Prediction of American College Test (ACT) Scores

Savannah Bjorkman, Carter Durfey, Blake Martin, Daniel Perkins

CS 201R-001, Fall 2023

Department of Computer Science

Brigham Young University

Abstract

In the United States, high school students take the American College Test (ACT), a standardized test used in the college admissions process. Many are curious to know what factors affect success on the test and lead to higher scores. To gain information on this question, we gathered data from those who have previously taken the test, along with additional information regarding their demographic and high school experience and activities. Then, we used multiple machine learning models including the multi-layer perceptron, k-nearest neighbor, and decision tree to predict future ACT scores based on previous data. We were able to achieve on average a 17% improvement over the baseline mean absolute error. In our testing, the best models seemed to be the decision tree and k-nearest neighbor.

1 Introduction

The American College Test is taken by high school juniors and seniors across the United States. This test is standardized, multiple-choice, monitored, and taken with pencil and paper. In just under 3 hours, the student will complete four sections on English, Reading, Math, and Science. Some colleges also require a score from the optional section, Writing, as well. Test scores are output as whole numbers between 1 and 36. Standardized test scores are vital in the college admissions process, as it is a common data point to compare all applicants against and is often used in rewarding scholarships. For this reason, many students take the test multiple times or receive outside tutoring to get more comfortable with the format of the test, hopefully scoring higher.

Only some factors that contribute to one student’s success on the ACT are within their control. We desire to collect data to understand how much each factor (both in and outside of a student’s control) contributes or takes away from their total score.

2 Method

2.1 Data Collection

To collect data, we sent out a survey as a Google Form. We posted this survey on our social media, including Instagram, Discord, Slack, and LearningSuite, as well as sending it out to friends and groups that we have at BYU. The responses were automatically recorded in a Google Sheet.

Because the data was in a Google Form, users were able to freely answer some of the questions. This led to various typos and a few missing features. Also, many variations of the same answer were provided, which would prevent our algorithms from understanding certain patterns in the data. Therefore, we had to manually clean the data. We removed missing values, encoded features that were not numeric values, and converted SAT scores to ACT scores. We also standardized the data to make sure all responses were structured the same way.

We were able to collect 256 responses on our survey, which was reduced to 250 data points once the responses with no ACT score reported were removed. We would have liked to obtain more data, and our model probably would have benefited from having more data to learn from.

2.2 Feature Selection/Survey Creation

We came up with 26 features including the person’s ACT/SAT score. We wanted to create a mix of features that a person can and cannot control. This included information such as the student’s GPA, gender, location, social life, and parental education (see Table 1 for more detail). We also added features that we assumed would have little to no impact (such as a person’s height). This would allow our model to either confirm or deny our initial thinking.

As stated previously, the survey was created using a Google Form. The features were put into questions and either given multiple-choice options or a free-response text box based on what was expected.

*Table 1: Features Collected and Examples*

|  |  |  |
| --- | --- | --- |
| Attribute | Type of Variable | Example Data |
| BYU/UVU/Other | Nominal | BYU |
| Father's Education Level | Nominal | Bachelor's Degree |
| Mother's Education Level | Nominal | High School Diploma |
| Gender | Nominal | Male |
| High School Graduation Year | Discrete | 2018 |
| Standardized Test Tutoring | Nominal | No |
| National Honor Society Member | Nominal | Yes |
| High School State | Nominal | Tennessee |
| Language Spoken in Home | Nominal | English |
| Hispanic/Latino | Nominal | No |
| Race | Nominal | White, Asian |
| Medication Taken Throughout High School | Nominal | No |
| High School GPA | Continuous | 4 |
| Number of Siblings | Discrete | 3 |
| Number of Clubs in High School | Discrete | 5 |
| Number of Parents in the Home | Discrete | 2 |
| Number of People in Graduating Class | Discrete | 400 |
| Hours of Videogames per Week | Continuous | 10 |
| Number of Times Moved Out-of-State | Discrete | 0 |
| Height (inches) | Continuous | 72 |
| Number of Girlfriends/Boyfriends/Partners | Discrete | 1 |
| Number of AP/IB Classes | Discrete | 9 |
| Number of Times ACT/SAT Taken | Discrete | 5 |
| High School Sports | Nominal | Yes |
| Extrovert/Introvert | Nominal | Introvert |
| ACT Score | Discrete | 36 |

*This table shows the features collected, the type of variable, and an example data point.*

3 Initial Results

Since our models for predicting the ACT scores of students are regression-based, we compared them using the MAE (mean absolute error). The mean absolute error measures the average difference between the model’s prediction and the correct ACT score each student received. For example, an MAE of 0 means that the model predicts each student’s ACT score exactly with 100% accuracy, and an MAE of 1 would mean that the model’s prediction was off by 1 on average.

First, to determine the MAE that we desire, we needed to find a baseline error. We calculated the mean ACT score for our survey, which was 30.4. Then, we used the Scikit-learn mean\_absolute\_error() function to determine the MAE that a model would have if it were to predict the mean ACT score every time. This returned a value of 3.337. This means that, on average, a typical student in our data set had an ACT score of 3.337 away from the mean of 30.4. So, any model that has an MAE greater than 3.337 is highly inaccurate and not worth consideration. However, the models that provide MAEs much less than 3.337 can be accurate predictors of novel data.

For our research, we randomly split up the data into a training set (75%) and a test set (25%). Each model worked to minimize the MAE on the training data. Then, to measure the true accuracy of the models, we ran the trained models on the test set. Models that performed well on the training set but poorly on the test set are considered to overfit the data. The numbers that are the most representative are the MAEs on the test sets.

**3.1 Multi-Layer Perceptron (MLP)**

A neural network is a machine learning algorithm that attempts to model the structure of a brain. The models we used were from Scikit-learn. They had an input layer that consisted of a matrix of the data taken from our survey. Then, they had one hidden layer used to find higher-order patterns between the features. Finally, the models finished with an output layer, predicting the ACT score of each student.

The models first randomize their weights. Then, they use a weighted sum of the values of the input to calculate the values for each hidden node. Next, another weighted sum of the hidden layer is calculated to make the final prediction. Finally, the prediction is compared to the real ACT score of each instance. Through Backpropagation, this error is used, along with a logistic function, to adjust the weights, minimizing future error. Theoretically, the MLP has the potential to approximate any real-world function. However, its ability to do so highly depends on the quality of the data. Furthermore, it is common for MLPs to overfit the training set, leading to inaccurate predictions on novel data.

We initially ran the MLP, using the Scikit-learn MLP with the default hyperparameters on the data set. This yielded a training MAE of 0.139, but the test MAE was 4.22, which is much worse than the baseline MAE. This means that the model vastly overfit the data. Because of this, we experimented with various hyperparameters, starting with the learning rate, which determines how quickly the weights are updated on each iteration. We tested learning rates of 0.001, 0.01, 0.1, 1, and 10. The models with low learning rates were computationally expensive, taking a few minutes to run without even converging. However, learning rates greater than 1 made the model hard to predict, resulting in a large variance in the MAEs. So, we determined that the most optimal learning rate was about 0.1 since it converged quickly and yielded a testing MAE of 3.464 (improving the accuracy by about 20%).

We also experimented with the momentum hyperparameter. This parameter attempts to optimize the gradient descent of an MLP by updating the rates quicker whenever changes are repeated in the same direction. This makes the MAE of the model less likely to converge at a local minimum and more likely to reach the lowest possible MAE. It also makes it possible to use a low learning rate, stabilizing the learning. By default, momentum is set to be 0.9 on the Scikit-learn function. To our surprise, after testing momentum values of 0, 0.01, 0.1, 0.5, and 0.9 the MLP had the smallest MAE when the momentum was 0, so the model likely did not have any local minimums.

Finally, we investigated the optimal number of hidden nodes for the MLP. Increasing the number of hidden nodes makes the model more likely to accurately predict the ACT scores on the training set, but too many nodes make the model slow and can lead to overfitting. The table below shows the results that we received after running the MLP with various amounts of hidden nodes. The training MAE consistently decreased as the number of nodes increased, and the model yielded the best testing MAE when it had 512 nodes. This was as expected since it is about twice the number of inputs that it received.

*Table 2: Hidden Nodes on the MLE*

|  |  |  |  |
| --- | --- | --- | --- |
| Hidden Nodes | Epochs Till Convergence | Training MAE | Test MAE |
| 8 | 160 | 3.273 | 3.518 |
| 16 | 155 | 3.274 | 3.519 |
| 32 | 172 | 3.273 | 3.518 |
| 64 | 191 | 3.273 | 3.518 |
| 128 | 925 | 1.544 | 4.126 |
| 256 | 833 | 0.676 | 3.398 |
| 512 | 1761 | 0.194 | 3.329 |
| 1024 | 3046 | 0.044 | 3.378 |

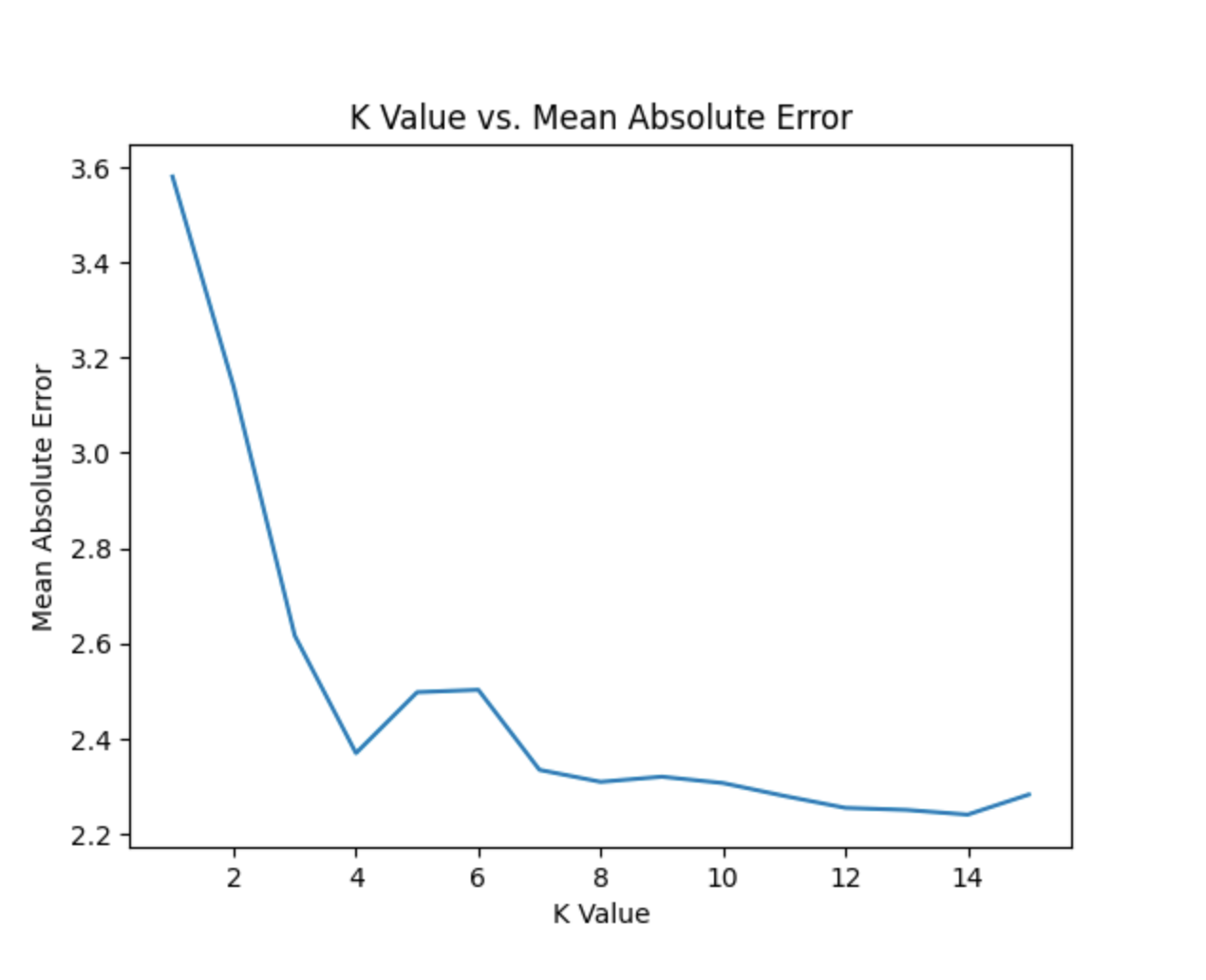
*This table shows the effects of the number of Hidden Nodes in the MLP to the MAE. As the number of hidden nodes increases, the model takes more epochs to converge but improves its training MAE. These results demonstrate that the optimal number of hidden nodes for this dataset is 512.*

Overall, the Multi-Layer Perceptron performed much worse than anticipated. At its best, it performed only slightly better than baseline accuracy. Given this, other machine learning models are likely more adequate for our goal of predicting students’ ACT scores.

**3.2 K-Nearest Neighbor**

Running the K-Neared Neighbor (KNN) model with default hyperparameters on the data, normally resulted in an MAE of around 3.00. However, after the KNN model was run 10 separate times, it became apparent that there was significant variety in the results. Some runs with default parameters output an MAE as low as about 2.75, while others output an MAE as high as almost 3.4. We believe the variability in the results was due to the smaller amount of data given to the machine to learn. If there had been more data, the output and MAE would have been more consistent.

An important decision to be made was regarding the number of neighbors to use for queries. The default parameter is set to 5. To gain more information, the KNN model was run with varying amounts of neighbors (all whole numbers between 1 and 10) on the same train test split. There was still variety in results (a continuing problem stemming from lack of extensive data), so the shape of the graph did fluctuate. However, after multiple runs, a general pattern became apparent as shown in the graph that represents the relationship between K value (number of neighbors) and MAE below. There is a significant decrease in MAE between when the k value equals 1 and 4. For k values between 4 and 5, there was a general plateau. Depending on the run, the graph would even show a small dip around the k value between 7 and 10. As a result, we decided to move forward using 8 neighbors in the model (setting the parameter ‘n\_neighbors’ to 8).

*Figure 1: KNN Model Run Using Different K Values*

*This image shows the mean absolute error for each run of the coordinating k value (number of neighbors used) on the K-Nearest Neighbor model.*

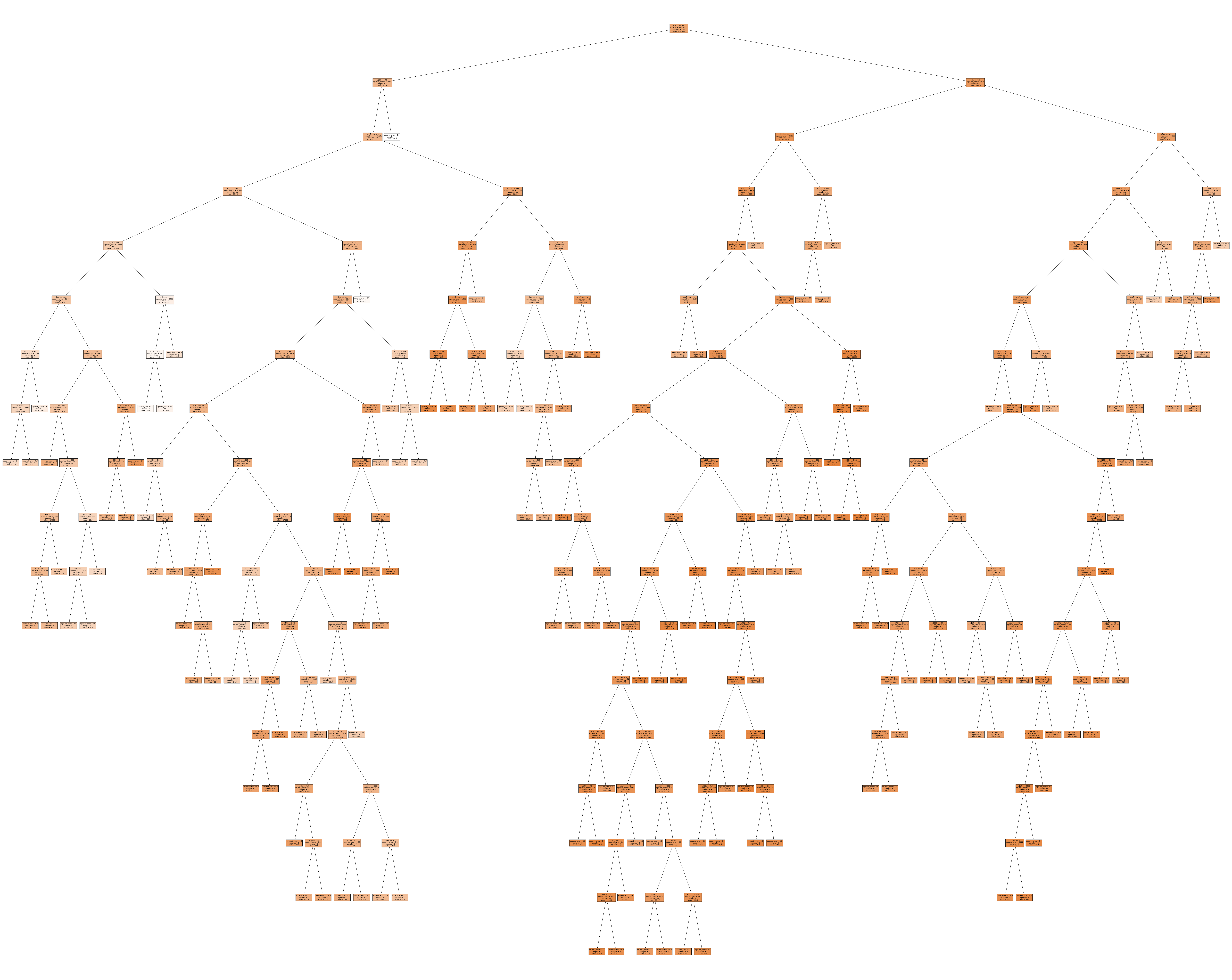
An additional aspect of the model to determine standardization while experimenting was the use of distance weighting, which is controlled through the ‘weights’ hyperparameter. When distance weighting is employed, the model considers the distance from a neighbor in its influence on output. Alternatively in uniform weighting, all neighbors contribute equally to the output. We ran the KNN model 100 times with and without distance weighting to see if there was a significant difference in MAE. With distance weighting, the MAE was 2.9297. Without distance weighting (using the default hyperparameter of ‘uniform’), the MAE was 2.9267. Therefore, there was not a significant difference in results. When the model ran without distance weighting, the average over 100 runs was only about .003. To us, this did not have intense significance. Regardless, moving forward, we used the ‘uniform’ hyperparameter (without distance weighting) on the KNN model.

**3.3 Decision Tree**

The decision tree is a machine-learning algorithm that is based on the idea of a branching tree. It works by building a predictive model in the form of a tree-like structure, where each internal node represents a decision based on a specific feature, each branch represents the outcome of that decision, and each leaf node represents the final prediction or classification.

In our first attempts to train the decision tree regressor on our ACT data, we decided to use the default parameter values to give ourselves an idea of where we were starting from. This didn’t produce great results, with the test MAE usually sitting around 3.6 which is about 9% worse than our expected baseline accuracy. The training MAE was usually pretty good—hovering around 0—which indicated that we may be overfitting the data.

Looking at the number of samples contained in each leaf node showed that we were getting a lot of leaves with very few samples which seemed to be contributing to the overfit.

*Figure 2: Decision Tree with Default Parameters*

*This image shows the decision tree with default hyperparameters visualized. Here you can see that there are a lot of leaf nodes with not very many data points, which indicates overfit.*

As can be seen from the figure above, the tree had way too many leaves for the number of data points we had, which means that the decision tree is overfitting the data. To help improve this we modified the default parameter for the minimum samples required in a leaf node (min\_samples\_leaf). By increasing this value slowly, we saw the MAE error slowly improve and we began to see major improvement eventually achieving an MAE as low as 2.41 which is roughly a 27% improvement over baseline accuracy. The best value for this parameter in our testing was 30; going up or down from there only produced worse results.

Other notable parameters that seemed to help accuracy were max depth (max\_depth), and maximum number of leaf nodes (max\_leaf\_nodes). We were able to achieve better MAE values by decreasing maximum depth with the best value being around 3. We also saw an improvement by reducing our maximum leaf nodes. The best value for this parameter seemed to be 10 leaf nodes from our testing. These seem to follow the theme that aggressive early-stopping maneuvers with the decision tree help achieve a better MAE.

*Table 3: Decision Tree Hyperparameter MAE*

|  |  |
| --- | --- |
| Hyperparameter | Mean Absolute Error |
| min\_samples\_leaf=30 | 2.41 |
| max\_depth=3 | 2.59 |
| max\_leaf\_nodes=10 | 3.11 |
| default | 3.64 |

*Minimum samples per leaf had the largest impact on the MAE while max depth and max leaf nodes also helped. Default parameters were by far the worst option.*

**4 Improvement Attempts**

**4.1 Dropping Features**

One of the first steps we took to improve our results was dropping unimportant features from our dataset. We started with 27 features, which we knew would be a lot for our models to learn with such little data. From the beginning, we removed the college column, because almost the entirety of our dataset went to BYU. We called this initial dataset containing all the other features X1.

Next, we ran the Multi-Layer Perceptron on various subsets of the features to determine which ones should not be included. We first determined that the height feature had little effect on the MAE of our models, making the MAE unpredictable. So, we removed that feature from X1 and called the new dataset X2. Then following the same process, we removed both graduation class size and graduation year, calling the new dataset X3. This seemed to vastly improve some of our models’ MAE, signifying that those features acted as noise to the model.

*Table 4: Feature Exclusion*

|  |  |  |
| --- | --- | --- |
| Features Excluded | Training MAE | Test MAE |
| None | 0.139 | 4.219 |
| Height | 0.117 | 4.413 |
| Height, Graduation Year, Graduation Class Size | 0.184 | 3.551 |

*This table shows how the training MAE and the testing MAE were affected after removing certain features. These MAEs were measured from the model’s results and the MLP with default hyperparameters.*

We also performed a principal component analysis (PCA) to condense the features into fewer features containing the most information, but unfortunately, PCA did not work as expected, and we ended up with far worse MAE on all of our models (up to 11). We also tried to remove several other features, as well as combinations of features, that we thought would be unimportant to the prediction of our models, but unfortunately, we weren’t able to find any other features that didn’t hurt our resulting MAE.

**4.2 Improved Hyperparameters**

In our study, we attempted to fine-tune hyperparameters for the various models we used, eventually achieving significant improvements. For the MLP, extensive parameter exploration led to a 16% enhancement over the baseline, resulting in a testing MAE of 3.464. K-Nearest Neighbor exhibited variability, prompting the selection of n\_neighbors as 8 with uniform weighting.

To address overfitting, Decision Tree adjustments were able to reach up to a 27% improvement, reaching a minimum MAE of 2.41. Feature selection involved omitting insignificant factors, while clustering attempts proved ineffective. Our refined models demonstrated an average 16% improvement over the baseline. We also performed random searches to attempt to find the best hyperparameters, but this didn’t yield great results, so we didn’t end up using those in the end.

**4.3 Clustering**

We attempted to cluster the data using both k-means clustering and Hierarchical Agglomerative Clustering (HAC). K-means clustering works by assigning each data point to a cluster with the nearest mean. This is generally calculated using the least squared Euclidean distance. HAC works using clustering linkage. The linkage decides how clusters should be combined. The linkages tested were single, average, complete, and ward.

Both methods proved ineffective. The efficiency of clustering was measured using a silhouette score. This score is based on how similar each data point in a cluster is to each other, and how different the cluster is from the other clusters. Our clustering attempts produced silhouette scores that are too small to consider clustering effective. In Figure 3, we see that when using clusters with four centers, the maximum silhouette score achieved is slightly above 0.1. As the number of clusters increased past four, the silhouette score dropped. Because of this, clustering was not considered for use in our final models.

*Figure 3:* A graph of a graph

Description automatically generated with medium confidence*k-means Silhouette Plot*

*This table shows the ineffectiveness of k-means clustering and the low silhouette scores achieved.*

**5 Final Results**

After determining the optimal hyperparameters for each model, we wanted to see if the decision tree continued to outperform the other models on average over several different runs. This was just to make sure we didn’t have a lucky split in our first tests, as well as to test which model overall was the most performant and the most accurate. To accomplish this, we ran each model ten times on our new set of features (X3), with different random training/test splits each time. We recorded the average training and test MAE.

The Multi-Layer Perceptron was the least effective model. It took the longest to run and had the highest testing MAE at 3.189. This was only about 5% more accurate than baseline accuracy. The K-Nearest Neighbor algorithm, on the other hand, was much quicker and more accurate. Its training MAE was almost 0, meaning that it predicted the ACT score on the training set well. Its test MAE was 2.809, which is about 16% more accurate than the baseline. Finally, the Decision Tree was the least accurate on the training set (because it used very aggressive early stopping) but the most accurate on the test set. Its MAE was 2.77, which is 17% more accurate than the baseline. This suggests that it is the optimal model for predicting student ACT scores. In our testing, it was the model that best fits the data.

*Table 5: Results*

|  |  |  |
| --- | --- | --- |
| Model | Training MAE | Test MAE |
| MLP | 1.890 | 3.189 |
| KNN | 7.327e-08 | 2.809 |
| DT | 2.588 | 2.771 |

*This table shows the results we had in our models, using an average of 10 different training and test splits. The Decision Tree proved to be the most accurate model, with a testing MAE of 2.77.*

**6 Conclusion**

In conclusion, optimization played a pivotal role. After tuning, the MLP achieved a 20% MAE improvement to 3.464. KNN, with refined parameters, saw a 16% MAE reduction to 2.809. The decision tree, with optimized hyperparameters and feature reduction, showed the most significant improvement—a 27% MAE reduction to 2.77. The decision tree was also able to achieve the lowest MAE on the problem between the models. This shows that the decision tree is the most effective model for this problem given our current dataset. Overall, both the decision tree and KNN were able to solve the problem to a certain extent, while the MLP was not able to achieve results that reliably beat the baseline accuracy.

**7 Future Work**

The next step to improve accuracy in predicting ACT scores is to collect and gather more data. For the machine-learning models to better learn the patterns in the data, it requires more data. It would also be beneficial if the data gathered was more diversified. In the original data set, largely taken from the student body at Brigham Young University, there was a lack of diversity visible both in demographics and ACT scores. Going forward, it would be helpful to have a greater diversity manifest in the dataset. This would have to be gathered from more creative methods than originally used. We hypothesize that by increasing both the size and diversity of the dataset, these models could achieve a better result.