The University of Texas at El Paso
CS 4361: Machine Learning

Sip to Success: A Machine Learning Approach to Forecasting Students' Final Grade Based on Alcohol Consumption Patterns

Team #7: Alexis Flores, Jose Sanchez, Nayeli Ramirez

December 9, 2023

Table of Content

I. Introduction

Exploring the realm of academia, we're delving into how lifestyle choices impact academic performance. Our goal is to predict final grades based on alcohol habits, shining a light on the link between lifestyle and academic success. We're curious about how finding a balance in drinking might be a key to better grades. This investigation aims to untangle how much our academic path is influenced by alcohol, adding to what we know about what shapes our time in college. Using data and machine learning, we're hoping to wrap it up with some useful insights that highlight why making smart choices during our college years matters.

II. Procedure

Upon embarking on our project, the recognition of a binary outcome—pass or fail—for students prompted us to categorize it as a classification problem. Opting for the decision tree model, renowned for its adeptness in handling such scenarios, was a logical choice. Once we knew what model we would be using we designed the architecture of our model[1].

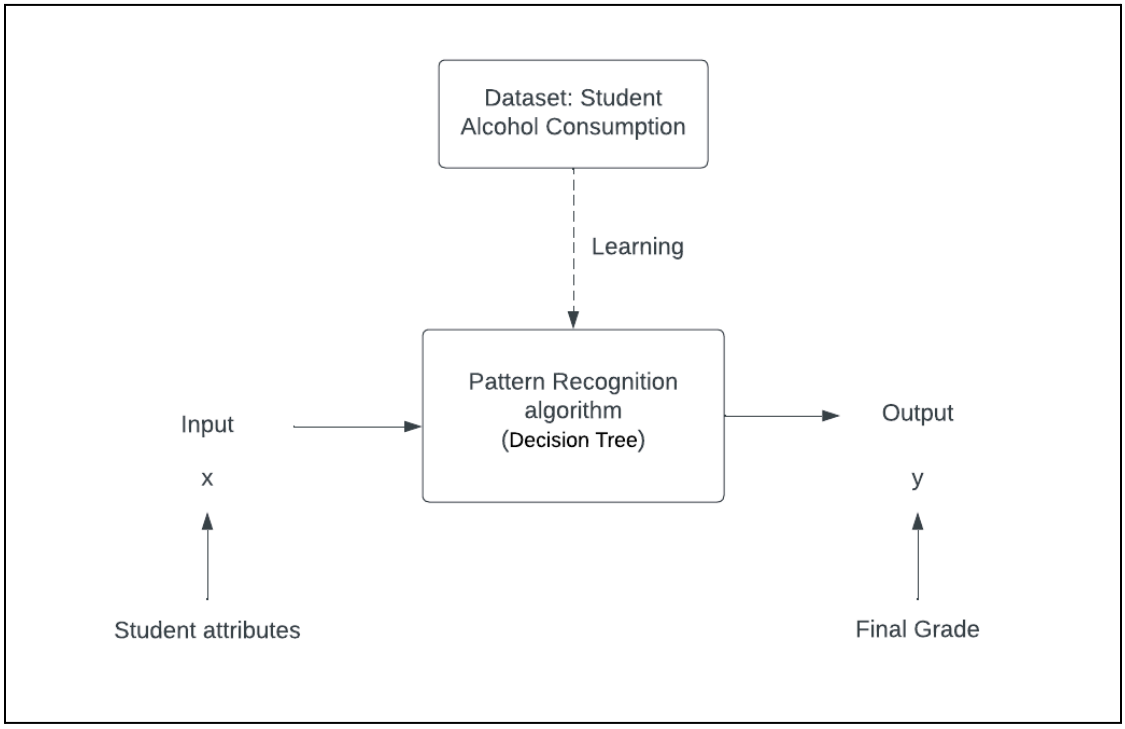

Figure 1. Models Architecture Design

Our strategic blueprint initiated with the acquisition of relevant data from online sources. Subsequently, we immersed ourselves in the intricacies of data preprocessing, meticulously scrubbing extraneous details and transforming numerical grades into a binary pass/fail format, guided by the grading scale from our data source. With our data now finely tuned, we proceeded to judiciously partition it into training and testing sets, a critical step for a comprehensive evaluation of our model's efficacy.

The coding journey of our decision tree began, with a keen focus on precision to ensure seamless integration with our dataset. The initial outcomes, while informative, sparked a realization—we needed a benchmark for comparison. To address this, we embarked on a parallel path, training a K-Nearest Neighbors (KNN) model using the same pristine dataset. With codes in place, we executed the KNN model, introducing an additional layer for evaluating the effectiveness of our decision tree model.

This dual-model approach facilitates a nuanced comparison, allowing us to discern the respective strengths and weaknesses of the decision tree and KNN models in predicting student pass/fail outcomes.

III. Data Set

Our dataset was sourced from Kaggle, a freely accessible platform offering a plethora of datasets. Originally comprising 30 features, our data preprocessing phase involved discerning and retaining only the features integral to the model's learning process, ultimately narrowing it down to 13 significant features that are listed below. Originating from students in Portugal our dataset needed to be converted specific to Portugal's grading scale[2]. The subsequent step involved coding to transform the original numerical final grades into a binary pass or fail format, represented as 0 or 1. Additionally, our code efficiently converted string inputs into numerical equivalents (0s or 1s). To ensure the accurate training and testing of our model, the data was segmented into distinct files through our code. This meticulous data preparation lays the groundwork for our machine learning models, aligning with the strategic approach outlined in our project.

Dataset Filtered Features

1.
2. sex - student's sex (binary: 'F' - female or 'M' - male)
3. age - student's age (numeric: from 15 to 22)

4. studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
5. failures - number of past class failures (numeric: n if 1<=n<3, else 4)
6. schoolsup - extra educational support (binary: yes or no)
7. famsup - family educational support (binary: yes or no)
8. paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
9. higher - wants to take higher education (binary: yes or no)
10. internet - Internet access at home (binary: yes or no)
11. freetime - free time after school (numeric: from 1 - very low to 5 - very high)
12. Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
13. Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
14. G3 - final grade (numeric: from 0 to 20, output target)

Grade	Qualification
20 ⋮ 17.5	Excellent
17.4 ⋮ 15.5	Very good
15.4 ⋮ 13.5	Good
13.4 ⋮ 9.5	Sufficient
9.4 ⋮ 3.5	Weak
3.4 ⋮ 0	Poor

Figure 2. Portuguese Schools Grading Scale

age	studytime	failures	freetime	Dalc	Walc	Finalgrade	sex_F	sex_M	schoolsup	schoolsup	famsup_n	famsup_y	paid_no	paid_yes	higher_nc	higher_ye	internet_	internet_yes
18	2	0	3	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0
17	2	0	3	1	1	0	1	0	1	0	0	1	1	0	0	1	0	1
15	2	3	3	2	3	0	1	0	0	1	1	0	0	1	0	1	0	1
15	3	0	2	1	1	1	1	0	1	0	0	1	0	1	0	1	0	1
16	2	0	3	1	2	0	1	0	1	0	0	1	0	1	0	1	1	0
16	2	0	4	1	2	1	0	1	1	0	0	1	0	1	0	1	0	1

Figure 3. Clean Dataset

IV. Decision Tree

A decision tree is a great fit for predicting final grades based on alcohol consumption because it can understand complex connections in the data, making it useful for this problem. Decision trees are easy to explain, handling different types of information well, like numbers and categories. They're also not easily thrown off by unusual data, making them reliable with real-world information. In summary, decision trees are a practical and effective choice for tackling the relationship between alcohol consumption and student grades in a straightforward manner.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split

X_train = pd.read_csv('X_train.csv')
y_train = pd.read_csv('y_train.csv')
y_train = y_train.values.ravel()
clf = DecisionTreeClassifier()

# Train the classifier
clf.fit(X_train, y_train)

# Load test data
X_test = pd.read_csv('X_test.csv')

# Make predictions on the test data
test_predictions = clf.predict(X_test)

# Save the predicted values to a CSV file
predicted_values = pd.DataFrame({'Predicted_Values': test_predictions})
predicted_values.to_csv('y_predict_decision_tree.csv', index=False)

accuracy = accuracy_score(y_test, test_predictions)
class_report = classification_report(y_test, test_predictions)
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{class_report}')
```

Figure 4. Decision Tree Code

V. KNN

A k-Nearest Neighbor model was a viable option for predicting a student's final grade based on alcohol consumption because it can classify data points based on their neighbors' classification. For the kNN model implementation, we had to change the categorical values to numerical values in the columns sex, schoolsup, famsup, paid, higher, and internet. Next, we split the dataset into training and test sets using scikit-learn. After running the first test of the kNN model our accuracy was 9%. We then changed the test size from 0.8 to 0.5 and that increased our accuracy to 15.15%. But after trying many other changes the accuracy of the model didn't increase. A possible reason for this unsatisfactory result could have been that kNN is meant to be used on small datasets and our dataset was not. Another reason could be that the converting of the categorical features to numerical values was not done correctly. One thing we could have done differently was to implement an additional model like logistic regression where instead of predicting the student's exact final grade the model would have predicted if a student was going to pass or fail the course. The reason for this is that after looking at the predictions made by the kNN model compared to the actual final grades in many cases the difference was only one digit off. So if the predictions would have been pass or fail then I believe the accuracy would have been higher.

```python
from typing_extensions import dataclass_transform
import pandas as pd
from sklearn.model_selection import train_test_split
import math
from collections import Counter
import numpy as np
from tabulate import tabulate

# 1. Load dataset
def load_dataset(filename):
    df = pd.read_csv(filename)
    return df

data = load_dataset('student-mat.csv')
data['sex'] = data['sex'].map({'F': 1, 'M': 1})
data['schoolsup'] = data['schoolsup'].map({'yes': 1, 'no': 0})
data['famsup'] = data['famsup'].map({'yes': 1, 'no': 0})
data['paid'] = data['paid'].map({'yes': 1, 'no': 0})
data['higher'] = data['higher'].map({'yes': 1, 'no': 0})
data['internet'] = data['internet'].map({'yes': 1, 'no': 0})
X = data.drop(columns=['Finalgrade']).values
#print(X)
y = data['Finalgrade'].values

# 2. Split dataset into training and test sets using scikit-learn
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)
print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")


# 3. Define the kNN algorithm
def euclidean_distance(instance1, instance2):
    """Calculate the Euclidean distance between two instances."""
    sum = 0
    for i in range(0, len(instance1)):
      x = (instance2[i] - instance1[i])**2
      sum = sum + x
    eucld_dist = math.sqrt(sum) #calculate distance
    return eucld_dist #return distance

def get_neighbors(X_train, test_instance, k):
    """Get the k nearest neighbors for a test instance."""
    distances = [] #stores all distances from test instance
    neighbors = [] #stores k nearest neighbors


    for i in range(0, len(X_train)): #iterate X training set
      dist = euclidean_distance(test_instance,X_train[i]) #calculate distance
      instance = [dist, i] #[distance, location in training set]
      distances.insert(len(distances), instance) #add curr instance to distances

    #sort distances from shortest to longest distance
    neighbors = sorted(distances)[:k] #assign 3 shortest to neighbors

    return neighbors

def get_response(neighbors, y_train):
    """Determine the class label for a current instance based on the majority
    class label of its k neighbors."""
    prediction = None
```

Figure 5. KNN Code Part 1

```
    predictions = [] #stores class label from all neighbors
    for neighbor in neighbors: #iterate neighbors
      y_location = neighbor[1] #get training set location stored in neighbor
      predictions = predictions + [y_train[y_location]] #add class label to list

    #get most common class label among all neighbors
    most_common = Counter(predictions).most_common(1)
    prediction = most_common[0][0] #assigns prediction with most common label

    return prediction

# 4. Use the kNN algorithm to predict the class labels of the test set
k = 3
predictions = []
for current_instance in X_test:
    neighbors = get_neighbors(X_train, current_instance, k)
    prediction = get_response(neighbors, y_train)
    predictions.append(prediction)

# 5. Calculate the accuracy of the predictions

# assign data

mydata = [['Predicted', 'Actual']]
for i in range(0, 30):
  x = [predictions[i], y_test[i]]
  mydata.append(x)


# display table
print(tabulate(mydata))

correct = sum([y_true == y_pred for y_true, y_pred in zip(y_test, predictions)])
accuracy = (correct / len(y_test)) * 100.0
print(f"Accuracy: {accuracy:.2f}%")
#'''
```

Figure 6. KNN Code Part 2

VI. Results

```
Accuracy: 0.6455696202531646
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.79      0.75        53
           1       0.45      0.35      0.39        26

    accuracy                           0.65        79
   macro avg       0.58      0.57      0.57        79
weighted avg       0.63      0.65      0.63        79
```

Figure 7. Decision Tree Results

```
Training set size: 197
Test set size: 198
---------  ------
Predicted  Actual
8          10
8          12
0          5
13         10
7          9
9          13
0          18
9          6
19         0
0          14
16         15
14         7
11         15
12         10
11         14
9          8
13         8
9          11
15         15
11         0
16         14
12         16
13         16
9          6
12         0
12         19
18         11
15         12
14         17
10         10
---------  ------
Accuracy: 15.15%
```

Figure 8. KNN Results

VII. Conclusion

In conclusion, the decision tree model emerged as the better choice for predicting final grades based on alcohol consumption in our study. Despite the accuracy not being exceptionally high, we attribute this to the features chosen for the model. It's important to acknowledge that the selection of features significantly influences model performance. In future iterations, considering different features or refining the existing ones might lead to improved accuracy for both training and testing models.

VIII. References

"Academic Grading in Portugal." *Wikipedia*, Wikimedia Foundation, 12 Dec. 2018, en.wikipedia.org/wiki/Academic_grading_in_Portugal.