Title: User Knowledge Modelling Data Set

Student ID : S4015465

Student Name : Vivek Aggarwal

Email : s4015465@student.rmit.edu.au

Affiliations: RMIT University.

Date of Report: 23rd May 2023

I certify that this is all my own original work. If I took any parts from elsewhere, then they are non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honour code by typing "Yes": *Yes*.

User Knowledge Modelling Data Set

Practical Data Science (COSC2670)

23rd May 2023

Vivek Aggarwal

s4015465@student.rmit.edu.au

Affiliations

RMIT University

Lecturer:

Dr. Yongli Ren

Table of Content

Abstract

The classification models, particularly K-Nearest Neighbours (KNN) and decision trees, to a dataset comprising different attribute information. The dataset includes features such as study time for goal object materials (STG), repetition number for goal object materials (SCG), study time for related objects with the goal object (STR), exam performance for related objects (LPR), exam performance for goal objects (PEG), and knowledge level of the user.

The KNN method is used to find patterns and categorise instances according to how similar they are to nearby data points. On the other hand, the decision tree model uses a hierarchical structure to reach decisions based on attribute thresholds. Both models are tested using suitable performance indicators after being trained on a portion of the dataset.

Introduction

Classification models are essential for organising and classifying data based on particular properties in the fields of data analysis and machine learning. They facilitate pattern recognition, result forecasting, and decision-making processes. In this study, classification models, particularly K-Nearest Neighbours (KNN) and decision trees, are applied to a dataset that contains attribute data about study habits and exam performance. The dataset includes characteristics like the amount of study time for goal object materials (STG), the quantity of study time for goal object materials (SCG), the amount of study time for related objects to the goal object (STR), the performance on exams for related objects (LPR), the performance on exams for goal objects (PEG), and the user's knowledge level (UNS).

This project's main goal is to employ KNN and decision tree models to categorise cases based on the given qualities and obtain knowledge about the correlation between study habits, exam performance, and user knowledge level. These models can be used to categorise situations and pinpoint important variables affecting student progress.

Methodology

Data

The dataset that includes attributes like STG, SCG, STR, LPR, PEG, and UNS that are related to study habits and exam results. The UNS column the target value is The dataset should be pre-processed by handling missing values, normalising any necessary numerical characteristics, and encoding categorical variables.

Data Exploratory Analyses

In order to understand the characteristics of the dataset. We performed exploratory data analysis to gain insights into the distribution of variables, identify outliers, and discover any relationships or patterns between the attributes using python libraries like matplotlib.

To analyse each column separately I have used Pie chart for the UNS column as it is categorical and then to analyse each attribute like STG, SCG, STR, LPR, PEG using bar graph. To analyse the relation between the columns we have used scatter plot which will tell the relation of UNS with respect to each attribute as well as the density of each attribute with respect to each column.

Training and Testing

We will train our model on the training set and test it on the testing set to observe how it performs on new data, allowing us to evaluate how effectively the model generalises to new observations. Most of the data is often found in the training set, which is used by the model to discover trends and connections between the input features (STG, SCG, STR, LPR, PEG) and the target variable (UNS). On the other hand, the testing set is used to evaluate the model's effectiveness by contrasting its predictions with the actual values of UNS.

We can assess the model's accuracy, precision, recall, or other performance metrics by dividing the data into training and testing sets to ascertain how well it can predict the knowledge level of users based on the provided attributes.

Data Modelling

K-Nearest Neighbours Model

A measure of similarity (typically distance) is used to identify the k nearest neighbours of a given instance, and the class label of the majority of the k nearest neighbours is then assigned as the predicted class label for the instance. The k-NN algorithm operates under the presumption that similar instances are likely to have similar class labels. The k-NN algorithm requires choosing the k neighbours to be considered for classification. The dataset will determine the best value of k, which should be carefully selected. A low number of k could result in overfitting, whereas a high value could lead to underfitting. It is usual practise to test out various k values and pick the one that offers the best performance.

The knowledge level (UNS) of the instances in the testing set can be predicted using the model once it has been trained. The training set's k nearest neighbours are calculated for each instance by the k-NN method, which then determines the class label based on a majority vote among the neighbours.

Decision Tree Model

In the decision tree model the feature that offers the greatest information gain or impurity reduction is chosen as the tree's root. This feature would be the most effective in classifying the cases according to their levels of expertise. For example, When the algorithm finds that the PEG feature offers the most information gain, it divides the dataset according to the various PEG values.

The algorithm then divides the data according to the selected attribute, generating child nodes for each attribute's potential values. For instance, if the selected property is "STG," the data will be split into subsets based on various "PEG" values (such as very_low, low, middle, and high). Until a stopping requirement is satisfied, such as reaching a maximum tree depth or a minimum number of occurrences in a node, the attribute selection and data splitting procedure is repeated again.

The Decision Tree method travels through the tree from the root to a leaf node, following the decision pathways determined by the attribute values of the instance, in order to predict the knowledge level for a new instance. The expected knowledge level is established once it reaches a leaf node based on the majority class of the instances in that leaf node.

RESULT

Task 1: Retrieving and Preparing the data

The data is extracted from the UCI Repository and stored in a Pandas data frame. The link tells about better understanding of the characteristics. I began by gathering information on the columns and confirming the data types for each one. Additionally, I checked the dataset for errors, null values, and white spaces. I discovered mistakes and white spacing in the dataset after completing these activities**.**

ERROR 1: White Spaces

We can observe the white space in the 'UNS' column. So, we deleted the white space using astype(str).str.strip()

```
In [7]: #REMOVING THE WHITE SPACES FROM EACH COLOUMN

        dataset_df['STG'] = dataset_df['STG'].astype(str).str.strip()
        dataset_df['SCG'] = dataset_df['SCG'].astype(str).str.strip()
        dataset_df['LPR'] = dataset_df['LPR'].astype(str).str.strip()
        dataset_df['STR'] = dataset_df['STR'].astype(str).str.strip()
        dataset_df['PEG'] = dataset_df['PEG'].astype(str).str.strip()
        dataset_df['UNS'] = dataset_df[' UNS'].astype(str).str.strip()
```

ERROR 2: Wrong data type

I checked the data type of each column and noticed some of them were of object type.

```
In [8]: #CHECKING FOR DATA TYPE OF EACH COLOUMN
        dataset_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 403 entries, 0 to 402
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   STG                    403 non-null    object
 1   SCG                    403 non-null    object
 2   STR                    403 non-null    object
 3   LPR                    403 non-null    object
 4   PEG                    403 non-null    object
 5    UNS                   403 non-null    object
 6   Unnamed: 6             0 non-null      float64
 7   Unnamed: 7             0 non-null      float64
 8   Attribute Information:  12 non-null    object
 9   UNS                    403 non-null    object
dtypes: float64(2), object(8)
memory usage: 31.6+ KB
```

So I changed the data type of all those columns except the 'UNS' column from object to categorical by using the astype(float) function.

```
In [9]: #CONVERTING ALL OBJECT DATA TYPE INTO FLOAT VALUES USING ASTYPE()
        dataset_df['STG'] = dataset_df['STG'].astype(float)
        dataset_df['SCG'] = dataset_df['SCG'].astype(float)
        dataset_df['STR'] = dataset_df['STR'].astype(float)
        dataset_df['LPR'] = dataset_df['LPR'].astype(float)
        dataset_df['PEG'] = dataset_df['PEG'].astype(float)
```

```
In [10]: #CHECKING CHANGED DATA TYPE
         dataset_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 403 entries, 0 to 402
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   STG                    403 non-null    float64
 1   SCG                    403 non-null    float64
 2   STR                    403 non-null    float64
 3   LPR                    403 non-null    float64
 4   PEG                    403 non-null    float64
 5    UNS                   403 non-null    object
 6   Unnamed: 6             0 non-null      float64
 7   Unnamed: 7             0 non-null      float64
 8   Attribute Information:  12 non-null    object
 9   UNS                    403 non-null    object
dtypes: float64(7), object(3)
memory usage: 31.6+ KB
```

Error 3: Unecessary columns

There were some unnecessary columns in the data that were not really needed. I have dropped those columns using drop() function.

```
In [11]: #DROPPING THE COLUMNS BECAUSE WE DONT NEED THEM
         dataset_df = dataset_df.drop('Attribute Information:',axis=1)
         dataset_df = dataset_df.drop(columns=['Unnamed: 6','Unnamed: 7'])
         dataset_df = dataset_df.drop(columns=[' UNS',])
```

```
In [12]: #CHECKING WHETHER COLUMNS ARE ACTUALLY DELETED OR NOT
         dataset_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 403 entries, 0 to 402
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   STG     403 non-null    float64
 1   SCG     403 non-null    float64
 2   STR     403 non-null    float64
 3   LPR     403 non-null    float64
 4   PEG     403 non-null    float64
 5   UNS     403 non-null    object 
dtypes: float64(5), object(1)
memory usage: 19.0+ KB
```

Error 4: Obvious typing mistake

When we display the unique values in the ‘UNS’ column, we see that there are 2 distinct values as ‘very_low’ and ‘very low’. Clearly they are supposed to be the same. So I changed the ‘very_low’ to ‘very low’ using replace() function.

```
In [13]: #CHECKING FOR UNIQUE VALLUES IN THE UNS COLUMN
         print(dataset_df['UNS'].unique())
```

```
['very_low' 'High' 'Low' 'Middle' 'Very Low']
```

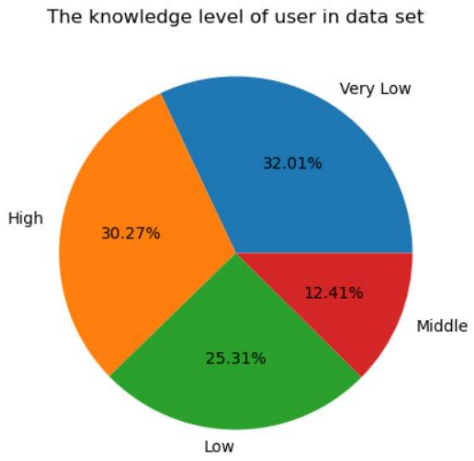
```
In [14]: #REPLACING VERY_LOW WITH VERY LOW
         dataset_df['UNS'] = dataset_df['UNS'].replace({'very_low': 'Very Low'})
```

```
In [15]: #CHECKING FOR THE UNIQUE VALUES AGAIN
         print(dataset_df['UNS'].unique())
```

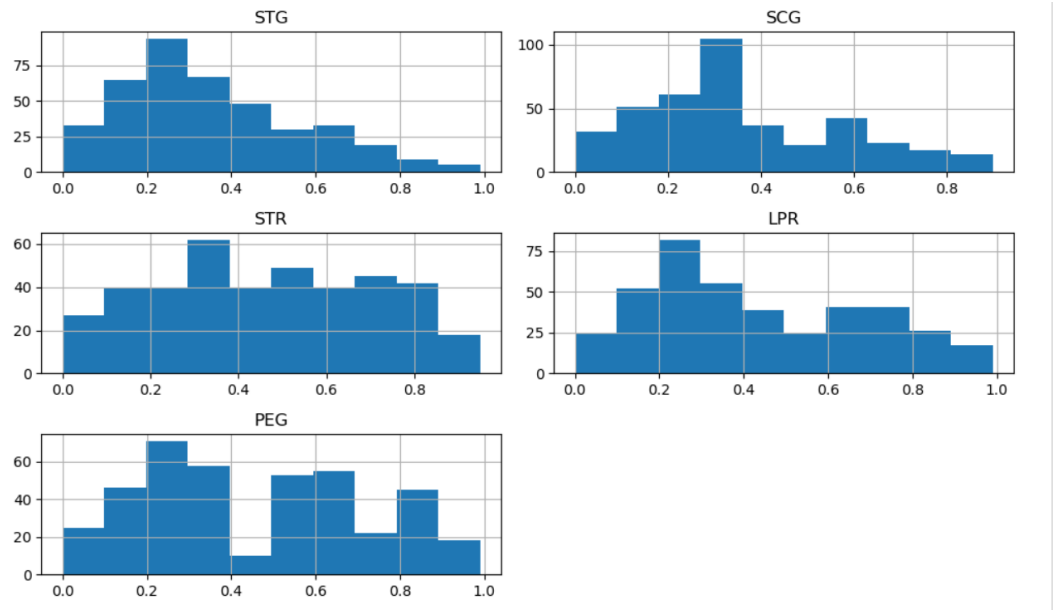
```
['Very Low' 'High' 'Low' 'Middle']
```

Task 2: Data Exploration

Data exploration, which involves looking at the dataset's structure, identifying patterns, correlations, and outliers as well as extracting significant data that can guide additional analysis and modelling, is a crucial stage in understanding and drawing conclusions from the dataset. We must first evaluate each column of the train and test dataset by generating a graph for it in order to forecast knowledge levels based on study habits and performance characteristics.



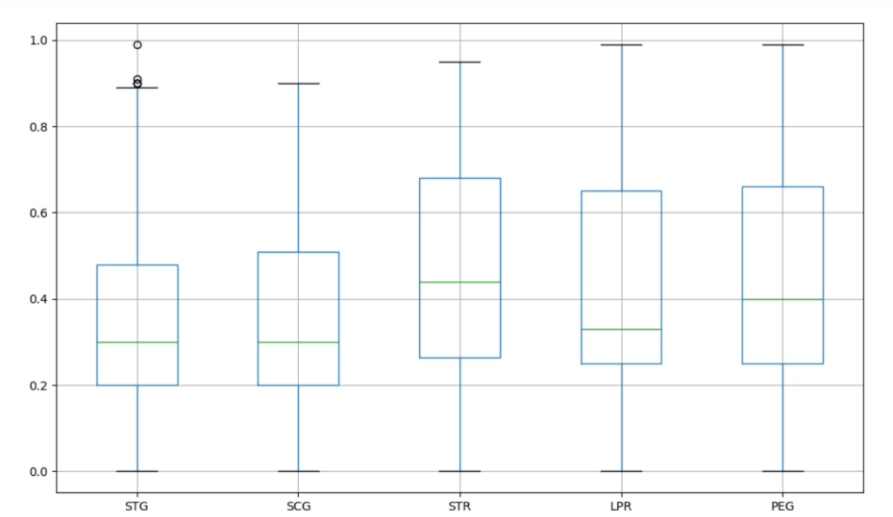
The given pie-chart gives the knowledge level of the users in the dataset. We can observe that the UNS column has maximum 32.01% of 'very low' attribute, whereas 'middle' is with the least percent of 12.41%.



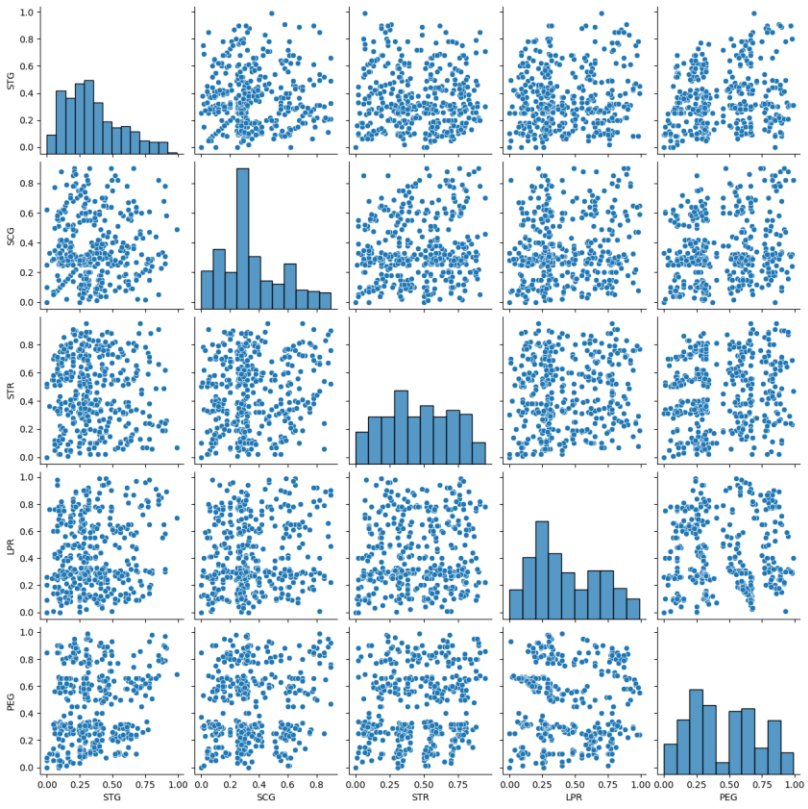
The graph for each individual column for the dataset is represented in the above figure.

The y axis in the above graphs represent the frequency of occurance of the attributes in the x axis.

- The frequency of STG column has the most occurance of 0.2 value.
- The frequency of SCG column has the most occurance of 0.3 value.
- The frequency of STR column has the most occurance of 0.3 value.
- The frequency of LPR column has the most occurance of 0.2 value.
- The frequency of PEG column has the most occurance of 0.2 value.



For each column of the dataset, STG, STR, SCG, LPR, PEG, a boxplot is made above in which we can observe that STG column has outliers. The green line in each boxplot represents the mean value of that column.



Above is pairplot of all the the columns of the dataset with respect to each other. The above plots shows no siginificance presence of outliers.

Task 3: Data Modelling

Before beginning the data modelling process, we must encode the string data into numeric form because the "Knowledge Level" column is a string object that the computer cannot comprehend.

Model 1: k-Nearest Neighbour (kNN)

Step 1:Before modelling, the dataset is trained and tested using the train_test_split package. It is necessary for this to divide the training data set into input features (attributes) and the target variable (knowledge levels). The testing set is kept apart and is used to assess the model's performance while the training set is used to train the model.

Step 2: Scaling the features in a dataset is required after training and testing the model since it normalises the dataset and ensures that all of the characteristics are on the same scale. This is necessary when working with machine learning algorithms that are sensitive to the size of the input characteristics.

Step 3: To determine accuracy with various parameters, the Knn model is run on the dataset. We are able to select the model with the highest accuracy after tuning the model at least five different ways with various parameters. The accuracy of the K-nearest neighbours (KNN) model illustrates how well it can forecast users' knowledge levels based on the provided parameters related to study habits and performance.

After the KNN model was trained on the training dataset and its performance was evaluated on the testing dataset, the accuracy score will display the percentage of instances in the testing dataset that were correctly detected. Based on the values of, it assesses how well the model can classify users' knowledge levels as Very Low, Low, Middle, High, or Very High.

```
In [34]: #REPEATING SAME PROCESS OF SPLITTING, INTANTIATING, PREDICITNG ETC

knn = KNeighborsClassifier(n_neighbors=5, weights='distance', p=1)

knn.fit(X_train, y_train)

y_prediction = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_prediction)
print('Accuracy:', accuracy)

print(classification_report(y_test, y_prediction))
```

```
Accuracy: 0.8888888888888888
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        21
           1       0.78      0.95      0.86        22
           2       0.92      0.88      0.90        26
           3       1.00      0.67      0.80        12

    accuracy                           0.89        81
   macro avg       0.91      0.86      0.88        81
weighted avg       0.90      0.89      0.89        81
```

Precision is the proportion of accurately produced favourably anticipated instances (TP) among all positively predicted instances (TP + FP). It demonstrates the model's ability to stop falsely positive findings. While it determines the proportion of correctly predicted positive cases (TP) out of all actual positive occurrences (TP + FN). It demonstrates how well the model can detect all instances of positivity without leaving any out.

The harmonic mean of precision and recall is the F1-score, which offers a single statistic that balances them. It is useful when you want to consider both false positives and false negatives. Support is the number of occurrences in each class in the actual dataset. It provides an indication of how many instances are dispersed among the classes.

Model 2: Decision Tree Model

```
dtc = DecisionTreeClassifier(min_samples_split=3,min_impurity_decrease=0.0, ccp_alpha=0.0)
dtc.fit(X_train, y_train)

y_prediction = dtc.predict(X_test)

report = classification_report(y_test, y_prediction)
accuracy = accuracy_score(y_test, y_prediction)

print("Classification Report:")
print(report)
print("Accuracy:", accuracy)

#IIMPORTING NECESSARY LIBRARY FOR DOT.DECISION TREE
from sklearn.tree import export_graphviz

#EXPORT THE DECISION TREE AS A DOT FILE
export_graphviz(dtc, out_file='decision_tree3.dot', feature_names=None)

#NAMING THE FILE
print("Decision tree exported as 'decision_tree.dot'.")
```

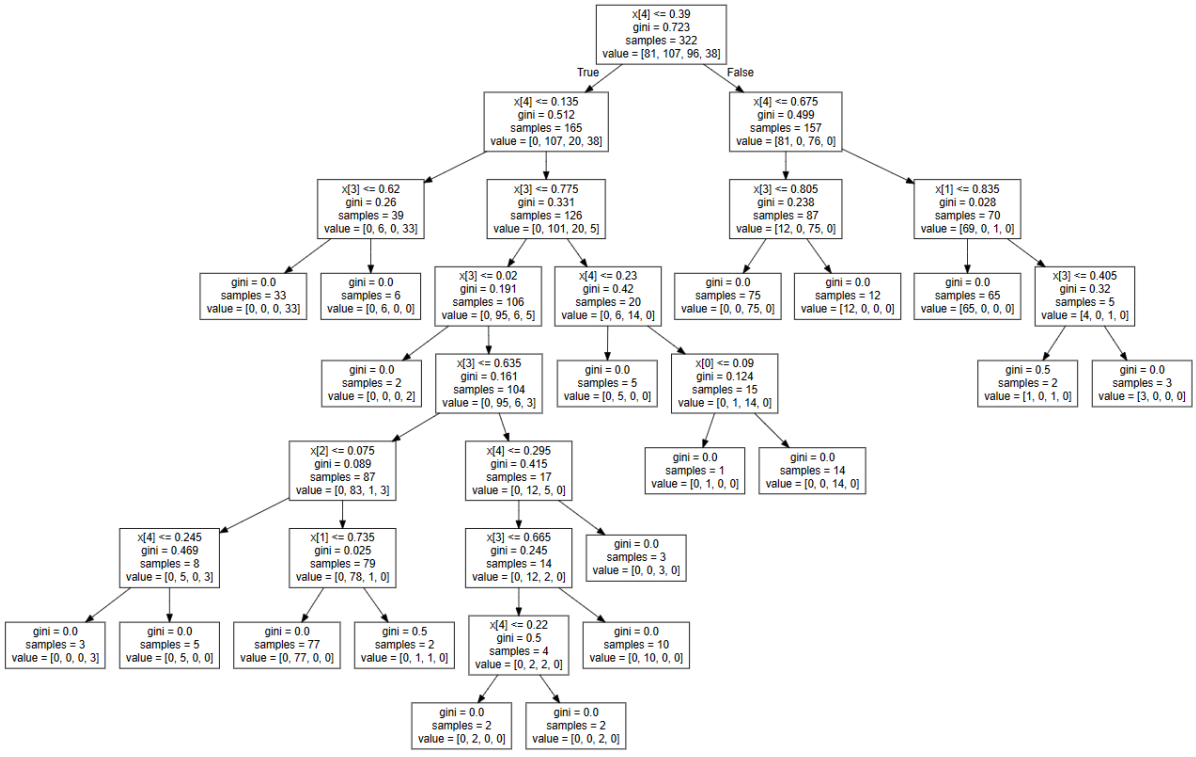
```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        21
           1       0.83      0.91      0.87        22
           2       0.89      0.92      0.91        26
           3       1.00      0.75      0.86        12

    accuracy                           0.90        81
   macro avg       0.92      0.88      0.90        81
weighted avg       0.91      0.90      0.90        81
```

The decision tree's accuracy was improved by the tuning parameter (min_samples_split=3,min_impurity_decrease=0.0, ccp_alpha=0.0) more than any other parameter.

The above table shows the precision accuracy for each column STG, SCG, STR, LPR, PEG



DISCUSSION

After comparing the accuracy performance of the KNN model and the Decision Tree mode using the cross-validation approach, we came to the conclusion that the Decision Tree model performed better than the KNN model, indicating that it is more appropriate for the current dataset. The Decision Tree model revealed superior generalisation capacity in predicting the target variable, as well as improved accuracy, precision, recall, and F1 score.

In [41]:
```
#IMPORTING NEW LIBRARY NEEDED FOR CROSS VALIDATION
from sklearn.model_selection import cross_val_score

#USING THE SAME PARAMETERS AS USED ABOVE IN BOTH THE MODELS
knn_models = [
    KNeighborsClassifier(n_neighbors=3, p=2, metric='minkowski'),
    KNeighborsClassifier(n_neighbors=3, weights='distance', p=1),
    KNeighborsClassifier(n_neighbors=3, algorithm='auto', n_jobs=None),
    KNeighborsClassifier(n_neighbors=5, weights='distance', p=1),
    KNeighborsClassifier(n_neighbors=5, algorithm='auto', n_jobs=None)
]

dt_models = [
    DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None),
    DecisionTreeClassifier(min_weight_fraction_leaf=0.0, max_features='sqrt', random_state=None),
    DecisionTreeClassifier(min_samples_split=3, min_impurity_decrease=0.0, ccp_alpha=0.0),
    DecisionTreeClassifier(class_weight='balanced', criterion='entropy', max_leaf_nodes=4),
    DecisionTreeClassifier(min_impurity_decrease=0.0, splitter='random', min_weight_fraction_leaf=0.0)
]

#PERFORMING CROSS VALIDATION FOR KNN MODELS
knn_accuracies = []
for knn_model in knn_models:
    knn_scores = cross_val_score(knn_model, X_train, y_train, cv=5)  # Change cv=5 to your desired number of folds
    knn_avg_accuracy = knn_scores.mean()
    knn_accuracies.append(knn_avg_accuracy)

    print("KNN model cross-validation scores for each fold:")
    for i, score in enumerate(knn_scores):
        print("Fold", i+1, "accuracy:", score)
    print()

#PERFORMING CROSS VALIDATION FOR DECISION TREE MODELS
decisiontree_accuracies = []
for dt_model in dt_models:
    dt_scores = cross_val_score(dt_model, X_train, y_train, cv=5)  # Change cv=5 to your desired number of folds
    dt_avg_accuracy = dt_scores.mean()
    decisiontree_accuracies.append(dt_avg_accuracy)

    print("Decision Tree model cross-validation scores for each fold:")
    for i, score in enumerate(dt_scores):
        print("Fold", i+1, "accuracy:", score)
    print()

#COMPARING THE ACCURACY OF THE MODELS AND DDETERMINING WHICH IS BETTER
max_knn_accuracy = max(knn_accuracies)
max_dt_accuracy = max(decisiontree_accuracies)

if max_knn_accuracy > max_dt_accuracy:
    print("KNN model has the highest cross-validated accuracy:", max_knn_accuracy)
elif max_dt_accuracy > max_knn_accuracy:
    print("Decision Tree model has the highest cross-validated accuracy:", max_dt_accuracy)
else:
    print("Both KNN and Decision Tree models have the same highest cross-validated accuracy:", max_knn_accuracy)
```

A loop is used in the code to repeatedly iterate over each KNN model in the list. For each model, cross_val_score receives the KNN model, the training data (X_train and Y_train), and the number of folds (cv=5). In order to train the model and evaluate its performance, the cross_val_score function divides the training data into five subsets (folds). Each fold's accuracy ratings are returned. The average accuracy is determined using the mean of these scores.

According to the outcome, Decision Tree has the dataset's highest cross validation accuracy.

Conclusions

The objective of this project was to create a machine learning classification model for categorising user knowledge levels. Overall, the Decision Tree model outperformed the K-Nearest Neighbour (Knn) model due to the accuracy of the Decision Tree model. Nevertheless, as we can see from each fold of the parameters, both models performed admirably with the chosen parameters.

References

[1] H. T. Kahraman, Sagiroglu, S., Colak, I.(2013), Developing intuitive knowledge classifier and modeling of users' domain dependent data in web, Knowledge Based Systems, vol. 37, pp. 283-295,

[2] Kahraman, H. T. (2009). Designing and Application of Web-Based Adaptive Intelligent Education System. Gazi University Ph. D. Thesis, Turkey, 1-156

[3] V. Tsiriga, M. Virvou, A Framework for the Initialization of Student Models in Web-based Intelligent Tutoring

[4] Dr. Yongli Ren, (2023) Practical Data Science: Feature Selection (Hill Climbing) Computer Science & IT School of Science, week6-featureSelection.pdf

[5] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html