further enhance model performance.

3.2 Support Vector Classifier Model

The Support Vector Classifier (SVC) model is the second machine learning model our team used to train and test this dataset. SVC learns to predict whether students pass or fail the final exam by using the data provided in the training set to place each of the students onto the feature space so that students who the model thinks will pass are placed on one side of a hyperplane and students who the model thinks will not pass are placed on the other side of the hyperplane. The hyperplane of an SVC model has margins that allow some degree of error so that some students who are placed on the wrong side of the hyperplane but within the margin limits can still be considered correct. We decided to use this model on the dataset because our goal is to choose a model that can classify data points into two classes, which SVC can do in an efficient and accurate way.

SVC has three different hyperparameters that can be tuned and refined to improve the model's accuracy: Kernel Type, Regularization Value (C), and Gamma. The Kernel Type represents the dimension of the feature space used to map the data onto, allowing different shapes and degrees of flexibility of the hyperplane based on the dimensionality. Using higher dimensionality or multi-dimensional kernels can be useful if the data being used is very convoluted and complicated so that a simple, linear hyperplane would not be sufficient to separate the two classes. The scikit-learn library used to implement the SVC model offers five kernel options: 'linear', 'poly', 'rbf", 'sigmoid', and 'precomputed'. 'linear', 'poly', and 'sigmoid' represent hyperplanes in the shape of a straight line, a polynomial function, and a sigmoid function. 'rbf', which is the standard value for the kernel hyperparameter, automatically detects the type of kernel needed and applies it to the model. 'Precomputed' requires that the developer implements the kernel into the model manually, rather than allowing the library to do it, but this will not be tested in the experiments.

The Regularization Value, represented as C in the code, is a numerical value that controls the width of the margin on either side of the hyperplane separating the two classes. As the C value increases, the width of the margin decreases and forces the model to learn to classify the data points with much less room for error. This can also lead to overfitting since the model stops trying to learn the trends in the datasheet based on its features, but instead has to learn the specific positions of each point in the dataset in order to stay within the small margins as much as possible. As the C value decreases, the width of the margin increases and allows the model to

classify data points with much more room for error error. If the C value is too small and the margin is too large however, the model can start underfitting and not learn enough about the data to make meaningful classifications.

Since the SVC model allows some of the data points to be support vectors, which are located on or within the margins on either side of the hyperparameter, these support vectors help the model decide where exactly the hyperplane should be placed in the feature space. The gamma hyperparameter is a numerical value that controls how much the hyperplane's location will be adjusted based on additional support vectors that are placed in the feature space. As the gamma value decreases, the greater the influence of the support vectors, meaning that the model only needs a few support vectors to cross their correct side of the hyperplane for its location to be adjusted. A very small gamma value can lead to the implementation of a hyperplane that is overfitting by capturing the placements of the data points in the training set much too closely, meaning that the model is not identifying generalizations for the dataset to be able to apply the hyperplane to fit new data points as well. As the gamma value increases, the smaller the influence of the support vectors, meaning that the model requires many more support vectors to cross their correct side of the hyperplane for the hyperplane's location to be adjusted. A very large gamma value can also lead to the implementation of a hyperplane that is too simple for the dataset, a sign that the model is underfitting since it cannot classify the dataset with enough accuracy and capture its complexities. The 'scale' value used for the gamma hyperparameter in the baseline model allows the model to automatically detect the most optimal gamma value to use for this specific dataset and problem statement.

These are the following baseline hyperparameters in this model:

Hyperparameter	**Value**
Kernel Type	rbf
Regularization (C)	1.0
Gamma	scale

Table 4: Baseline Model Hyperparameters - Support Vector Classifier

This is the code used to implement the SVC model using baseline parameters. It was built using Python's scikit-learn library, which contains a class called SVC that implements this machine learning model. This code also uses the .fit and .predict functions within SVC to place the data points of training set in the feature space and to make predictions based on the placements.:

```python
from sklearn.svm import SVC

svc = SVC(kernel='rbf', C=1.0, gamma="scale", class_weight='balanced')
svc.fit(train_inputs, train_targets)

svc_preds = svc.predict(test_inputs)
```

The results from a model using these baseline hyperparameters were a total test accuracy of 91%, but a 94% test accuracy for the model's classification of students who passed the final exam and an 87% test accuracy for the model's classification of students who failed the final exam. This means that 50 students out of 56 who actually passed the exam were correctly identified by the model, but 6 of those students who actually passed the exam were placed on the wrong side of the hyperparameter by the model and were classified as having failed the exam. In contrast, 41 students out of 44 who actually failed the exam were correctly identified by the model, but 3 of those students who actually failed the exam were placed on the wrong side of the hyperparameter and classified as having passed the exam. To try to improve the accuracy of this model, we will alter the values of the hyperparameters one at a time and record their results below.

Hyperparameter	**Values Tested**	**Results**
Kernel Type	linear, poly, sigmoid	**Linear** → Total Test Accuracy: 91% (96% passing accuracy, 86% failing accuracy) **Poly** → Total Test Accuracy: 88% (91% passing accuracy, 85% failing accuracy)

		Sigmoid → Total Test Accuracy: 90% (93% passing accuracy, 87% failing accuracy)
Regularization (C)	0.01, 0.1, 10, 100,	**C = 0.01** → Total Test Accuracy: 56% (56% passing accuracy, 0% failing accuracy) **C = 0.1** → Total Test Accuracy: 90% (98% passing accuracy, 83% failing accuracy) **C = 10** → Total Test Accuracy: 90% (94% passing accuracy, 85% failing accuracy) **C = 100** → Total Test Accuracy: 87% (92% passing accuracy, 82% failing accuracy)
Gamma	0.01, 0.1, 0.5, 1	**gamma = 0.01** → Total Test Accuracy: 91% (94% passing accuracy, 87% failing accuracy) **gamma = 0.1** → Total Test Accuracy: 87% (92% passing accuracy, 82% failing accuracy) **gamma = 1** → Total Test Accuracy: 61% (59% passing accuracy, 100% failing accuracy) **gamma = 10** → Total Test Accuracy: 56% (56% passing accuracy, 0% failing accuracy)

Table 5: Model Hyperparameter Testing - Support Vector Classifier

Based on the results of the experiments above, it is clear that the rbf kernel used in the baseline model was automatically using a linear kernel for the model, since the output for this experiment and the baseline model were very similar. Additionally, the original C value used in the baseline model resulted in a better accuracy than the other values tested during the experiments. Finally, we believe that the baseline model automatically detected that the most optimal gamma value was 0.01 through the 'scale' hyperparameter since the output for the baseline model and this experiment was the same. Therefore, the most optimal version of the SVC model for this data would have a linear kernel, with C = 1 and gamma = 0.01.

3.3 K-Nearest Neighbors Model

The K-Nearest Neighbors (KNN) model is a simple and effective way to predict whether a student will pass or fail an exam based on their daily habits and personal background. KNN can identify patterns by comparing each student to others with similar lifestyle profiles. For example, if a new student spends a similar amount of time studying and sleeping as students who passed, KNN is likely to predict that the new student will also pass. KNN is useful because it doesn't assume a specific relationship between inputs and the target. This is important when working with behavior data, which can be nonlinear. The KNN algorithm works by locating the "k" most similar students in the training dataset and predicting the target based on the majority outcome (pass or fail) among those neighbors. By adjusting the number of neighbors (k), we can control whether the model focuses more on the nearest students (for more personalized predictions) or a broader group (for more general patterns). In this project, our team chose to test a KNN model to provide a straightforward way to test whether simple lifestyle similarities among students can effectively predict exam success.

We began by training a baseline KNN model using k = 5, a commonly used default value that balances local sensitivity with generalization. The model was trained on the same standardized and shuffled dataset. After evaluating the performance on the test set, we extended our analysis by experimenting with different values of k to observe how the number of neighbors affects classification accuracy.

Below is the code used to implement the KNN model to predict outcomes based on the proximity of data points in the feature space. The KNN model is used to predict whether a student will pass or fail their final exam by comparing them to students with similar habits and background characteristics.

```python
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(train_inputs, train_targets)

knn_preds = knn.predict(test_inputs)
```

We tested k = 1, 3, 5, 7, 9, and 11, to observe how the number of neighbors impacts the model's performance. Smaller values like k = 1 made the model highly sensitive to noise and often resulted in overfitting by relying too heavily on individual data points. However, larger values such as k = 9 or 11 produced more stable and generalized predictions, but sometimes lost the distinctions between students who passed and those who failed. Overall, we found that k = 5 or k = 7 offered the best balance between accuracy and generalization for this dataset.

Through this testing, we were able to demonstrate how KNN can be an effective model for pass/fail prediction. While it may not outperform deep learning in terms of accuracy, it offers insights into how lifestyle similarity among students can influence academic performance predictions.

K (Neighbors)	Test Accuracy
1	0.68
3	0.68
5	0.71
7	0.71
9	0.67
11	0.71

Table 6: Test Accuracy for Each K

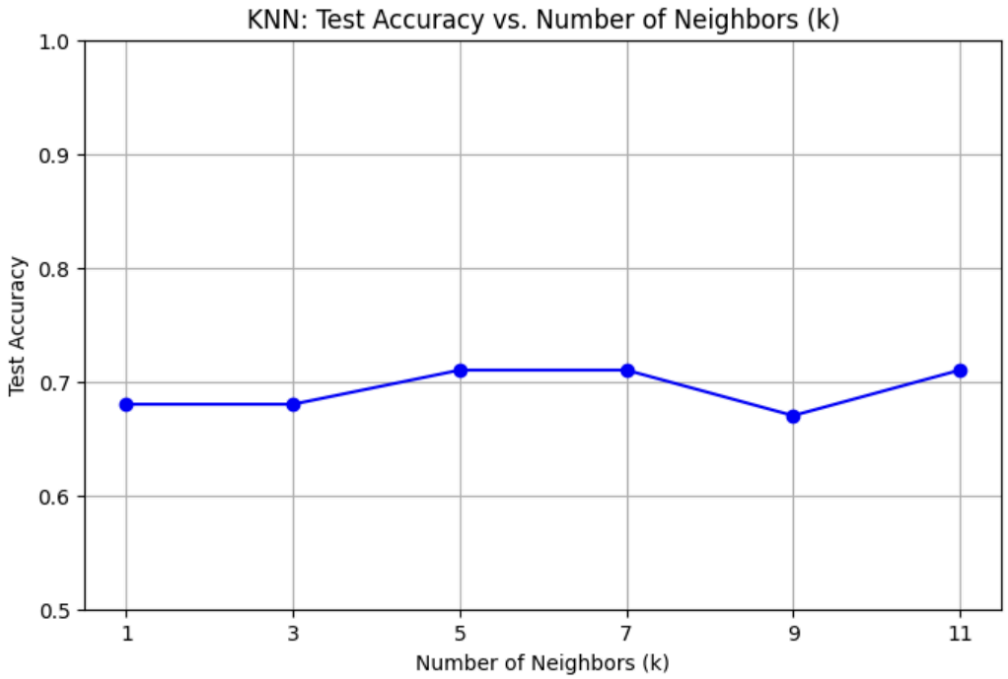

Figure 2: KNN: Test Accuracy vs. Number of Neighbors (k)

4 Results

Based on the results of the three machine learning models tested in the experiments above, our team discovered that the Support Vector Classifier (SVC) model was best suited for predicting whether or not a given student would pass or fail based on various personal and study habits using the data in this particular dataset. Among the models tested, the SVC achieved the highest overall test accuracy of 91%. This outcome aligned with our expectations during experimentation. When evaluating the Sequential Neural Network, we found that it was able to learn complex, non-linear relationships in the data, especially after tuning key hyperparameters. The final optimized Sequential model achieved a test accuracy 89.57% and a loss of 0.2918.. However, it was too complex for the dataset since it was best represented by the linear kernel of the SVC model. The K-Nearest Neighbors (KNN) model was simpler and showed lower results in comparison. Across different values of k, the model's accuracy ranged from 67% to 71%, with the highest performance at k = 5 and k = 7. However, KNN's performance was more sensitive to small variations in the number of neighbors and did not generalize as well across both classes. It also showed a tendency to underfit or overfit depending on the selected k. In general, KNN

lacked the strong generalization capabilities of the SVC and the adaptive learning capacity of the neural network.

The reason the SVC model performed the best overall likely comes down to the structure of the dataset and the decision boundary between passing and failing students. SVC models are effective in binary classification tasks where a clear margin exists between classes. In this case, the input features such as study hours, mental health rating, sleep, and class attendance appear to separate passing and failing students in a way that can be captured by the hyperplane. The SVC's ability to maximize this margin while tolerating some misclassification of the support vectors made it well-suited for this dataset. Overall, while all three models demonstrated the ability to predict student exam performance with success, the SVC model had the highest classification accuracy and demonstrated strong generalization capabilities. The Sequential model offered flexibility and high performance with tuning and the KNN model provided a simple baseline that showed patterns in the data. From experimenting with different models for this dataset, we found that the SVC model is the most effective for accurately predicting academic outcomes based on the students' lifestyle and background features.

5 Future Work

In the future our team would like to expand upon the work done for this project by testing these models with even larger datasets containing data from students with a much more diverse range or personal and study habits in order to ensure that the sequential model, which was identified as the most ideal model for this problem statement, would be able to accurately predict the results of the students' final exams. It would also be interesting to include additional factors that might influence a student's exam scores, such as the number of homework assignments turned in on time or at all, the number of times a student raises their hand or participates in class, etc.. It might be discovered, through additional experiments, that factors like this might indicate the student's success in the exam much better than the factors currently being analyzed by the models. By integrating these new variables into the existing model framework, we could better understand which behaviors are most strongly linked to academic performance, and improve the model's learning.

Since this is a predictive model, it could be implemented in a tool and used to provide interventions or targeted support for students who might be struggling with their home life and

are not able to focus fully on their school work, or to figure out study habits that work best for them. For example, the model could flag students who are at risk of failing a class and could allow teachers or advisors to offer resources, check-ins, or academic guidance. This would create a proactive support system for students and make a difference in their academics, especially if they weren't aware of their academic performance. Students would benefit greatly from having this early support and direction from educators and be able to learn so much more in ways that work best for them. Rather than waiting for a student to fail before offering help, this model would give schools the ability to act sooner. This would potentially prevent academic setbacks before they occur and students could develop stronger self-awareness and study habits that carry forward into later academic years. This project has a lot of potential to help younger generations of students to achieve academic success. With more development, this research could contribute to more supportive and personalized learning environments for students.