FORECASTING COLLEGE ADMISSIONS WITH DECISION TREE CLASSIFICATION

ABSTRACT

College admission is a crucial decision that affects the future of numerous students. However, predicting the outcome of the admission based on the attributes can be challenging due to the complexity and uncertainty of the process. In this report, we propose an application of decision tree classification in forecasting college admissions. Decision tree classification is a supervised learning technique that can handle both categorical and numerical features, and produce interpretable and few prediction rules.

Leveraging historical admission data from students, the model aims to predict future admission outcomes based on various criteria such as academic aggregate, standard test scores, research papers, and work experiences.

The study evaluates the effectiveness of decision trees in handling complex admission patterns and compares their performance to admission methods. From these insights, students will be able to assess universities, universities can make decisions rapidly, enhance transparency, and contribute to optimizing the admission selection process.

Introduction

The transition from high school to college is a pivotal phase in a student's academic journey. For aspiring undergraduates and graduates, gaining admission to their desired college represents a significant achievement, opening a world of opportunities. However, the college admission process is often surrounded by complexity, leaving students and universities struggling with uncertainty.

Traditional methods of selecting students, while rooted in established practices, may not fully capture academic potential, and may overlook different factors that contribute to a student's success.
Acknowledging these challenges, this project goes into data-driven decision-making, exploring the potential of decision tree classification to transform the college admission process. Decision tree classification is a powerful machine-learning technique, that has demonstrated remarkable success in different domains. Using this algorithm, we aim to develop a predictive model that can accurately forecast college admission decisions.

Motivation

The project idea was obtained from the Training student data based on various fields such as Bachelor's aggregate, GRE, TOEFL, IELTS, Duolingo, Research papers and so on. Instead of displaying the list of eligible students for admission to the college, the project helps the university to validate the students if he/she is eligible and helps students check their eligibility in the portal. Classification algorithms were studied earlier hence ID3 decision tree algorithm was implemented.

The objective of this application is to enhance the efficiency and transparency of the college admission process. By using the decision tree classification, we can evaluate student applications, considering multiple factors such as academic grades, standardized test scores, research papers, and work experiences. This approach has the potential to find the best students from vast applications.

This project also aims to understand the numerous factors that affect college admission decisions. This information will help universities make better decisions about who to admit, ensuring that the process is fair and considers all the categories of each applicant. The impact of this research goes beyond college admissions and could change how institutions use data to make decisions. As colleges use more data, methods like decision tree classification could help them use resources better, improve student support, make the experience fairer, and help students assess their admission to their interested institutions.

Data Collection and Preparation

Every Machine learning algorithm requires both training and testing data and it is very important to have an accurate training data to train the model correctly and effectively.

In our project we are generating the training data using trainingcsv.py file and we are using various fields such as bachelors aggregate, GRE etc. to obtain the result and prediction for the test data.

Feature	Explanation	Data Type
Name	Name of the student	String
Aggregate	Scores of the student in his/her Previous degree	float64
GRE	GRE scores of the student	int64
TOEFL	TOEFL scores of the student	int64
Duolingo	Duolingo scores of student	int64
Backlogs	Count of Backlogs of the student in his/her previous degree	int64
Research Papers	Count of Research papers done by the student	int64
LOR	Count of LOR's submitted by the student	int64
Work Experience	Work experience of the student	Boolean
Prediction	Prediction (admitted/rejected with reason)	String

Using the training dataset, the algorithm is generated. Below is the sample training data used to train the model.

dataset

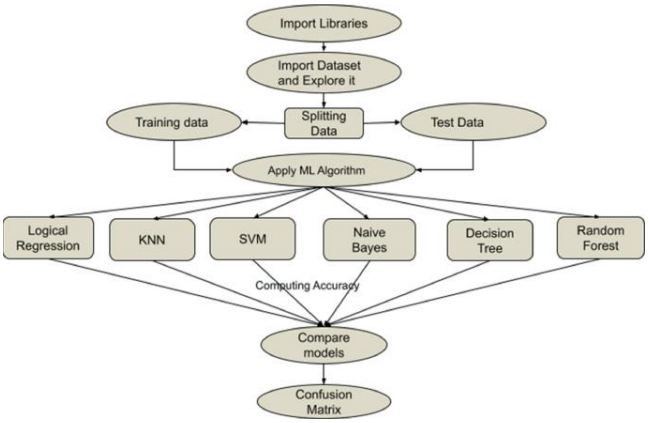
name	Bachelors Aggregate	IELTS	TOEFL	Duolingo	Backlogs	Research Papers	LOR	Work experience in years	GRE	Predictio
Allison Garcia	9.5	7.0	0	89	0	0	4	0	322	Yes
Robert Wall	6.9	8.0	0	93	0	1	4	4	319	Yes
Jerry Jones	7.5	8.0	108	96	1	1	2	5	298	Yes
Dawn Wright	6.5	8.5	97	78	0	0	1	1	320	No
David Webb	9.0	0	0	72	0	0	4	2	302	No
Terry Kaufman	8.5	7.5	109	0	1	0	4	2	298	Yes
Jacob Torres	6.5	0	0	85	0	0	3	3	299	No
Maria Bennett	8.5	7.0	103	77	0	1	2	4	290	Yes
Cody Grimes	9.0	7.0	80	86	1	0	3	5	335	Yes
Brittany Reyes	8.5	0	0	0	0	0	2	2	292	No
Kyle Morgan	9.5	7.0	0	91	0	0	4	0	326	Yes
Philip Elliott	9.5	0	115	0	1	1	3	4	297	Yes
Aaron Turner	8.5	0	107	94	0	1	4	1	309	Yes
Laurie Moran	7.5	0	86	0	0	1	2	4	311	Yes
Andrew Kelly	7.5	7.0	108	100	0	1	4	4	327	Yes
Daniel Hamilton	9.5	5.0	0	78	0	1	4	0	309	No
Paula King	6.0	8.0	94	80	0	0	1	0	310	Yes
Laura Willis	6.5	5.5	0	0	1	1	4	5	321	No
Carrie Martinez	9.5	7.5	107	100	1	1	3	4	322	Yes
Sharon Williams	7.5	7.0	113	98	0	0	1	4	340	Yes
Alexis Chen	7.0	8.0	107	83	0	0	2	0	291	No

And after the model is trained and tested for some test data, it validates through the decision tree for any test data given by the user

Data Mining Techniques

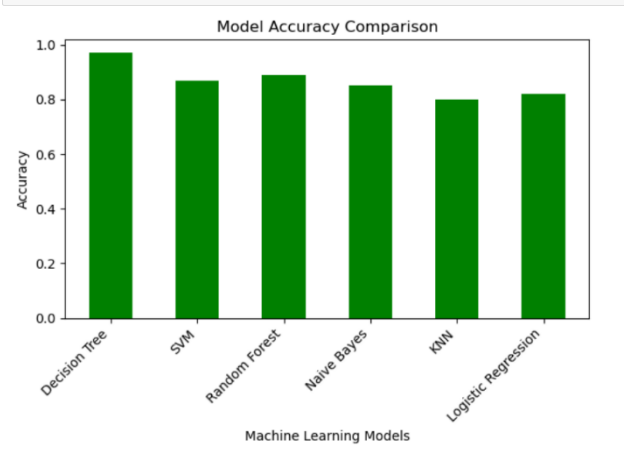
Data mining is a highly application-driven domain and has incorporated many techniques from other domains such as statistics, machine learning, pattern recognition, databases and data warehouse systems, information retrieval, visualization, algorithms, high-performance computing, and many others.

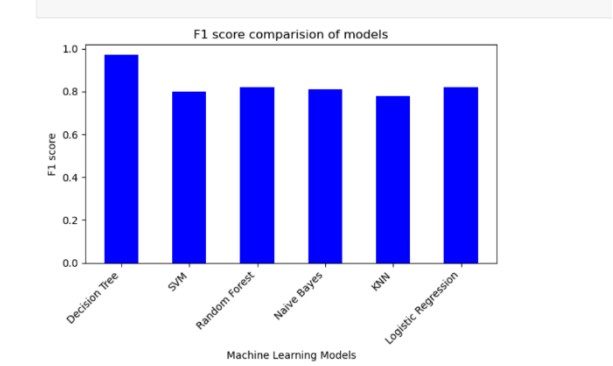
The interdisciplinary nature of data mining research and development contributes significantly to the success of data mining and its extensive applications.

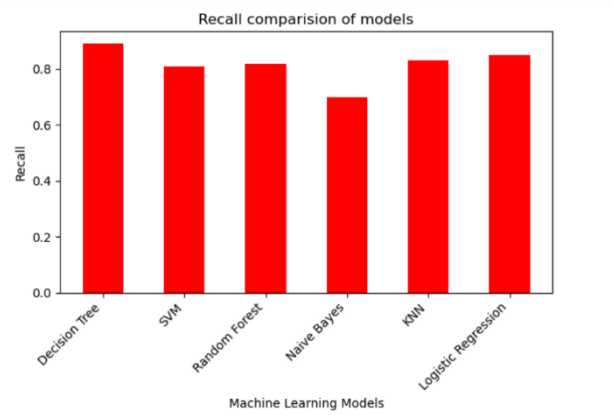


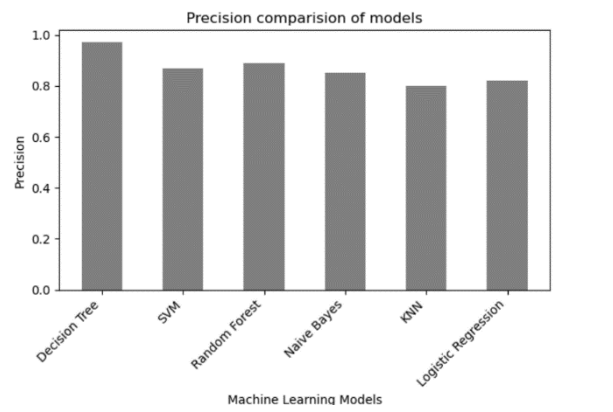
There are many algorithms/ techniques such as random forest, SVM, decision tree, naive bayes, KNN, logistic regression, that can be used to validate and test the model.

Comparing the accuracy, F1-score, Recall and Precision values for different types of algorithms.









And after comparing the accuracy, F1-score, Recall and precision values for all the considered techniques, we can conclude that Decision tree would be the best choice in out case to validate our data.

Decision Tree

A decision tree is a classification scheme which generates a tree and a set of rules, representing the model of different classes, from a given data set. The set of records available for developing classification methods is generally divided into disjoint subsets- a training set and a test set.

Former is used for deriving the classifier, while the latter is used to measure the accuracy of the classifier.

Construction of Decision Tree

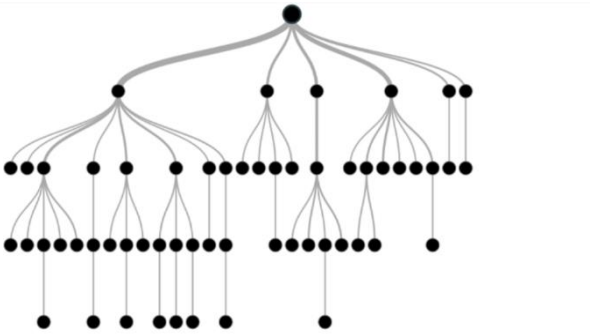
A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data.

In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision Tree Representation

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the sub tree rooted at the new node.



Decision tree will be obtained from the trained and test data from the given input details from the user.

When the user inputs any data to the GUI, the data passes through the decision tree from the root to the leaf considering all the fields such as bachelor's degree, GRE, IELTS, TOEFL, Duolingo, Research paper etc. and the conclusion or output is decided by the final leaf node.

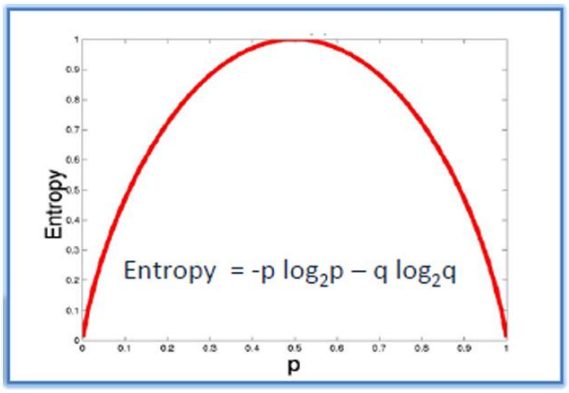
Entropy

Entropy H(S) is a measure of the amount of uncertainty in the (data) set S (i.e., entropy characterizes the (data) set S).

Where,

S - The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm) X- Set of classes in the set.
p(x) - The proportion of the number of elements in class x to the number of elements in set S
When H(S)=0, the set is perfectly classified (i.e. all elements in S are of the same class).

In ID3, entropy is calculated for each remaining attribute. The attribute with the smallest entropy is used to split the set S on this iteration. The higher the entropy, the higher the potential to improve the classification here.



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Information Gain

Information gain IG(A) is the measure of the difference in entropy from before to after the set S is split on an attribute A.

In other words, how much uncertainty in S was reduced after splitting set S on attribute A.

Where,

H(S) – Entropy of set S

T – The subsets created from splitting set S by attribute A such that

S=Ut->T(t)

p(t) – The proportion of the number of elements in t to the number of elements in set S
H(t) – Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the largest information gain is used to split the set on this iteration.

Step 1: Calculate the entropy of the target
Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain or decrease in entropy.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Gini Index

Gini index Gini is the measure of diversity. There are several ways of calculating the index of diversity for a set of records. With all of them, a high index of diversity indicates that the set contains an even distribution of classes, while a low index means that members of a single class. The best splitter is the one that decreases the diversity of the record sets by the greatest amount, in other words, we want to maximize diversity (before split) – (diversity (left child) + diversity (right child))

The Gini index is a diversity measure of economics. It can also be used to evaluate the goodness of a split.

If a data set T contains n classes, then Gini(T) is defined as

Gini(T)=1-$\sum pi^2$

where pj is the relative frequency of class j in T. If the split divides T into T1 and T2, then the index of divided data is given as

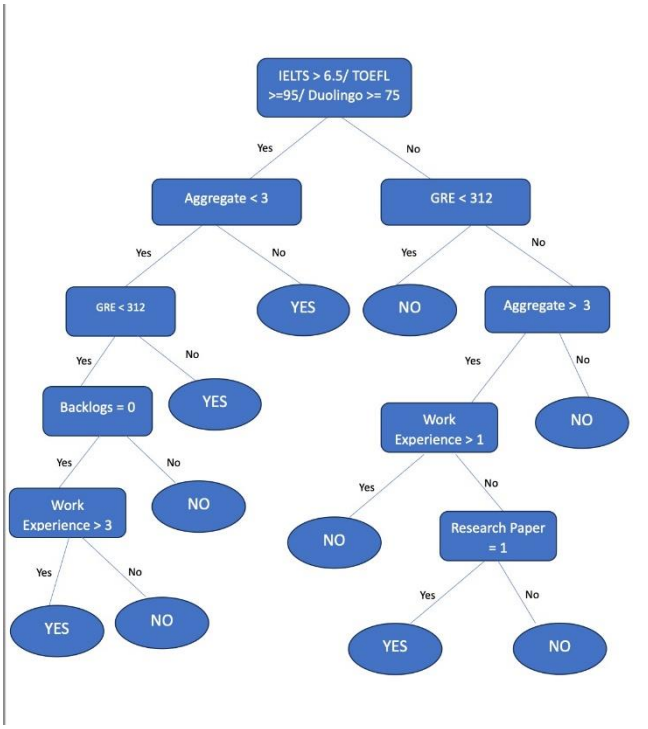
Ginisplit(T)=(n1/n) * Gini(T1) +(n2/n) *Gini(T2)

Proposed Approach

Below is the flowchart of execution of the program from the beginning step by step along with error that might occur and when the successful execution is done.

The user logs in to the portal and the user has to input values for various fields such as name, bachelors aggregate, GRE, IELTS, Duolingo, Research paper, LORs, SOP and so on. Based on all these input values

the model will predict whether the student might get admitted to the college or receive a rejection. For making this decision, the model uses the trained data set.



If the user inputs some incorrect or invalid data, the error message is shown.

Tools and Technologies Used

Sci-Kit learn

The scikit-learn project started as scikitslearn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately developed and distributed third-party extension to SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

Python

Python has been used in artificial intelligence projects as a scripting language with modular architecture, simple syntax, and rich text processing tools, Python is often used for natural language processing.

Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Its package management system is conda.

The Packages used are

Tkinter
Tree
Csv (Comma Separated Values)
Math
Numpy

The Main Functions used in this project are

clf.fit()

Parameters:

X: {array-like, sparse matrix}, shape (n_samples, n_features)

Training vectors, where n_samples is the number of samples and n_features is the number of features. For kernel=" precomputed", the expected shape of X is (n_samples, n_samples).

y: array-like, shape (n_samples,)

Target values (class labels in classification, real numbers in regression)

sample weight: array-like, shape (n_samples,)

Per-sample weights. Rescale C per sample. Higher weights force the classifier to put more emphasis on these points.

Returns: self: object

clf.predict()

Perform classification on samples in X.

For a one-class model, +1 or -1 is returned.

Parameters:

X : {array-like, sparse matrix}, shape (n_samples, n_features)

For kernel=" precomputed", the expected shape of X is [n_samples_test, n_samples_train]

Returns:
y_pred : array, shape (n_samples,)
Class labels for samples in X.

DecisionTreeClassifier()
class sklearn.tree.DecisionTreeClassifier (criterion='gini',max_depth=None, min_weight_fraction_leaf=0.0, max_leaf_nodes=None, min_samples_split=2, max_features=None, min_impurity_decrease=0.0, splitter='best',min_samples_leaf=1, random_state=None, min_impurity_split=None, class_weight=None, presort=False)[decision tree classifier.

Parameters:

criterion: string, optional (default="gini") The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter: string, optional (default="best") The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : int or Nne, optional (default=None) The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : int, float, optional (default=2) The minimum number of samples required to split an internal node: If int, then consider min_samples_split as the minimum number.

If float, then min_samples_split is a fraction, and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_weight_fraction_leaf : float, optional (default=0). The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_features : int, float, string or None, optional (default=None)
The number of features to consider when looking for the best split: If int, then consider max_features features at each split. If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.

If "auto", then max_features=sqrt(n_features). If "sqrt", then max_features=sqrt(n_features). If "log2", then max_features=log2(n_features). If None, then max_features=n_features.

random_state: int, RandomState instance or None, optional (default=None)

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np. random.

max_leaf_nodes: int or None, optional (default=None) Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes. min_impurity_decrease: float, optional (default=0.)

Result and Discussion

This chapter includes the snapshots along with the result of the project along with the datasets used in the project.

Data Set Information

Training data considered:

dataset

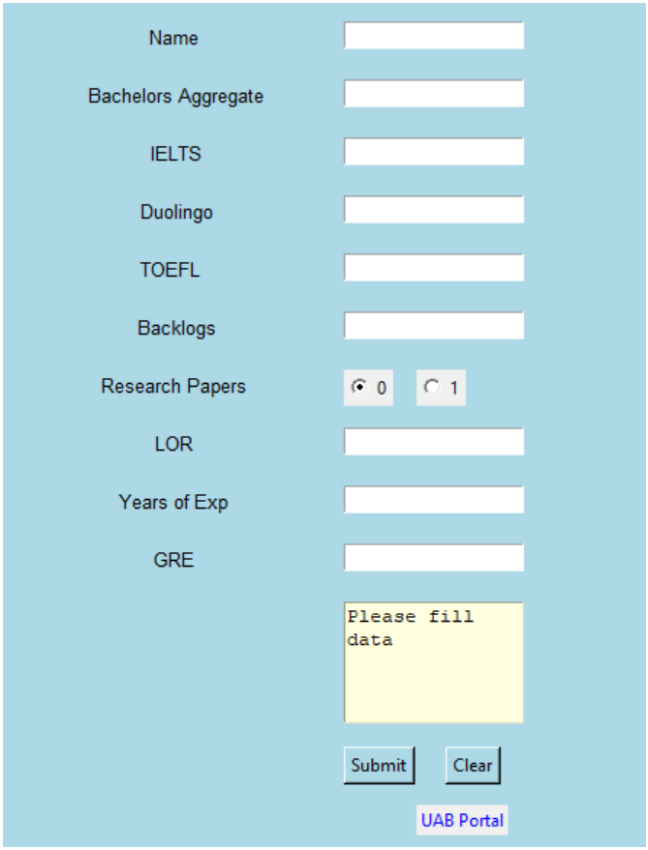
name	Bachelors Aggregate	IELTS	TOEFL	Duolingo	Backlogs	Research Papers	LOR	Work experience in years	GRE	Prediction
Allison Garcia	9.5	7.0	0	89	0	0	4	0	322	Yes
Robert Wall	6.9	8.0	0	93	0	1	4	4	319	Yes
Jerry Jones	7.5	8.0	108	96	1	1	2	5	298	Yes
Dawn Wright	6.5	8.5	97	78	0	0	1	1	320	No
David Webb	9.0	0	0	72	0	0	4	2	302	No
Terry Kaufman	8.5	7.5	109	0	1	0	4	2	298	Yes
Jacob Torres	6.5	0	0	85	0	0	3	3	299	No
Maria Bennett	8.5	7.0	103	77	0	1	2	4	290	Yes
Cody Grimes	9.0	7.0	80	86	1	0	3	5	335	Yes
Brittany Reyes	8.5	0	0	0	0	0	2	2	292	No
Kyle Morgan	9.5	7.0	0	91	0	0	4	0	326	Yes
Philip Elliott	9.5	0	115	0	1	1	3	4	297	Yes
Aaron Turner	8.5	0	107	94	0	1	4	1	309	Yes
Laurie Moran	7.5	0	86	0	0	1	2	4	311	Yes
Andrew Kelly	7.5	7.0	108	100	0	1	4	4	327	Yes
Daniel Hamilton	9.5	5.0	0	78	0	1	4	0	309	No
Paula King	6.0	8.0	94	80	0	0	1	0	310	Yes
Laura Willis	6.5	5.5	0	0	1	1	4	5	321	No
Carrie Martinez	9.5	7.5	107	100	1	1	3	4	322	Yes
Sharon Williams	7.5	7.0	113	98	0	0	1	4	340	Yes
Alexis Chen	7.0	8.0	107	83	0	0	2	0	291	No

For numerical data, if the value is not given, mean of that attribute is considered as the value. For categorial data, if the value is not given, mode of the attribute is considered as the value.

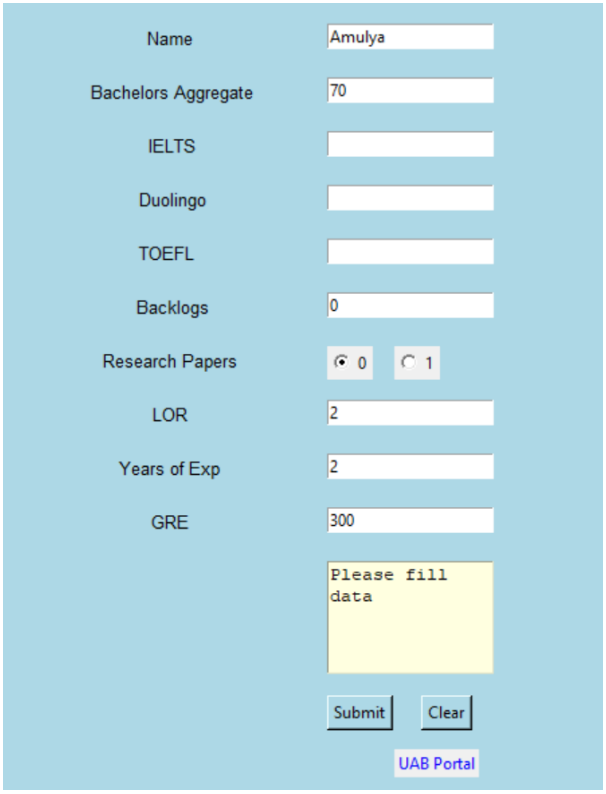
Snapshots of the project GUI

Below are few sample cases showing the project outputs for various types of details provided.

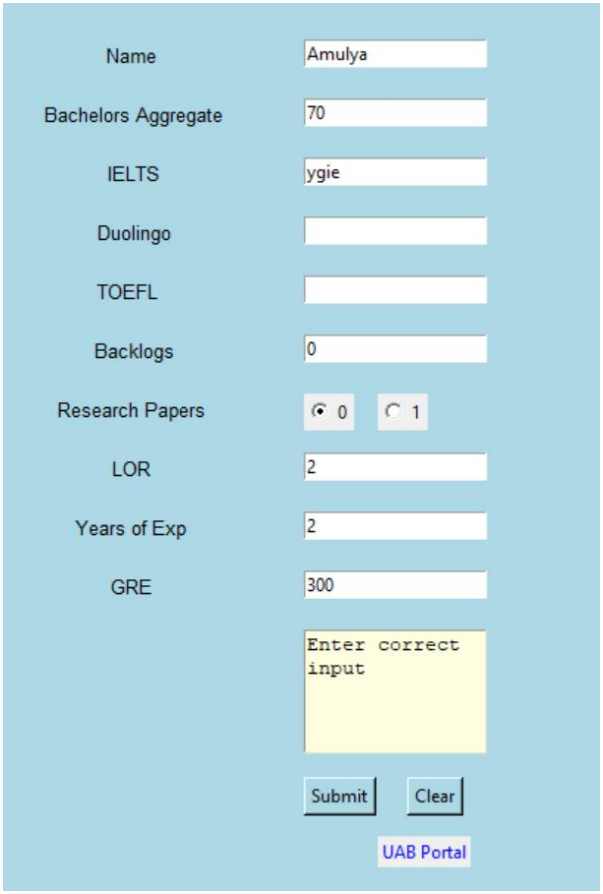
GUI with all clear inputs (no inputs provided)



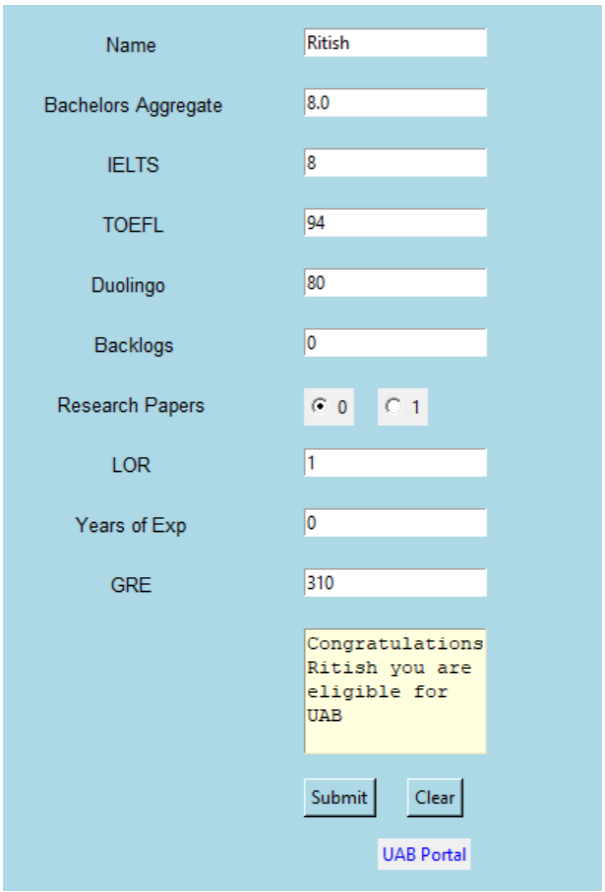
GUI with some missing data (English test scores mandatory



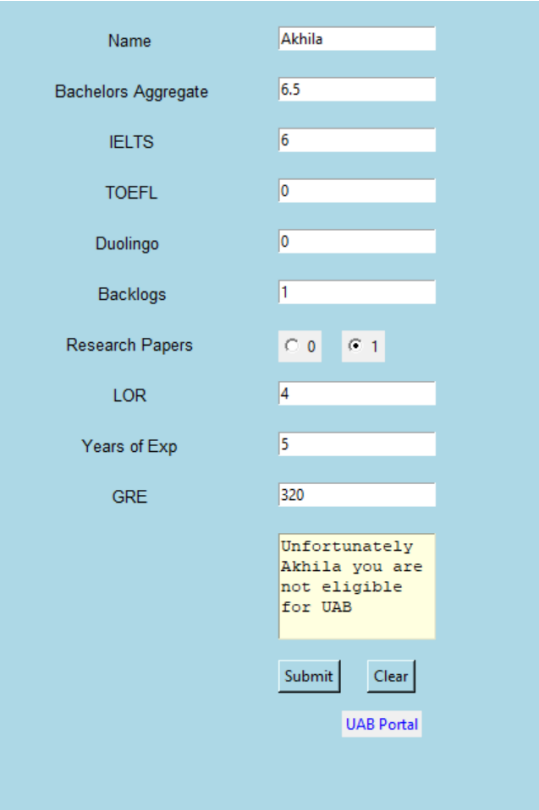
GUI with some invalid input (IELTS in this case)



GUI showing result is Student is admitted



GUI showing result is Student is Rejected



In the similar way, the project will be useful to check with various types of data and below are the few other testcases

Name	Bachelors	IELTS	TOEFL	Duolingo	Backlogs	Research	LOR	Work exp	GRE	Prediction
Akhila K	3.4	0	74	116	1	1	3	3	339	No
Akhila Ko	2.4	8	80	94	0	0	1	0	310	Yes
Arvind	3.2	8	80	94	0	0	1	0	310	Yes
Ritish	3.2	0	80	94	0	0	1	0	310	No
Ammu	3.2	7	80	94	0	0	1	0	310	No
Rick	3.2	7.5	80	94	0	0	1	0	310	Yes
Amulya	3.2	7	0	100	1	0	3	2	310	Yes
Mike	2.4	7.5	0	110	1	1	3	5	312	Yes
Ryan	2.6	0	120	110	2	0	3	3	321	Yes
Sam	2.8	0	120	0	1	0	3	0	300	Yes

Conclusion

In conclusion, using decision tree classification to predict college admissions provides a useful prediction application. Decision trees can be used to model admission outcomes by utilizing attributes like bachelors aggregate, GRE, TOEFL, IELTS, Duolingo, LOR's, Research papers and Work experience among others. But it's important to understand that college admissions are dynamic processes with changing standards and unique factors. Decision trees are useful algorithms, which is helpful for forecasting college admissions.

Future Scope

The future scope of the project uses different input for decision tree classification to provide a college admission. It may include vast variety of attributes for the selection. Students may also check the eligibility of institution using this application. Scalability features ensure adaptability to evolve admission criteria while visualization tools such as decision trees enhance interpretability. A better user interface takes data and calculate the model evaluation metrics including accuracy and precision and ensure robust performance, will emphasis more on data security and privacy compliance and get a continuous feedback for more improvements in future to forecast college admissions.

This project helps to identify whether the students can get a direct entry admission, INTO admission or pathway admission.

References

Hendrik Blockeel and Jan Struyf (2002), "Efficient Algorithms for Decision Tree Cross-validation", Journal of Machine Learning Research, 621-650.

Shikha Chourasia (2013), "Improved Methods of ID3 Decision Tree Classification", International Journal of Scientific and Research Publications, 2250-3153.

Amany Abdelhalim and Issa Traore (2009), "A New Method For Learning Decision Trees from Rules", International Conference on Machine Learning and Applications, 108-121.

https://scikit-learn.org/stable/modules/tree.html

https://www.geeksforgeeks.org/decision-tree-implementation-python/

We declare that we have completed this assignment completely and entirely on our own, without any consultation with others. We have read the UAB Academic Honor Code and understand that any breach of the Honor Code may result in severe penalties.

We also declare that the following percentage distribution ***faithfully*** represents individual group members' contributions to the completion of the assignment

Name	Overall Contribution (%)	Major work items completed by me	Signature or initials	Date
Ritish Nedunoori	20	Model generation, Report and PPT	RN	12-03-2023
Amulya Sai Boppana	20	Dataset creation, Report and PPT	ASB	12-03-2023
Aravind Nune	20	Model generation, Report and PPT	AN	12-03-2023
Akhila Kodudula	20	Accuracy and prediction calculation, Report and PPT	AK	12-03-2023
Akhila Kankanala	20	Accuracy and prediction calculation, Report and PPT	AK	12-03-2023