IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication August 25, 2023

Digital Object Identifier TBD

Utilization of KNN Algorithm to Speculate the Dropout and Success Rate of Undergraduate Students

PRATIGYA PAUDEL[1], SUSHANK GHIMIRE[1]
[1]Institute of Engineering, Thapathali Campus, Bagmati 44600 Nepal (e-mail: pratigyapaudel0@gmail.com)
Corresponding author: Pratigya Paudel (e-mail: pratigyapaudel0@gmail.com).

"This work was completed as a part of a college practical for Data Mining (CT725)."

ABSTRACT K-Nearest Neighbors (KNN) classifiers are widely recognized for their simplicity and effectiveness in various machine learning tasks, including classification and regression. Their intuitive nature and ability to handle large datasets efficiently have made them popular in diverse domains. In this study, we explore the application of K-Nearest Neighbors in the context of predicting students' dropout and academic success, using a dataset that comprises various attributes related to students' demographics, past performance, and socio-economic factors. The goal is to investigate whether the provided attributes can reliably predict whether a student is likely to drop out or achieve academic success. This research seeks to leverage the KNN algorithm's ability to classify instances based on their proximity to existing data points. By measuring the distance between instances in the dataset and their k-nearest neighbors, we aim to develop a predictive model that effectively classifies students into categories of dropout or academic success.

INDEX TERMS Nearest Neighbour, Supervised Machine Learning

I. INTRODUCTION

KNN K-NEAREST NEIGHBOURS is one of the simplest supervised machine learning algorithms. KNN is used for classification and regression tasks to predict on an data instance, given the dataset. KNN uses feature similarity to get predictions on the newer data points with the help of already existing datapoints and their labels. The lazy learner algorithm does not learn anything from the dataset but rather stores the dataset and uses the newer values to perform actions on the dataset. The use of Euclidean distance determines the nearest datapoints to the given data instance which is used to match the features to a class label with respect to the k nearest neighbours. The use of higher number of neighbours helps reduce biasness from outliers while predicting a target class. The best possible results form a KNN algorithm is obtained by iteratively determining the best value of K for the highest accuracy.

The dataset "Predict students' dropout and academic success" comprises information from a higher education institution, amalgamated from multiple databases. It encompasses diverse undergraduate degrees including agronomy, design, education, nursing, journalism, management, social service, and technologies. The dataset incorporates details available during student enrollment, encompassing academic trajectory, demographics, and socio-economic factors. Additionally, students' academic performance at the end of the first and second semesters is included. The classification task is structured as a three-category problem. However, an inherent class imbalance is evident, with one class being predominant. To address this, various techniques will be explored to enhance model performance and mitigate bias. The primary aim is to construct classification models utilizing this dataset to predict three key outcomes: students' dropout, enrolled and graduate.

II. METHODOLOGY

A. THEORY

K-Nearest Neighbors (KNN) operates by It classifying data points by measuring their proximity to other data points in the dataset. Given a certain number of neighbors (K), a data point is classified based on the majority class among its nearest neighbors. In KNN, each data point is represented as a point in a multi-dimensional space, and the classification decision is made based on the classes of its closest neighbors. This method is particularly useful for tasks where the decision boundaries are not well-defined and might be irregular. KNN's interpretability lies in its simplicity – the classification of a data point is determined by the classes of

its closest neighbors. This approach can be easily understood and visualized, which is advantageous in fields that demand transparency and clarity, like medical diagnoses or financial risk assessments.

By selecting relevant features and determining the number of neighbors (K), KNN can effectively categorize data points into classes based on their proximity in the feature space. This adaptability to various datasets and its simplicity in interpreting results make KNN a valuable tool for scenarios where a clear understanding of the decision-making process is essential.

The classification report is a common tool used to evaluate the performance of a classification model. It provides a detailed summary of various metrics that assess how well the model has performed in classifying different classes or categories. The report is typically organized in a tabular format, with rows representing each class and columns representing different evaluation metrics. The most commonly included metrics in a classification report are:

- Precision: Precision is the ratio of true positive predictions to the total number of positive predictions. It measures the accuracy of the model in correctly identifying positive instances.
- Recall: Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the total number of actual positive instances. It measures the model's ability to correctly identify positive instances.
- F1-score: The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, taking into account both precision and recall.
- Support: Support refers to the number of actual occurrences of each class in the dataset. It represents the number of samples belonging to a particular class.
- Accuracy: Accuracy is the overall correctness of the model's predictions. It measures the proportion of correctly classified instances out of the total number of in
- Macro average: The macro average is the average of the performance metrics across all classes. It treats each class equally, regardless of their support.
- Weighted average: The weighted average is the average of the performance metrics, weighted by the support of each class. It provides a more accurate representation when classes have imbalanced support.

B. MATHEMATICAL FORMULAE

K-Nearest Neighbors (k-NN) classification hinges on distance metrics to quantify the similarity between data points. Although the widely used metric is the Euclidean distance, there exists flexibility to employ alternative measures. Presented below are formulas to compute distances between data points, enabling the determination of the closest neighbors:

1) Calculation of Euclidean Distance

Euclidean distance is a measure of the straight-line distance between two points in a multi-dimensional space. It is a commonly used distance metric in various fields to quantify the similarity or dissimilarity between data points.

The Euclidean distance between two points $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ in n-dimensional space can be calculated using the following formula:

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1}$$

2) KNN Selection Criteria

Let F be a distance function. $x_1, x_2, x_3, \ldots, x_n$ are training examples. The class $C(v)$ for a training example v is determined as:

$$C(v) = \arg \min_{x \in \{x_1, \ldots, x_n\}} (F(v, x)) \tag{2}$$

C. INSTRUMENTATION TOOLS

The entirety of the process is done using Python. Google Colab, short for Google Colaboratory, is an online platform provided by Google for running and sharing Jupyter notebook environments and it was used for all of the coding. Google colab provides a number of built-in functions for data analysis. The process of making the kNN classifier has been carried out using a number of available functions within the scikit-learn library. The dataset is visualized using pandas. Then, the process of building a KNN Classifier is done using the readily available functions within scikit-learn. The results are then visualized using different visualization tools like Seaborn and matplotlib.

D. WORKING PRINCIPLE

1) Dataset Collection

The dataset for classification of the students into the groups of enrolled, graduated and drop-out was downloaded from the UCI Machine learning datasets repository. The obtained dataset was prepared to be loaded into a pandas dataframe.

2) k-Nearest Neighbours Model Training

A K-Nearest Neighbors (KNN) classifier is instantiated using the scikit-learn library for dataset splitting. The classifier is trained on the training dataset, utilizing the specified number of neighbors (K) to determine classification based on proximity. Once the training is complete, the validation set is inputted into the trained classifier to predict values. The outcomes generated by the classifier are subsequently compared against the true labels of the validation dataset.

3) Data Analysis and Visualization

The dataset is visualized and the accuracy results for the various values of K is plot in different figures.

III. RESULTS

A. KNN CLASSIFIER WITH DEFAULT SETTINGS

A KNN classifier was trained to get the best possible results on the test dataset with respect to the training dataset. Using

a split of 7:3, the data instances on the test set was passed through the model to predict the target classes.

Before working on the model, the different attributes and their count were plotted in multiple histograms to visualize the distribution of the dataset with respect to all the existing features in it. The horizontal plot with the instances of the target classes showed a high disparity between the number of data points in the dataset that referred to the different classes. The number of enrolled students was only a small fraction of the total dataset.

The use of the default parameters during the instantiating of the KNN model yielded a sub-par 67% accuracy on the test dataset. The default settings for a KNN model as used by scikit-learn are 5 nearest neighbours, calculated on the basis of minkowski distance. The classification report showed poor results on the dataset as well.

As the smallest target class by volume, the confusion matrix for the different classes showed very poor results for classifying graduate students. The rest two classes are predicted much better with a large portion of those datasets being predicted correctly. However, its worth noting that there were still a high number of off-diagonal elements in the matrix.

B. KNN CLASSIFIER WITH PREDEFINED WEIGHTS

A KNN classifier was trained to get the best possible results on the test dataset with respect to the training dataset. Using a split of 7:3, the data instances on the test set was passed through the model to predict the target classes.

The use of the predefined during the instantiating of the KNN model yielded a sub-par 67% accuracy on the test dataset. The default settings for a KNN model as used by scikit-learn are 5 nearest neighbours, calculated on the basis of minkowski distance. The classification report showed poor results on the dataset as well.

As the smallest target class by volume, the confusion matrix for the different classes showed very poor results for classifying graduate students. The rest two classes are predicted much better with a large portion of those datasets being predicted correctly. However, its worth noting that there were still a high number of off-diagonal elements in the matrix. Thus, no notable change was obtained by supplying some weights to start off with initally on our own.

C. KNN CLASSIFIER WITH ITERATIVE CHANGES IN K

KNN models were created by changing the number of the K-nearest neighbours to search for, while selecting the target class. The iterative process yieled newer results from the dataset. The highest obtained accuracy from the curve was 73% while using 17 nearest neighbours for classification. The number of nearest neighbours plotted against the accuracy had a curve similar to a normal distribution curve.

IV. DISCUSSION AND ANALYSIS

The preceding sections demonstrated the effective utilization of a K-Nearest Neighbors (KNN) Classifier on the dataset for predicting students' dropout and academic success. By applying the KNN Classifier to the dataset, we achieved successful classification of instances into distinct categories. The KNN model operates by measuring the proximity of data points and assigning labels based on the majority class among their closest neighbors.

The KNN Classifier streamlined the dataset's complexity by leveraging proximity-based classification rules. This approach led to a clear differentiation between different student outcomes, where data points' classes were determined by the labels of their nearest neighbors.

A noteworthy insight from the KNN Classifier outcomes is the correlation between the model's performance and the choice of the number of neighbors (K). A higher K value often yields better accuracy in distinguishing between student outcomes. However, it's crucial to strike a balance, as excessively high K values can introduce noise or irrelevant patterns from the dataset. Thus, selecting an optimal K value is pivotal to achieving a well-performing model. A visual representation can be observed in the comparison of confusion matrices between KNN classifiers with different K values. It is clear that using more number of nearest neighbours can get better results until the accuracy hits a ceiling and then it starts dropping.

Furthermore, KNN Classifier enables analysis of feature importance. By examining the attributes of the nearest neighbors that contribute to classification, we can identify key factors influencing students' dropout and academic success predictions. This insight facilitates a deeper understanding of the underlying dynamics driving student outcomes.

The KNN Classifier emerges as a potent tool for predicting students' dropout and academic success, leveraging proximity-based classification. Through appropriate selection of the number of neighbors and interpretation of feature importance, we can attain accurate and interpretable results for forecasting student outcomes using the KNN model.

V. CONCLUSION

This lab adhered to the principles of applying a K-Nearest Neighbors (KNN) Classifier. The deployment of the KNN Classifier on the dataset for predicting students' dropout and academic success underscores its effectiveness within this specific context. The primary aim of utilizing the KNN Classifier, which is classification, was successfully accomplished by accurately translating decision rules derived from the students' dataset. By considering diverse attributes and adopting suitable decision criteria, the KNN Classifier adeptly captured the inherent patterns and associations within the dataset, resulting in precise classification outcomes.

The analysis uncovers that the performance of the KNN Classifier is influenced by several factors, including the number of neighbors (K) and the pertinent selection of features tailored to the students' dataset. Higher K values often contribute to enhanced accuracy in predicting student outcomes. Nevertheless, an equilibrium must be struck to prevent overfitting and ensure the inclusion of only relevant attributes. The process of feature selection, driven by the choice of

neighbors, significantly contributes to the classifier's ability to accurately anticipate students' dropout and academic success. By accounting for informative attributes distinctive to the students' context, the KNN Classifier proficiently differentiates between students at risk of dropout and those likely to succeed. The outcomes attained through the KNN Classifier accentuate its proficiency in addressing student outcome prediction tasks and delivering intelligible models. Extracting insights from decision criteria and associated features offers valuable comprehension of the factors impacting student classifications. Furthermore, the KNN Classifier allows for the assessment of feature importance, shedding light on the pivotal attributes that influence the classification process, specifically tailored to student outcomes.

The adoption of the KNN Classifier underscores its effectiveness in predicting students' dropout and academic success. By considering the right balance of neighbors, feature selection, and interpreting decision criteria, precise and interpretable classification results can be realized. The KNN Classifier provides a constructive approach to grasp the underlying structures and patterns within the dataset related to student outcomes, thereby establishing itself as a potent tool for such prediction tasks.

While the KNN Classifier stands as a powerful tool for student outcome prediction, it is not without limitations. Notably, it can struggle with higher dimensionality, where the curse of dimensionality can impact its performance. Careful consideration should be given to feature selection and preprocessing to mitigate these challenges.

VI. REFERENCES

- David Bowser-Chao and Debra L. Dzialo. "Comparison of the use of nearest neighbours and neural networks in top-quark detection." *Physical Review D*, vol. 47, no. 5, pp. 1900–1905, Mar. 1993. doi: 10.1103/physrevd.47.1900.

PRATIGYA PAUDEL is a fourth year student, studying computer engineering under IOE, Thapathali Campus. She has been involved in a lot of machine learning projects and has a keen eye for data analysis and AI related stuff. With the enthusiasm for Artificial Intelligence (AI), she is driven by the potential of AI to transform industries and tackle complex challenges. Her academic journey has equipped her with a strong foundation in AI concepts, including machine learning and data analysis. She possesses a relentless curiosity and is always eager to explore the latest advancements in AI. Her goal is to apply her knowledge and make a meaningful contribution in the field.

SUSHANK GHIMIRE is a fourth year student, studying computer engineering under IOE, Thapathali Campus. He possesses a lot of interest, working with data. His educational path has provided him with a solid understanding of AI concepts, encompassing machine learning and data analysis. He possesses an unwavering curiosity and is constantly eager to delve into the latest advancements in AI. His objective is to leverage his knowledge and expertise to create a significant impact in the field.

APPENDIX

A. TABLES

Attribute	Description
Marital status	The marital status of the student
Application mode	The method of application used by the student
Application order	The order in which the student applied
Course	The course taken by the student
Daytime/evening attendance	Whether the student attends classes during evening
Previous qualification	Qualification obtained by student before enrolling
Previous qualification (grade)	Grade of previous qualification
Nationality	The nationality of the student
Mother's qualification	The qualification of student's mother
Father's qualification	The qualification of student's father
Mother's occupation	The occupation of student's mother
Father's occupation	The occupation of student's father
Admission grade	Grade at admission
Displaced	Whether the student is a displaced person
Educational special needs	Whether the student has special educational needs
Debtor	Whether the student is a debtor
Tuition fees up to date	Whether tuition fees are up to date
Gender	The gender of the student
Scholarship holder	Whether the student is a scholarship holder
Age at enrollment	Age of student at enrollment
International	Whether the student is international
Curricular units 1st sem (credited)	Number of credited units in 1st semester
Curricular units 1st sem (enrolled)	Number of enrolled units in 1st semester
Curricular units 1st sem (evaluations)	Number of evaluations in 1st semester
Curricular units 1st sem (approved)	Number of approved units in 1st semester
Curricular units 1st sem (grade)	Grade average in 1st semester
Curricular units 1st sem (without eval.)	Number of units without evaluations in 1st semester
Curricular units 2nd sem (credited)	Number of credited units in 2nd semester
Curricular units 2nd sem (enrolled)	Number of enrolled units in 2nd semester
Curricular units 2nd sem (evaluations)	Number of evaluations in 2nd semester
Curricular units 2nd sem (approved)	Number of approved units in 2nd semester
Curricular units 2nd sem (grade)	Grade average in 2nd semester
Curricular units 2nd sem (without eval.)	Number of units without evaluations in 2nd semester
Unemployment rate	Unemployment rate
Inflation rate	Inflation rate
GDP	Gross Domestic Product
Target	Dropout, Enrolled, or Graduate at the end of course

TABLE 1. Description of Attributes

B. FIGURES AND PLOTS

FIGURE 1. System Block Diagram

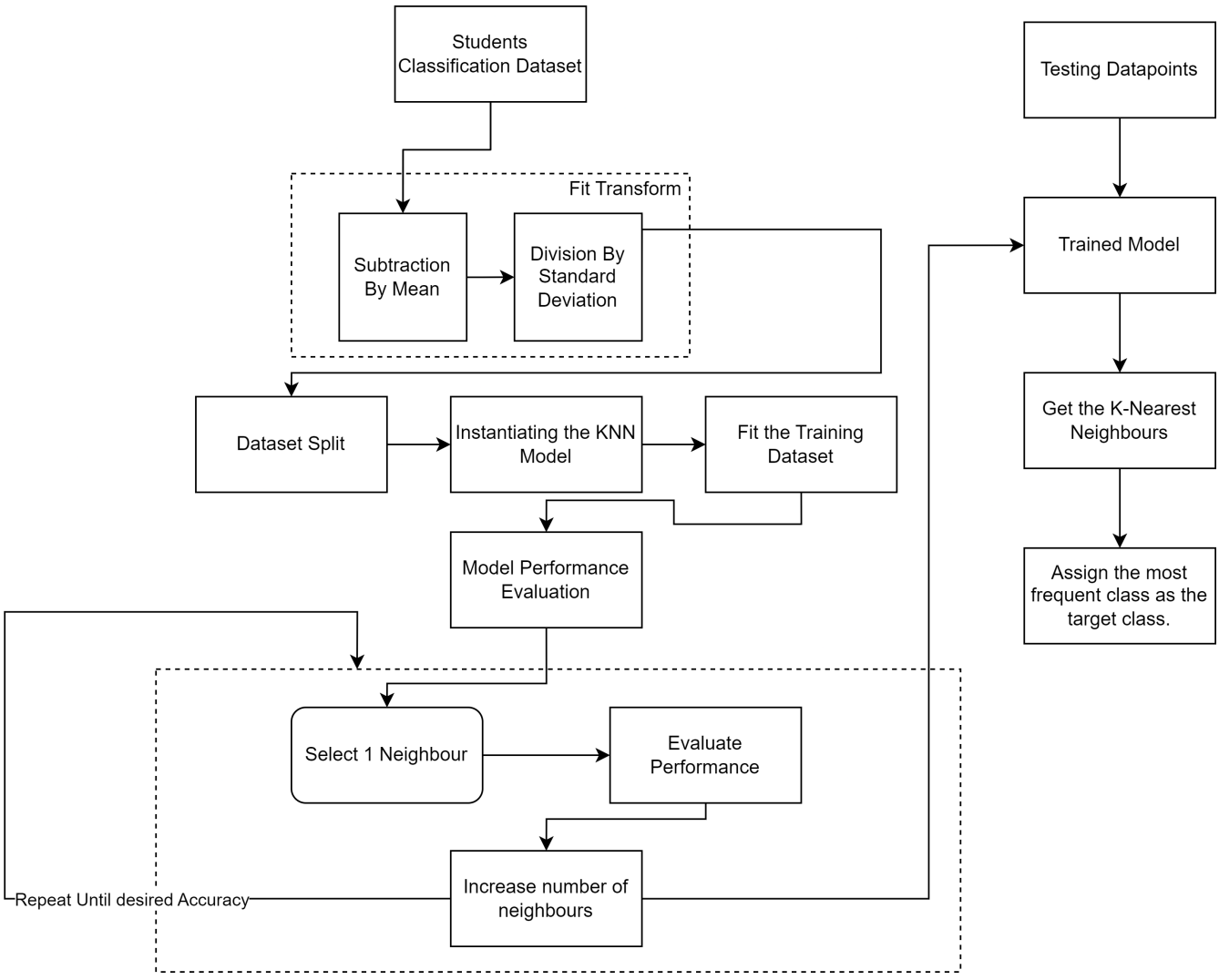
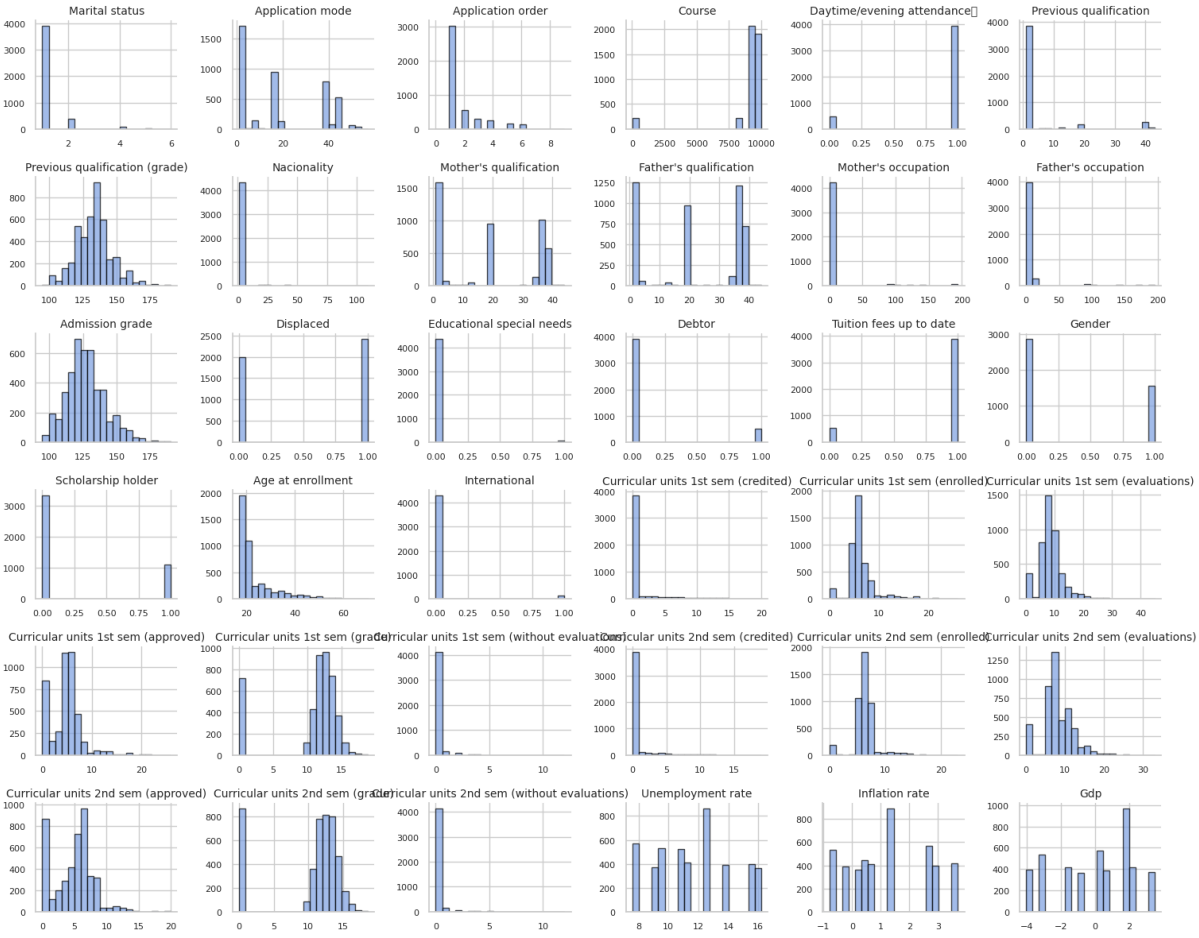


FIGURE 2. Histogram of Attribute values



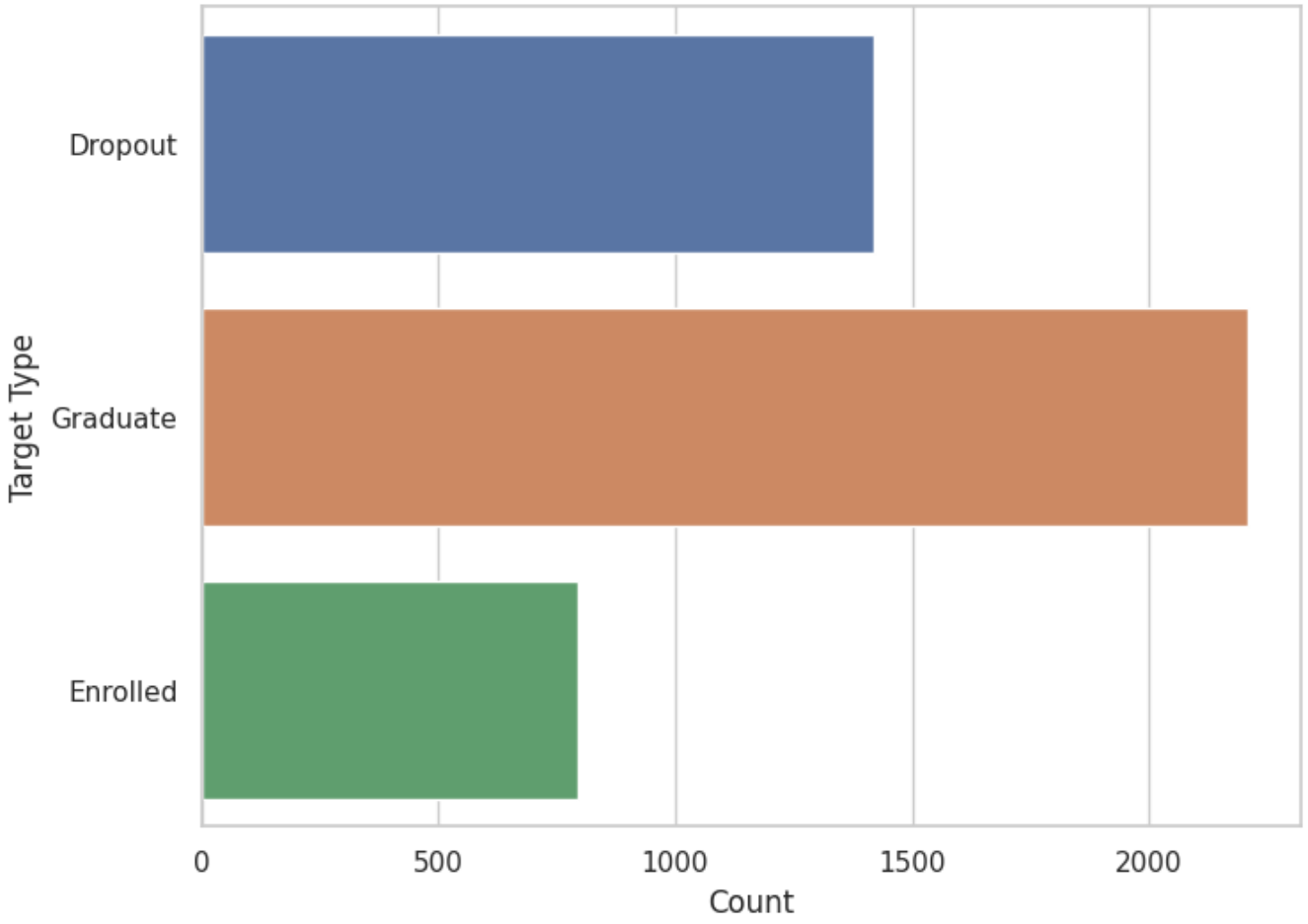


FIGURE 3. Horizontal plot for volume of target class datapoints

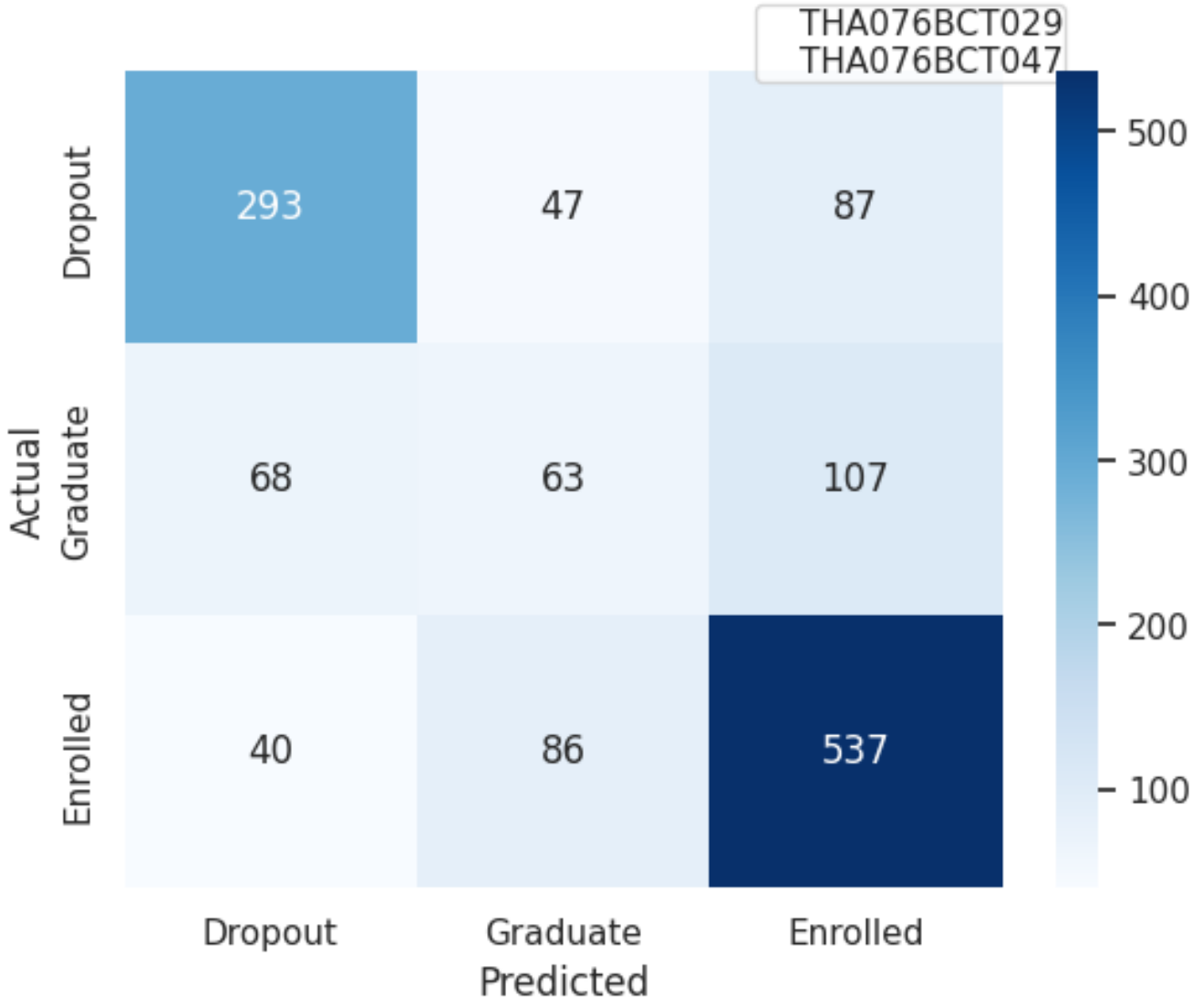


FIGURE 4. Confusion matrix for default settings KNN model

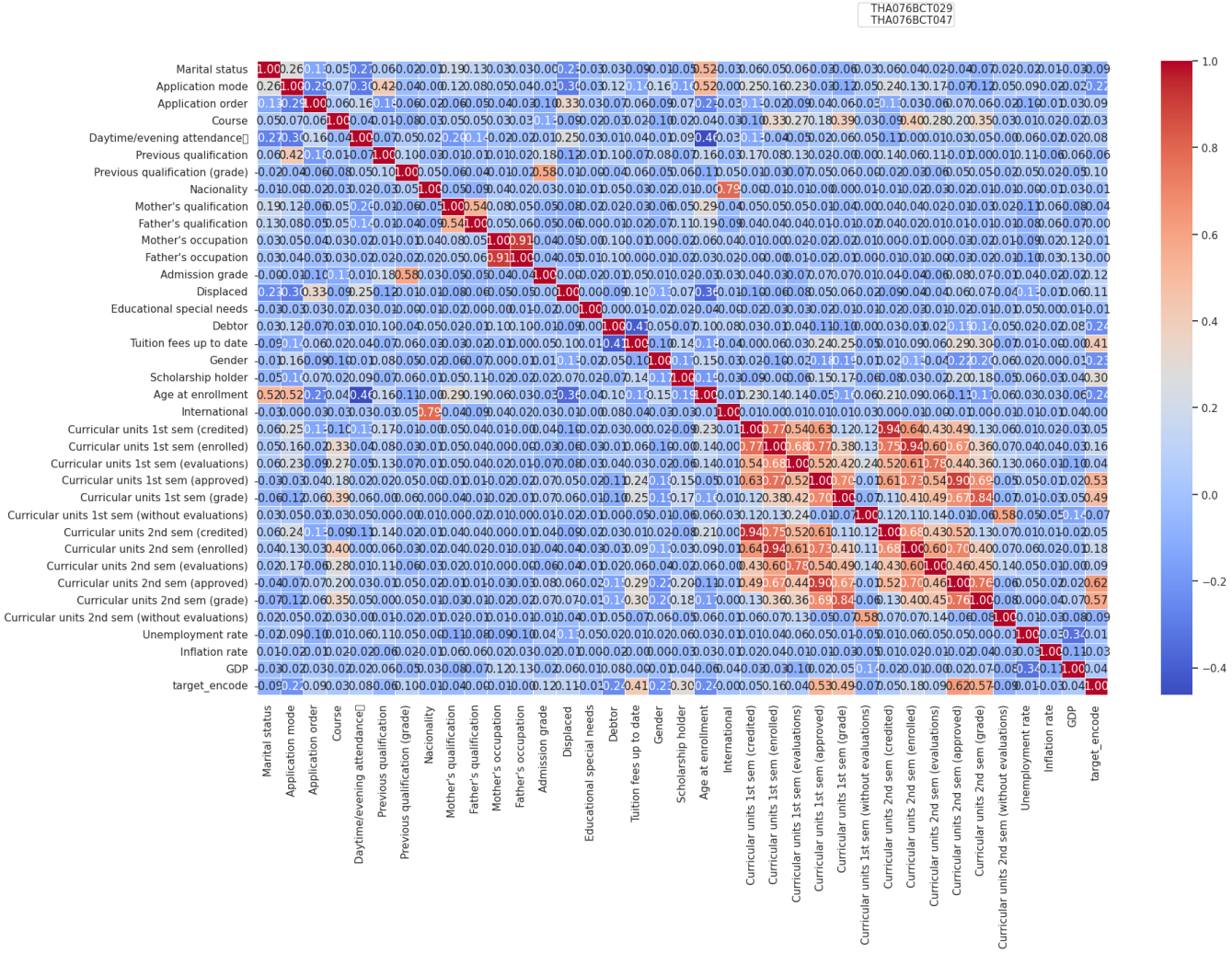


FIGURE 5. Correlation matrix of Attribute values

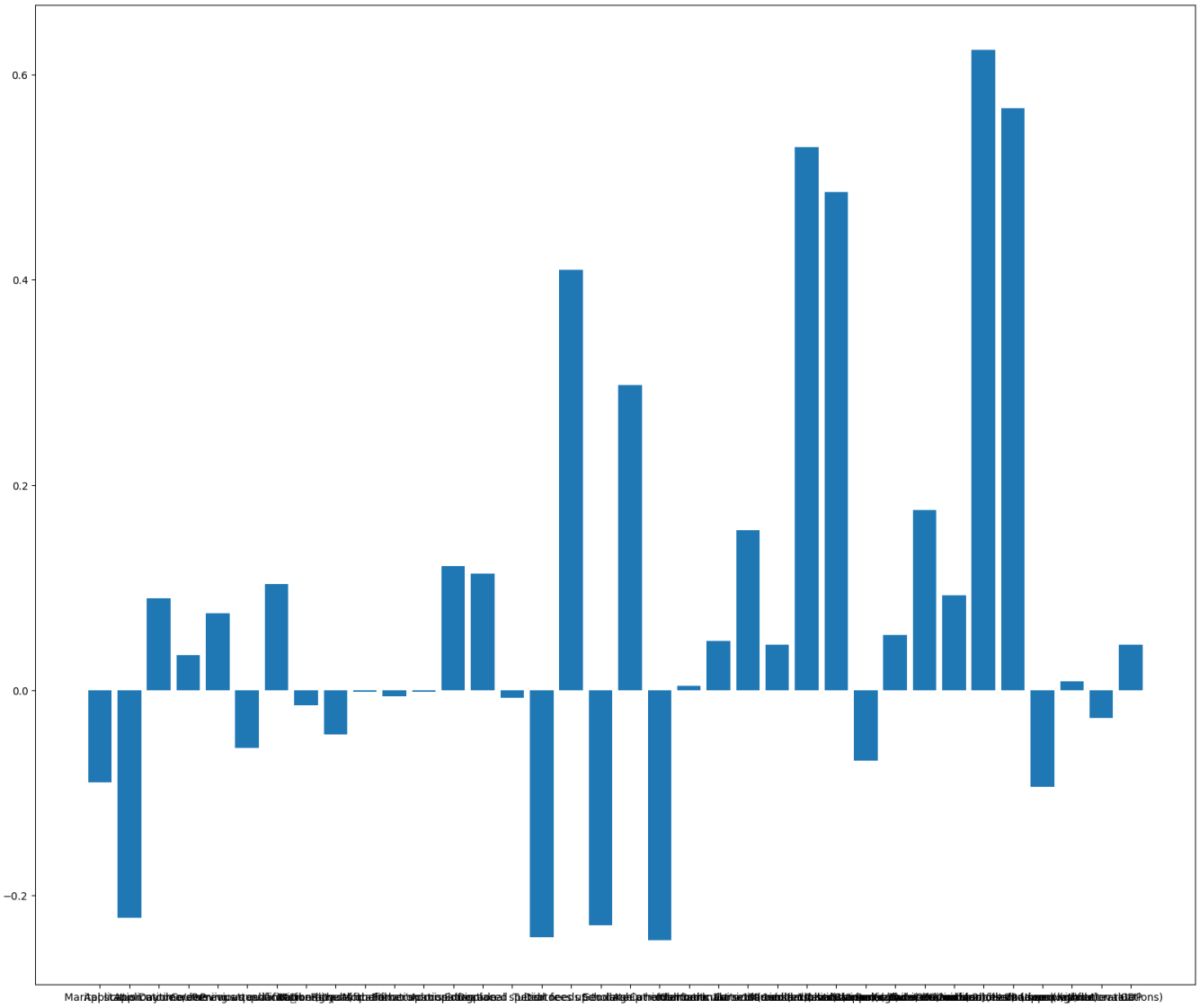


FIGURE 6. Histogram of attribute correlation values

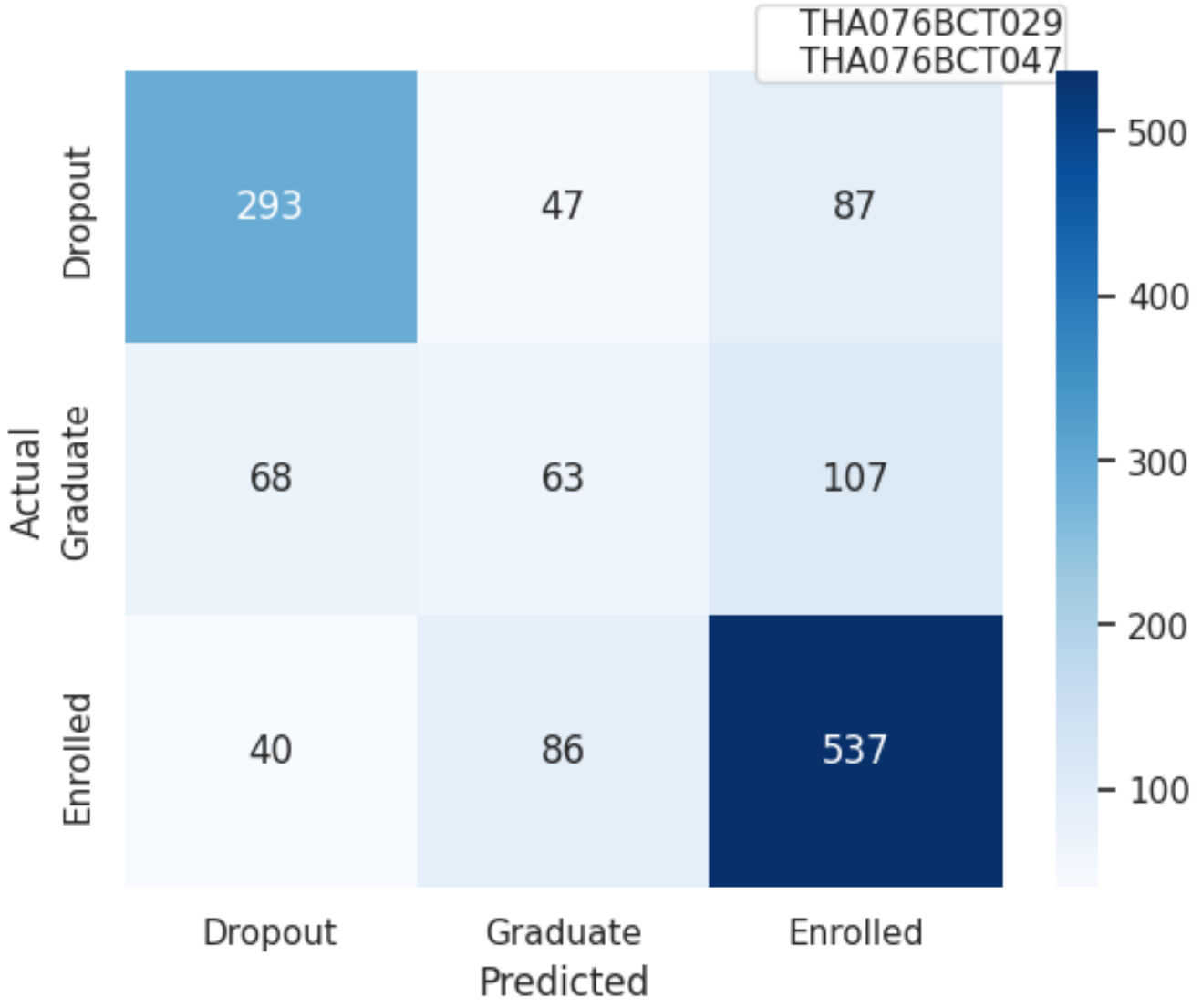


FIGURE 7. Confusion matrix for manual weights model

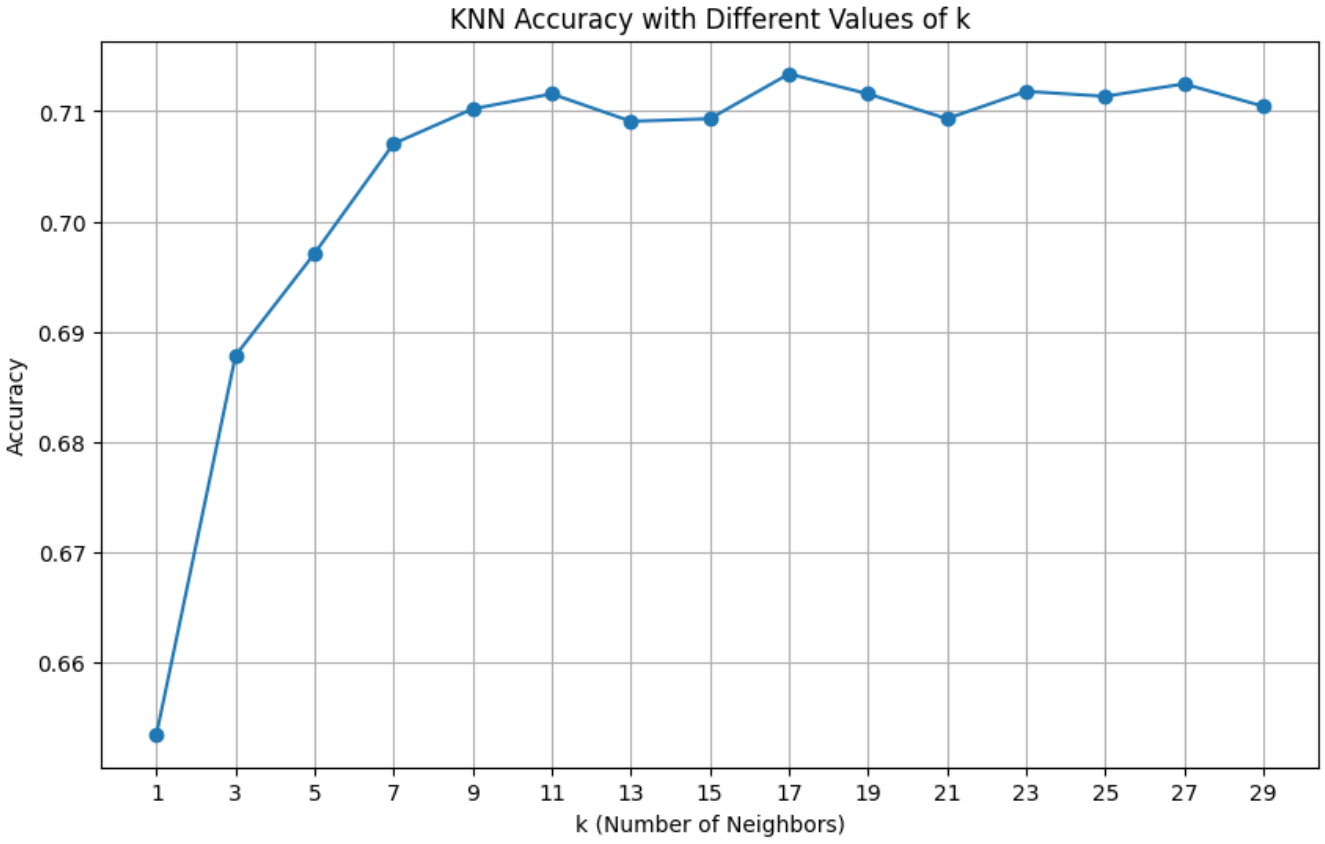


FIGURE 8. Line plot for Accuracy against Number of neighbours

C. CODING

```
#Necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder
from scipy.stats import zscore
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

#Dataset Visualization
#read dataset
data = pd.read_csv('/content/data.csv', sep=';')

fig, axes = plt.subplots(nrows=6, ncols=6, figsize=(15, 12))
plt.subplots_adjust(hspace=0.5)

# Plot histograms for each attribute
for ax, column in zip(axes.flatten(), data.columns):
    ax.hist(data[column], bins=20, color='#7b9fe0', alpha=0.7, edgecolor='black')
    ax.set_title(column.capitalize(), fontsize=10)
    ax.tick_params(axis='both', which='both', labelsize=8)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

# Remove empty subplots
if data.shape[1] < 30:
    for ax in axes.flatten()[data.shape[1]:]:
        ax.remove()

plt.tight_layout()
plt.show()

#plot graph of count on the target
sns.set(style="whitegrid")   # Optional: Set a seaborn style
plt.figure(figsize=(8, 6))   # Optional: Set the figure size

# Replace underscores with spaces in the 'Target' column
data1 = data.copy()
data1['Target'] = data1['Target']

sns.countplot(y='Target', data=data1, orient='h')

plt.xlabel('Count')   # Optional: Set the x-axis label
plt.ylabel('Target Type')   # Optional: Set the y-axis label

plt.show()

#correlation between the attributes
```

```
classes = data['Target'].unique()
cor=data.corr(method='pearson')
plt.figure(figsize=(20, 12))

# Choose a color map for the heatmap (optional)
# You can find more colormaps at: https://matplotlib.org/stable/tutorials/colors/colormaps.html
cmap = 'coolwarm'

# Draw the heatmap
sns.heatmap(cor,annot = True, fmt=".2f",cmap=cmap,linewidths=0.5)

# Add a title to the heatmap
legend_handles = [
    plt.Line2D([], [], color='black', marker='o', markersize=10, label='THA076BCT029\nTHA0
]
plt.legend(handles=legend_handles, loc='upper left', bbox_to_anchor=(0.7, 1.1), ncol=len(le

# Rotate the x-axis labels for better readability (optional, adjust as needed)
plt.xticks(rotation=90)

# Show the plot
plt.show()

#Dataset processing
targets = data.groupby(['Target']).size()
print(targets)
label_encoder = LabelEncoder()
data['target_encode'] = label_encoder.fit_transform(data['Target'])
label_counts = data['Target'].value_counts()
# Print the count of each label
print(label_counts)
new_data= data.drop(labels = ["Target",'target_encode'],axis = 1)
new_data.head()
label=data['target_encode']
new_data[:20].describe()

# Perform z-score normalization on each column
df_normalized = new_data.apply(zscore)

#Train Test Split
Y=label
X_train, X_test, y_train, y_test = train_test_split(df_normalized, Y, test_size = 0.3, rand

#Kneighbors Classifier
# Create a default KNN model
knn = KNeighborsClassifier()

# Get the parameters of the KNN model
params = knn.get_params()

print(params)
knn.fit(X_train, y_train)
y_pred1=knn.predict(X_test)
report = classification_report(y_test, y_pred1, zero_division=1, digits=4)
print(report)
accuracy = accuracy_score(y_test, y_pred1)
```

```
print(f"Accuracy: {accuracy:.4f}")


conf = confusion_matrix(y_test, y_pred1)
# Set the colormap for the heatmap
cmap = 'Blues'
# Create a heatmap of the confusion matrix
sns.heatmap(conf, annot=True,cmap=cmap, xticklabels=classes, yticklabels=classes,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
legend_handles = [
    plt.Line2D([], [], color='black', marker='o', markersize=10, label='THA076BCT029\nTHA0
]
plt.legend(handles=legend_handles, loc='upper left', bbox_to_anchor=(0.7, 1.1), ncol=len(le
plt.show

last_row = cor['target_encode']

# Print the last row
val=last_row.values[:-1]
print(val)
feat=cor.columns[:-1]
print(feat)

fig = plt.figure(figsize =(20, 17))

# Horizontal Bar Plot
plt.bar(feat,val )

# Show Plot
plt.show()

#For Weaighted Kneighbour Classifier
np.random.seed(10)
weights = np.random.rand(11)
feat=cor.columns[:-1]
print(feat)
knn_model = KNeighborsClassifier(metric_params = {'weights':weights})

# Print the instantiated model
print(knn_model)
paramss = knn_model.get_params()

print(paramss)
knn_model.fit(X_train, y_train)
y_pred2=knn_model.predict(X_test)
report = classification_report(y_test, y_pred2, zero_division=1,digits=4)
accuracy = accuracy_score(y_test, y_pred2)
print(f"Accuracy: {accuracy:.4f}")

from sklearn.metrics import confusion_matrix
conf = confusion_matrix(y_test, y_pred2)
# Set the colormap for the heatmap
cmap = 'Blues'
# Create a heatmap of the confusion matrix
sns.heatmap(conf, annot=True,cmap=cmap, xticklabels=classes, yticklabels=classes,fmt='d')
```

```
plt.xlabel('Predicted')
plt.ylabel('Actual')
legend_handles = [
    plt.Line2D([], [], color='black', marker='o', markersize=10, label='THA076BCT029\nTHA0
]
plt.legend(handles=legend_handles, loc='upper left', bbox_to_anchor=(0.7, 1.1), ncol=len(le
plt.show

#Cross Validation Score Calculation and Visualization

import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

# Step 1: Load the dataset and split into features (X) and labels (y)
# x_data = data.drop(['Target'],axis = 1)
X, y = df_normalized, label

# Step 2: Choose a range of k values to try with step size 2
k_values = list(range(1, 31, 2))  # For example, trying odd k values from 1 to 30

# Step 3: Perform k-Fold Cross-Validation for each k value and collect accuracy scores
accuracy_scores = []
for k in k_values:
    # Instantiate KNN model
    knn_model = KNeighborsClassifier(n_neighbors=k)

    # Perform k-fold cross-validation and get accuracy scores
    cv_scores = cross_val_score(knn_model, X, y, cv=10)  # Using 10-fold cross-validation

    # Get the mean accuracy from cross-validation
    accuracy = np.mean(cv_scores)
    print(f"Average Accuracy for k={k}: {accuracy:.4f}")
    accuracy_scores.append(accuracy)

# Step 4: Find the k value with the highest accuracy
best_k = k_values[np.argmax(accuracy_scores)]
best_accuracy = max(accuracy_scores)

# Step 5: Plot the relationship between k values and accuracy scores
plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracy_scores, marker='o')
plt.xlabel('k (Number of Neighbors)')
plt.ylabel('Accuracy')
plt.title('KNN Accuracy with Different Values of k')
plt.xticks(k_values)
plt.grid(True)
plt.show()

print(f"Optimum value of k: {best_k}")
print(f"Accuracy with optimum k: {best_accuracy:.5f}")
```