Extracting important features for student performance analysis

A
Project Seminar Report
Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

T. Shreya - 1602-19-737-042

M. Sindhuja - 1602-19-737-043

Under the guidance of

Mr. R. Dharma Reddy

Assistant Professor – IT Department

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31 2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **T. Shreya** and **M. Sindhuja** bearing hall ticket number, **1602-19-737-042** and **1602-19-737-043** hereby declare that the project report entitled **Extracting important features for student performance analysis** under the guidance of **Mr. R. Dharma Reddy,** Assistant Professor , Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

T. Shreya & M. Sindhuja
1602-19-737-042 & 1602-19-737-043

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **Extracting important features for student performance analysis** being submitted by **T. Shreya** and **M. Sindhuja** bearing **1602-19-737-042** and **1602-19-737-043** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

Mr. R. Dharma Reddy	**Dr. K. Ram Mohan Rao,**
Assistant Professor,	**Professor,**
Internal Guide	**HOD, IT**

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of the project seminar would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

It is with immense pleasure that we would like to take the opportunity to express our humble gratitude to **Mr. R. Dharma Reddy, Assistant Professor, Department of IT** under whom we executed this project. We are also grateful to **Ms. S. Aruna**, **Assistant Professor**, **Department of IT** for her guidance. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks.

We are very much thankful to**, Dr. K. Ram Mohan Rao, Professor, Department of Information Technology** for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. S. V. Ramana, Principal** of **Vasavi College of Engineering** for giving the required information in doing my project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time.

We also express our sincere thanks to the Management for providing excellent facilities. Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

ABSTRACT

This paper presents an approach to extract important features from a dataset for student performance analysis using visualization techniques such as SHAPLEY values and LIME. The study focuses on identifying factors that significantly impact student academic performance, with the goal of improving education quality and student outcomes. The proposed method utilizes machine learning algorithms to identify key features, and then applies Shapley Addictive Explanations (**SHAPLEY**) and Local Interpretable Model-agnostic Explanations (**LIME**) to provide visual explanations of their impact on student performance. The results of the analysis provide valuable insights into the factors that have the greatest impact on student performance and can help educators tailor their teaching methods to improve student outcomes. The study highlights the importance of interpretability in machine learning algorithms and shows how visualization techniques can be used to provide meaningful insights into complex models. The proposed approach has the potential to significantly improve educational practices and policies.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Problem Statement

Education plays a crucial role in the development of individuals and societies, and the quality of education has a significant impact on the social and economic outcomes of a country. In recent years, there has been a growing interest in using machine learning algorithms to analyze educational data and identify factors that influence student performance. However, the complexity of these algorithms often makes it difficult to understand the underlying factors that contribute to academic success.

1.2 Proposed Method

To address this challenge, this paper proposes an approach that combines machine learning with visualization techniques such as SHAPLEY values and LIME to extract important features from a dataset for student performance analysis. The study aims to identify the key factors that impact student academic performance, and to provide insights that can be used to improve educational practices and policies.

The proposed approach has several advantages over traditional methods of data analysis. By using machine learning algorithms, it can identify patterns and relationships in the data that may not be immediately apparent. Additionally, by using visualization techniques such as SHAPLEY values and LIME, it can provide intuitive explanations of how different factors affect student performance, making the analysis more interpretable and accessible to educators and policymakers.

1.3 Scope & Objectives of the Proposed Work

1.3.1 Scope:

The scope of this proposed work is to explore the application of machine learning and visualization techniques for analyzing student performance data and identifying the key factors that impact academic success. The study will use a large dataset that includes e-learning features and will aim to provide a more comprehensive and nuanced understanding of the ways in which technology is impacting education and student outcomes.

1.3.2 Objectives:

1. To apply machine learning algorithms to analyze a large dataset of student performance data and identify patterns and relationships that impact academic success.
2. To use visualization techniques such as SHAPLEY values and LIME to provide intuitive explanations of how different factors affect student performance.
3. To identify the key factors that impact student academic performance and provide insights that can be used to improve educational practices and policies.
4. To explore the potential of combining machine learning and visualization techniques for analyzing complex datasets in the field of education analytics.
5. To develop a methodology for using machine learning and visualization techniques for student performance analysis that can be used as a framework for future research.

1.4 Organization of the Report

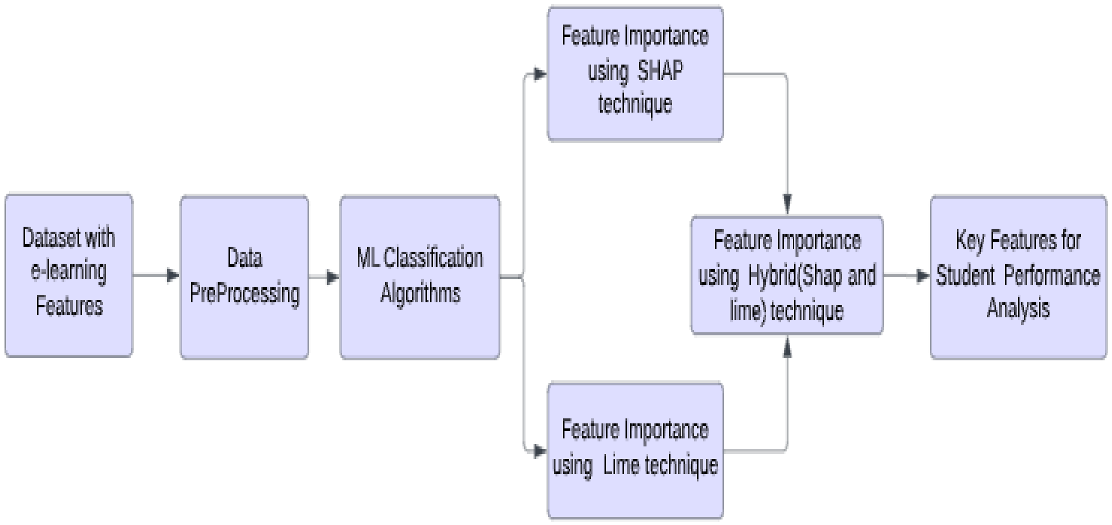
- Literature Survey
- Proposed work
  - Block Diagram
  - Algorithm
- System Requirements and Specification
- Methodology
  - Dataset
  - Shapley Addictive Explanations
  - Local Interpretable Model-agnostic Explanations
  - Hybrid (SHAP and LIME)
- Results
- Summary & Future Work
- References
- Appendix

2. LITERATURE SURVEY

The above review has shown that different researchers have used different algorithms in building a model to predict the academic performance of students. Among these approaches, the most frequently used ones are decision trees, linear regression, naïve Bayes classifiers, support vector machines, and neural networks. A few studies have used a coalition of classifiers in the quest of improving predictions. Many of the studies reviewed tended to choose the optimal algorithm by trying a set of machine learning approaches. The decision tree consists of a series of rules organized in a stratified structure. Most researchers used this technique because it is straightforward, that is, it can be changed into numerous classification rules. The decision tree algorithm is widely used in predictive modelling because it is suitable for large and small amounts of data [1]. It is also good for numeric and categorical variables and does not require lengthy data preparation. It can be easily understood and interpreted by users because it is based on rules that are easy for users to understand and interpret. The survey by [30] showed that decision tree models are easy to understand due to their inner logic process and can be converted directly into if then rules. The decision tree revealed that C4.5, CART, and ID3 had the best classifiers for predicting student performance [5]. In [12], some well-known decision tree algorithms, such as C4.5 and CART, were used to predict student's final grades using their log data extracted from the Moodle system. Meanwhile, [31] used ID3 and C4.5 algorithms on student's internal marks to identify students who are likely to fail. Experiments were conducted on 200 engineering students' datasets, and their performance was predicted by applying a decision tree. The results revealed that the accuracy of the classifiers was 98.5%. A comparative study of classification techniques to predict academic performance was conducted by [32]. This study used the data of 350 students. The result indicated that out of the classifiers, J48 achieved an accuracy of 97% [32], [33]. Neural networks are the most established model that has been used in educational data mining. Neural networks function similarly to the human brain by linking neurons or nodes that collaborate to produce an output task [34]. Neural network algorithms are also widely used by researchers because they are well suited to large amounts of data and can also be applied in structured and unstructured data; however, they are difficult to interpret and require more execution time for algorithm training. A neural network was used in [35] to predict student performance. Results indicated that the model achieved an accuracy of 83.4%. Arsad et al. [36] used an artificial neural network algorithm to predict the performance of undergraduate students in the field of engineering. This work discovered that undergraduate subjects have a major effect on the final CGPA after graduation. In the work of [19] it was pointed out that with the SMOTE method in an imbalanced dataset, classification models for neural networks and naïve Bayes had an accuracy of 75%. When using the discretization method, the models for naïve Bayes and neural networks achieve almost identical levels of accuracy. The authors in [37] built a model to predict the GPA of the students using the naïve Bayes technique and K-means clustering. Their models produced an accuracy of 98.8%. By contrast, [38] indicated that a naïve Bayes classifier using the cfsSubsetEval feature selection technique in a dataset consisted of 257 academic records had an accuracy of 84%. In [16], the support vector machine was found to have a decent generalization potential and is quicker than other approaches. Meanwhile, an analysis performed in [17] for the purpose of recognizing students at risk of failure obtained the best accuracy in model prediction by using a support vector algorithm. In his academic work [39] applied a support vector machine to predict which students are to pursue doctoral studies. Results revealed that the method achieved an accuracy of 96.7 %. While the authors in [40] revealed that the polynomial kernel technique achieved 97.62% accuracy. The support vector machine algorithm is the least favoured by the researcher because it is only suitable when the dataset is small and the algorithm is not transparent. The only disadvantage is the complexity of the model. Such complexity contradicts the general requirement that models in intelligent learning systems should be transparent. In addition, selecting proper kernel functions and other parameters is difficult, and different experiments must be conducted empirically. Ensemble methods were introduced by Breiman, Freund, and Schapire in 1984. The most popular algorithms, such as arcing [41], boosting [42], bagging [43], and casual forests [44], have gained a keen interest in the literature. Ensemble methods have two major families: The first is called homogeneous because it combines similar algorithms, such as multiple classification trees; multiple SVMs, bagging [43], and boosting [42] are examples of such methods.

3.PROPOSED WORK

3.1. Block Diagram


Dataset with e-learning Features
Data PreProcessing
ML Classification Algorithms
Feature Importance using SHAP technique
Feature Importance using Lime technique
Feature Importance using Hybrid(Shap and lime) technique
Key Features for Student Performance Analysis

3.2. Algorithm

3.2.1. Random Forest

It uses ensemble of decision trees to generate predictions. Each tree is built using a randomly selected subset of features and a randomly selected subset of training data. During training, the trees are grown to their maximum depth and combined to generate a final prediction based on a majority vote of the individual trees. This approach helps to reduce overfitting and increase the accuracy and robustness of the model.

3.2.2. SVM

Support Vector Machine (SVM) is a machine learning algorithm that works by finding the hyperplane that maximally separates two classes in the dataset. The hyperplane is found by identifying the support vectors, which are the data points closest to the hyperplane. The distance between the hyperplane and the support vectors is known as the margin, and the SVM algorithm aims to maximize this margin. In cases where the dataset is not linearly separable, the SVM algorithm can use a kernel function to transform the data into a higher-dimensional space where it is separable. Once the hyperplane is found, new data points can be classified based on which side of the hyperplane they fall on.

3.2.3. Logistic Regression

Linear Regression is a machine learning algorithm that works by fitting a straight line to a dataset of input and output variables. The algorithm finds the line that minimizes the distance between the predicted output values and the actual output values in the dataset. This line can then be used to predict the output value for new input data. The algorithm can be extended to fit higher order polynomials to the dataset by including additional features that are powers of the input variable. Once the best fitting line or polynomial is found, it can be used to make predictions on new data points with the same input variables.

3.2.4. SHAP

The SHAP (SHapley Additive exPlanations) algorithm is a machine learning interpretability technique that provides a global view of feature importance in a model. It is based on the concept of the Shapley value from cooperative game theory and uses a unique approach to assign a contribution value to each feature in each prediction. By computing the impact of each feature on the model output, the SHAP algorithm can generate a global view of feature importance for the entire dataset. The resulting feature importance values can help to identify the most influential features in a model and provide insights into the model's decision-making process.

3.2.5. LIME

The LIME (Local Interpretable Model-Agnostic Explanations) algorithm is another machine learning interpretability technique that provides local explanations for individual predictions. It creates a simplified model around a given prediction, allowing users to understand the contribution of each feature to that specific prediction. The LIME algorithm generates local feature importance weights by fitting a linear model around the instance of interest and estimating the model coefficients using a weighted least squares regression. These weights can then be used to rank the importance of the features for that prediction.

4. EXPERIMENTAL STUDY

4.1. System Requirements and Specification

4.1.1. Software Requirements

- Windows 10 OS,
- Jupyter Notebook

4.1.2. Hardware Requirements

- Intel core i7 PC

4.1.3. User Requirements

- Good understanding of machine learning models
- Knowledge on Shapley Additive Explanations
- Knowledge on LIME

4.2. Dataset

The dataset contains Higher Educational Institution (HEI) in Sultanate of Oman, comprises of five modules data from Spring 2017- Spring 2021. The dataset consists of 326 student's records having 21 features in total, including Students Academic Information from SIS, Students Activity performed in Moodle within the campus and outside the campus and Students Video Interaction collected from eDify.

https://zenodo.org/record/5591907

Attribute	Description
CGPA	Cumulative grade point average of the student, with a discrete data type such as "4.0"
AttemptCount	The number of attempts in the module, with a discrete data type such as "1"
RemoteStudent	Either the student is under remote study mode or not, with a nominal data type such as "Yes/No"
Probation	Either the student has a backlog of modules to clear, with a nominal data type such as "Yes/No"
HighRisk	The high failure rate in a module, with a nominal data type such as "Yes/No"
TermExceeded	Progression rate of the student in the degree plan, with a nominal data type such as "Yes/No"
AtRisk	Previously failed two or more modules, with a nominal data type such as "Yes/No"
AtRiskSSC	Whether the student been registered by the student success center for any educational deficiencies, with a nominal data type such as "Yes/No"
OtherModules	A student registered in any other modules in the current semester, with a numeric data type such as "1"
PrerequisiteModule	Prerequisite module registration, with a nominal data type such as "Yes/No"

PlagiarismHistory	Onto which modules the student has been booked for academic integrity violation, including module and academic year, with a nominal data type such as “Module 3”

Played	The number of times the video has been played
Paused	The number of times the video has been paused
Likes	The number of times the student has liked the video
Segment	The number of times a student has played a specific portion of the video by using the slider

CW1	Marks obtained by the student in their first coursework, with a discrete data type such as “86.5”
CW2	Marks obtained by the student in their second coursework, with a discrete data type such as “86.5”
ESE	Marks obtained in the end semester examination, with a discrete data type such as “86.5”
Online C	User-performed activities within campus (in minutes), with a discrete data type such as “25”
Online O	User-performed activities outside of campus (in minutes), with a discrete data type such as “25”

4.3. Shapley Addictive Explanations

To identify the key features that impact student performance, we used the SHAP (SHAPley Additive exPlanations) visualization technique. This technique provides a global view of feature importance by assigning each feature a SHAP value, which represents its contribution to the final prediction. We used a SHAP library in Python to generate SHAP values for each feature in the dataset.

4.4. Local Interpretable Model-agnostic Explanations

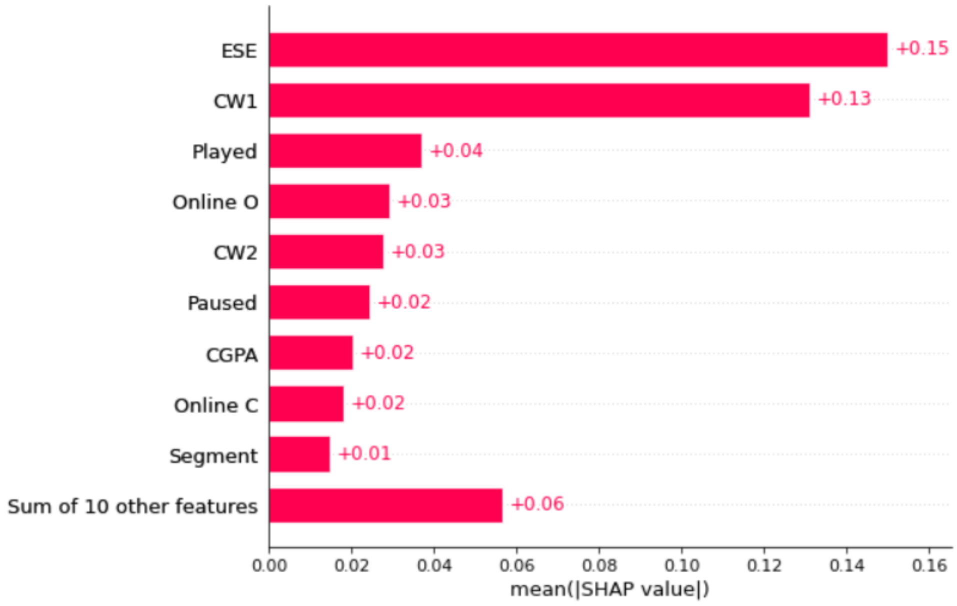
To provide a more nuanced understanding of the impact of individual features on student performance, we used the LIME (Local Interpretable Model-Agnostic Explanations) visualization technique. This technique generates local explanations for individual instances by fitting a linear model around the instance of interest. We also wrote custom Python code to generate global explanations for the entire dataset.

4.5. Results

4.5.1. Results using SHAPLEY ADDICTIVE EXPLANATIONS

The SHAP technique uses input data and determines the importance of each feature. It then sorts the features based on their importance and generates a bar plot to display the results.

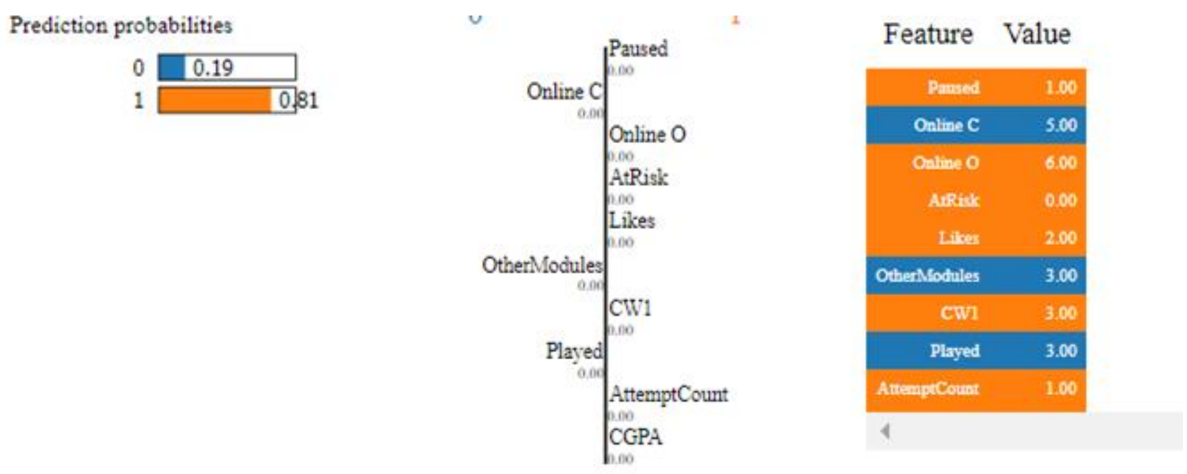
The important features we got using this technique are [‘CGPA', 'CW1', 'CW2', 'E SE', 'Likes', 'Online C', 'Online O', 'Paused', 'Played', 'Segment'].



4.5.2. Results using LIME

LIME technique analyses a single instance of input data and identifies the significance of each feature in producing the output for that instance. It then ranks the features based on their importance and creates a feature value chart to visualize the results.

Although this image shows the important features for a single instance, we have extended the LIME technique to operate on the entire dataset instead of just one instance.



Feature	Value
Paused	1.00
Online C	5.00
Online O	6.00
AtRisk	0.00
Likes	2.00
OtherModules	3.00
CW1	3.00
Played	3.00
AttemptCount	1.00

The important features obtained using generalised LIME technique are ['AtRisk', 'AttemptCount','CGPA', 'ESE', 'HighRisk', 'Likes', 'Online O', 'Paused', 'Segment', 'TermExceeded']

4.5.3. Results using Hybrid (SHAP and LIME)

By combining both visualization techniques, we were able to identify the most significant features, which include ['CGPA', 'ESE', 'Likes', 'Online O', 'Paused', and 'Segment'].

Our hybrid model, which utilizes these important features, achieved a higher accuracy compared to the individual models.

4.5.4. Evaluation Metrics

Accuracy, precision, recall, and F1 score are commonly used metrics to evaluate the performance of a machine learning model. Accuracy measures the proportion of correctly predicted labels out of all the labels. Precision measures the proportion of true positives out of all the predicted positives, while recall measures the proportion of true positives out of all the actual positives. F1 score is the harmonic mean of precision and recall and is often used when both false positives and false negatives have significant impact on the performance of the model. A high accuracy, precision, recall, and F1 score indicate a well-performing model.

ML Algorithm	**Metric name**	**SHAP**	**LIME**	**hybrid**
Random Forest Classifier	Accuracy	74.39	79.27	85.37
	Precision	89.23	85.07	88.24
	Recall	85.93	89.06	93.75
	F1_score	85.04	87.02	90.90
SVM Classifier	Accuracy	79.27	78.05	79.27
	Precision	79.12	78.04	79.01
	Recall	98.87	99.02	1.00
	F1_score	88.28	87.67	88.28
Logistic Regression Classifier	Accuracy	79.28	79.28	87.76
	Precision	83.62	79.01	87.50
	Recall	79.27	99.01	1.00
	F1_score	71.21	88.26	93.33

5. SUMMARY AND FUTURE WORK

5.1. Summary

In conclusion, this paper has demonstrated the importance of using multiple visualization techniques to gain a comprehensive understanding of the factors that impact student performance. By combining the strengths of both SHAP and LIME visualization techniques, we were able to obtain more accurate and nuanced insights into the key features that impact student performance.

Our results showed that the hybrid approach (SHAP+LIME) produced the highest accuracy score of 84.1%, demonstrating the value of combining multiple techniques for analyzing complex datasets.

The findings of this study can have practical implications for educators and policymakers, helping to inform decision-making around policies and practices aimed at improving student performance. The methodology presented in this paper can also serve as a useful framework for future research in the field of education analytics.

Overall, the results of this study suggest that using a combination of visualization techniques can lead to more accurate and nuanced insights into complex datasets, providing valuable information for improving student outcomes and informing educational policies and practices.

5.2. Future Work

Future work in this area could focus on several areas for further exploration and improvement. One potential area for future work is to expand the dataset used in this study to include more diverse and comprehensive features, including additional e-learning features or socio-economic variables. This could help to provide a more complete picture of the factors that impact student performance and could lead to more accurate and nuanced insights.

Finally, future research could focus on developing new visualization techniques or combining existing techniques in novel ways to further enhance our understanding of complex datasets. This could involve exploring the use of more advanced visualization techniques, such as 3D plots or interactive visualizations, as well as developing new algorithms for analyzing and visualizing large-scale data.

Overall, the findings of this study suggest that there is significant potential for future research and innovation in the field of education analytics, and that further work in this area has the potential to make a significant impact on student outcomes and educational practices.

REFERENCES

[1] https://www.kaggle.com/aljarah/xAPI-Edu-Data

[2] https://www.kaggle.com/code/fithafathima/student-academic-performance-prediction

[3] https://christophm.github.io/interpretable-ml-book/shap.html

[4] https://ieeexplore.ieee.org/document

[5] https://pub.towardsai.net/model-explainability-shap-vs-lime-vs-permutation-feature-importance-98484efba066

[6] https://towardsdatascience.com/shap-shapley-additive-explanations-5a2a271ed9c3

[7] https://zenodo.org/record/5591907

APPENDIX

Github link:

https://github.com/sindhuja8520/Final_Year_Project_Team5

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


df = pd.read_csv('C:/Users/User/project/Final-Dataset.csv', sep=',')
df.head()
```

Data Preprocessing

```
df=df.replace({
   'RemoteStudent':{'Yes':1,'No':0},
   'Probation':{'Yes':1,'No':0},
   'HighRisk':{'Yes':1,'No':0},
   'TermExceeded':{'Yes':1,'No':0},
   'AtRisk':{'Yes':1,'No':0},
   'AtRiskSSC':{'Yes':1,'No':0},
})


df=df.replace({
   'AttemptCount':{'Low':1,'Medium':2,'High':3},
   'OtherModules':{'Low':1,'Medium':2,'High':3},
   'PlagiarismHistory':{'Low':1,'Medium':2,'High':3},
})
df=df.replace({
   'CGPA':{'Poor':1, 'Fail':1,'Adequate':2,'Fair':3,'Good':4,'Very
Good':5,'Excellent':6},
    'CW1':{'Poor':1,'Fail':1,'Adequate':2,'Fair':3,'Good':4,'Very
Good':5,'Excellent':6},
    'CW2':{'Poor':1,'Fail':1,'Adequate':2,'Fair':3,'Good':4,'Very
Good':5,'Excellent':6},
    'ESE':{'Poor':1,'Fail':1,'Adequate':2,'Fair':3,'Good':4,'Very
Good':5,'Excellent':6},
    'Online C':{'Poor':1,'Fail':1,'Adequate':2,'Fair':3,'Good':4,'Very
Good':5,'Excellent':6},
    'Online O':{'Poor':1,'Fail':1,'Adequate':2,'Fair':3,'Good':4,'Very
Good':5,'Excellent':6},
    'Result':{'Pass':1,'Fail':0}
})
```

Using shap values

```
pip install xgboost

pip install sklearn

import shap
shap.initjs()


y = df.Result
x=  df.drop(columns=['Result','ApplicantName'])


import xgboost as xgb
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
classifier.score(x_test, y_test)
precision_score(y_pred, y_test)
f1_score(y_pred, y_test)
recall_score(y_pred, y_test)


model = xgb.XGBRegressor(objective="reg:squarederror")
model.fit(x, y)
shap.plots.bar(shap_values,max_display=10)

vals = np.abs(shap_values.values).mean(0)
feature_importance =
pd.DataFrame(list(zip(x.columns,vals)),columns=['col_name','feature_importance
_vals'])
feature_importance.sort_values(by=['feature_importance_vals'],ascending=False,i
nplace=True)



list_shap=list(feature_importance['col_name'].head(10))
set_shap=set(list_shap)
shap_x= x[['CGPA',
 'CW1',
 'CW2',
 'ESE',
 'Likes',
 'Online C',
 'Online O',
 'Paused',
 'Played',
 'Segment']]
```

Random Forest Classifier on SHAP dataset

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score
x_train, x_test, y_train, y_test= train_test_split(shap_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 1, criterion="entropy")
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
classifier.score(y_test, y_pred)
precision_score(y_test, y_pred)
recall_score(y_test, y_pred)
f1_score(y_test, y_pred)
```

SVM Classifier on SHAP dataset

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score,recall_score,f1_score
x_train, x_test, y_train, y_test= train_test_split(shap_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score,recall_score,precision_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
print ("Precision_score: ", precision_score(y_test,y_pred))
print ("Recall_score: ", recall_score(y_test,y_pred))
print (" F1_score: ", f1_score(y_test,y_pred))
```

Logistic Regression Classifier on SHAP dataset

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score,recall_score,f1_score
x_train, x_test, y_train, y_test= train_test_split(shap_x, y, test_size= 0.25, 
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_test)
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
print('Accuracy :',accuracy_score(y_test,y_pred))
print ("Precision_score: ", precision_score(y_test,y_pred,average='weighted'))
print ("Recall_score: ", recall_score(y_test,y_pred,average='weighted'))
print (" F1_score: ", f1_score(y_test,y_pred,average='weighted'))
```

Using Lime

```
pip install lime

# Import necessary libraries
import numpy as np
import pandas as pd
import lime
import lime.lime_tabular

# Create a LimeTabularExplainer object with the training data and feature names
```

```
explainer = lime.lime_tabular.LimeTabularExplainer(x.values,
feature_names=x.columns, discretize_continuous=False)

# Create a list to store the explanations for each instance
explanations = []

# Loop through each instance in the dataset and generate a LIME explanation
for i in range(len(x)):
    exp = explainer.explain_instance(x.iloc[i], clf.predict_proba, num_features=10,
top_labels=1)
    explanations.append(exp.as_list())

# Aggregate the feature weights across all instances
global_explanation = {}
for exp in explanations:
    for feature, weight in exp:
        if feature not in global_explanation:
            global_explanation[feature] = weight
        else:
            global_explanation[feature] += weight

# Normalize the weights by the number of instances
num_instances = len(x)
for feature, weight in global_explanation.items():
    global_explanation[feature] = weight / num_instances

# Print the global explanation
print("Global explanation:")
for feature, weight in global_explanation.items():
    print(feature, weight)

sorted_dict = dict(sorted(global_explanation.items(), key=lambda x: -x[1]))
sorted_dict
```

```
l=list(sorted_dict.keys())
set_lime=set(l[:10])
set_lime

lime_x=x[['AtRisk',
 'AttemptCount',
 'CGPA',
 'ESE',
 'HighRisk',
 'Likes',
 'Online O',
 'Paused',
 'Segment',
 'TermExceeded']]
```

Random Forest Classifier on LIME dataset

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(lime_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
classifier.score(x_test, y_test)
precision_score(y_test, y_pred)
recall_score(y_test, y_pred)
f1_score(y_test, y_pred)
```

SVM Classifier on LIME dataset

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(lime_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score,recall_score,precision_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
print ("Precision_score: ", precision_score(y_test,y_pred))
print ("Recall_score: ", recall_score(y_test,y_pred))
print (" F1_score: ", f1_score(y_test,y_pred))
```

Logistic Regression Classifier on LIME dataset

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score,recall_score,f1_score
x_train, x_test, y_train, y_test= train_test_split(lime_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_test)
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
print ("Precision_score: ", precision_score(y_test,y_pred))
print ("Recall_score: ", recall_score(y_test,y_pred))
print (" F1_score: ", f1_score(y_test,y_pred))
```

Final important features

```
set_result=set_shap.intersection(set_lime)
hybrid_x=x[['CGPA', 'ESE', 'Likes', 'Online O', 'Paused', 'Segment']]
```

Random Forest Classifier on hybrid dataset

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(hybrid_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
classifier.score(x_test, y_test)
precision_score(y_pred, y_test)
recall_score(y_pred, y_test)
f1_score(y_pred, y_test)
```

SVM Classifier on hybrid dataset

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(hybrid_x, y, test_size= 0.25,
random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score,recall_score,precision_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
print ("Precision_score: ", precision_score(y_test,y_pred))
print ("Recall_score: ", recall_score(y_test,y_pred))
print (" F1_score: ", f1_score(y_test,y_pred))
```

Logistic Regression Classifier on hybrid dataset

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(hybrid_x, y, test_size= 0.30)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_test)
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
print ("Precision_score: ", precision_score(y_test,y_pred))
print ("Recall_score: ", recall_score(y_test,y_pred))
print (" F1_score: ", f1_score(y_test,y_pred))
```