Treyson Grange
Stockton Smith
CS 5830
Project 4

Part I:

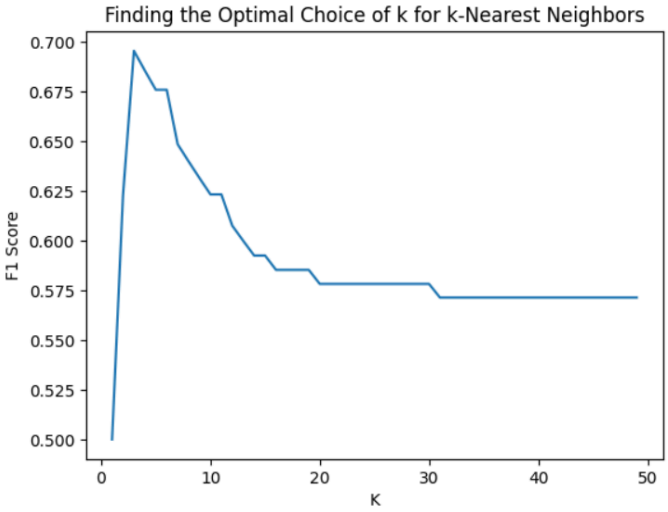
Introduction:

In this report, we explored the best k value for the k nearest neighbors method in hopes of predicting heart disease based on the Cleveland dataset. Through experimentation of different attributes and k values, we aimed to provide valuable insights into the practical application of k-NN in cardiovascular disease prediction. Possible indicators of heart disease included age, sex, cholesterol, and so on. Age indicates cumulative risk, while sex influences risk patterns, as apparently men have a higher risk of heart disease. High cholesterol increases plaque buildup, which also raises heart disease risk. This model is important and useful because if we can predict heart disease without thorough testing, we can make the process of diagnosis easier.

Methods:

We first loaded the Cleveland Heart Disease dataset and dealt with missing values by replacing them with NaNs. We then converted the target variable into binary form, indicating the presence or absence of heart disease. Next, we selected a set of features that we believed would be relevant for predicting heart disease: age, sex, chest pain, resting blood pressure, and cholesterol. After that, we standardized the features to ensure they were on the same scale. Following preprocessing, we split the dataset into training and testing sets using `sklearn.model_selection.train_test_split`.

For the model itself, we used the k-nearest neighbors algorithm and searched for the optimal value of k between 1 and 100. To evaluate the model's performance, we calculated precision, recall, and F1-score. This approach allowed us to change the inputs to search for the best k value and best attributes to use, allowing us to improve our model to be as effective as possible.



From this, we see that there is a stark difference in performance as we get a higher k value. This is interesting for a few reasons; with such a small k value, it's possible that our chosen attributes are overfitting the model. Additionally, there exists the threat of the curse of dimensionality, which means that as the number of dimensions/features increases, the "nearest" neighbors might not provide meaningful results.

For the testing data, we loaded the new dataset and handled missing values like before. Using the same features as in training, we scaled the data. Then, we applied the k-nearest neighbors model with the best k value found earlier. By considering any non-zero prediction as heart disease, we made predictions for the new dataset. Finally, we printed out the predictions to see how well our model performed.

To cross validate our data, we conducted a series of cross-validations using the nearest neighbors approach to assess the predictive performance of different combinations of features in predicting the target variable. Initially, we defined a list of attribute combinations, each representing a subset of features to be used in the analysis. We then split the data into training and testing sets for each attribute combination using the train_test_split function. Finally, we trained the nearest neighbors model on the training data and used it to predict heart disease for the test data.

Results:

As a test, a k value of 1 with every attribute being tested on resulted in the following:

Precision	Recall	F1 Score
.84	.875	.8571428571428571

These scores seem to be good on the surface; however, to allow for higher adaptability, we elected to use only the five attributes mentioned above (age, sex, chest pain, resting blood pressure, and cholesterol), resulting in an optimal k of 6:

Precision	Recall	F1 Score
.51	1	.676

These scores indicate that our model has a hard time with correct diagnoses, but excels when it comes to recall, averaging our F1 score into the 67% range. While .676 for a F1 score isn't great, it is better than flipping a coin to decide if someone has a disease.

Part II:

Introduction:

For our additional dataset, we chose to analyze factors relating to students in higher education that could potentially predict whether or not they will drop out of school before graduating. The factors we chose to analyze (which we hypothesized to be the most likely indicators out of all the provided attributes in the dataset) include their age at enrollment, whether or not their tuition fees are up to date, their average grade from their previous degree or level of education, and whether or not they are a scholarship holder. We hypothesized that our findings from this experiment would be significant enough to potentially assist school officials in identifying students who may be more likely to be in need of additional help and resources.

Dataset:

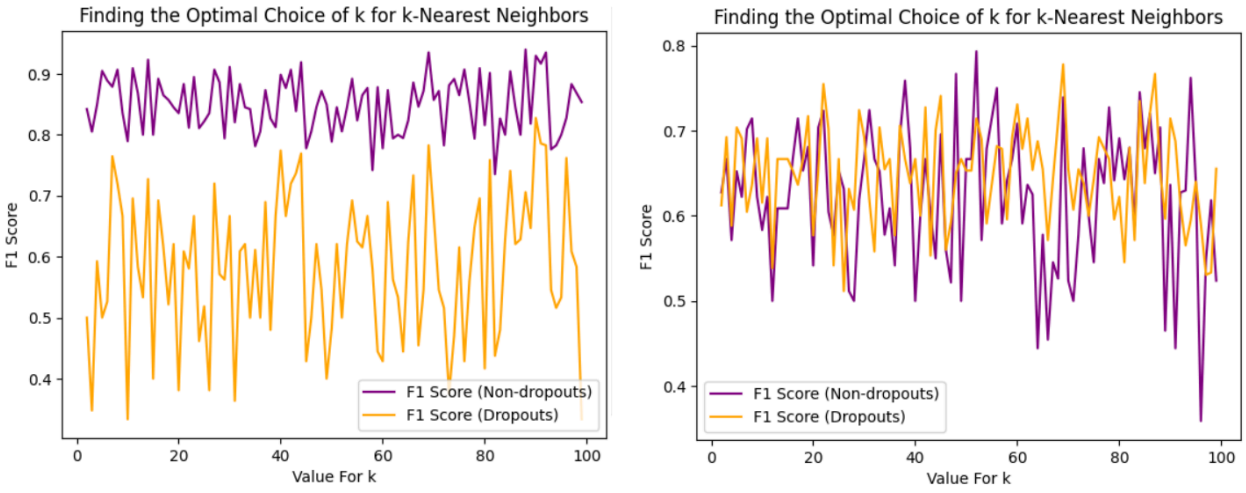
The student dropout dataset can be downloaded from the UC Irvine Machine Learning Repository (found here). The dataset is then downloaded as "data.csv" and must be placed in the "datasets" directory of our git repository. However, the dataset is separated by semicolons rather than by commas, rendering it incompatible with the pandas "read_csv" function. To account for this, we included the `sep=';'` parameter in the read_csv function call, allowing analyses to be made on the data as usual.

Methods:

After testing, training, and cross-validating 11 different combinations of our attributes, we elected to use the model that used each of the four attributes outlined above (Other high performers were the models using just age and scholarship, as well as the model using age, grades, and scholarship, but these models tended to have higher variability). At first glance, the results of our experiment seemed to show that our

suspected indicators were effective, consistently generating an F1 score of about 0.85. However, we then noticed that this F1 score was only being generated for the negative case; that is, for those students not labeled as dropouts. For dropout students, the F1 score tended to gravitate around the 0.5 range. We realized that our model likely suffered from class imbalance; since over 75% of our data members were non-dropouts, it was likely that most of the chosen neighbors would also be non-dropouts. To account for this bias, we chose to undersample the non-dropout data so that dropouts and non-dropouts would be equally represented. The results are shown below, using an optimal k-value of 9 (which, across many different runs of this data, proved itself to be one of the highest performing values. Note that there is a lot of variability in this measurement):

Results:



The chart on the left shows the original F1 scores, and the chart on the right shows the F1 scores that came after employing the undersampling method. It can be seen that our hypothesis was correct; our results were indeed biased because of the imbalance between dropouts and non-dropouts. Using the undersampling method allowed us to obtain much more consistent and meaningful data. We now obtain an average F1 score of about 0.65 for predicting both dropouts and non-dropouts. Prediction and recall values averaged around 0.65 as well.

Overall, our results seem to indicate that a student dropout is actually a fairly difficult variable to predict; our best model averages F1 scores of around 0.65, which is far from perfect, but is certainly more consistent than simply making a random guess. For this reason, our model may not be completely appropriate for use by school officials, but it can at least provide a fair amount of guidance when attempting to identify students in need of help.