# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

***Abstract*— This study employs machine learning methods to assess the likelihood of students being admitted to postgraduate programs. Initially, a linear regression model is trained using a Kaggle dataset. Principal Component Analysis (PCA) is then utilized to streamline the dataset by reducing its dimensions while preserving most of its information. Three distinct classification algorithms - Gaussian Naive Bayes, K-Nearest Neighbors (K-NN), and Decision Trees - are applied to both the original and PCA-transformed datasets. Each algorithm undergoes optimization through hyperparameter tuning to enhance performance metrics like F1 score, confusion matrix, and ROC curves. Following this, the models' performance is evaluated, with logistic regression (LR) exhibiting superior performance compared to other models available in the PyCaret library. Decision boundaries for each model are depicted to illustrate their fitting on the dataset. Moreover, the interpretability of the models is explored using Shapley values, an explainable AI technique, to discern the factors influencing admission decisions. In conclusion, the study demonstrates the effectiveness of these algorithms in determining students' eligibility for university admission.***

***Index Terms*—Principal component analysis, binary classification, gaussian naïve bayes, K-nearest neighbor, decision tree.**

## I. INTRODUCTION

In today's competitive employment landscape, merely possessing an undergraduate degree is no longer sufficient. There has been a noticeable trend in recent years towards heightened job requirements, leading to an unprecedented demand for high-quality higher education. Consequently, many students are choosing to pursue education in foreign countries. However, enrolling in these overseas institutions requires meeting certain academic standards. Despite the abundance of colleges and universities, students often find themselves uncertain until the eleventh hour about the acceptance of their applications due to the lack of definitive documentation outlining prerequisites. In this context, leveraging machine learning (ML) and data mining methods can play a crucial role in minimizing false positive and false negative decisions in the admissions process. In recent years, machine learning (ML) approaches have become integral in developing systems capable of classifying the qualifications required for university admission. Students can leverage these systems to gauge their likelihood of acceptance into their desired colleges. By evaluating schools based on these criteria, students can streamline their options effortlessly. This report was initiated by employing Principal Component Analysis (PCA) to reduce the dimensionality of the dataset concerning applicants' admission probabilities. Subsequently, three renowned classification algorithms—Gaussian Naive Bayes, K-Nearest Neighbor Analysis (K-NN), and Decision Tree—are applied to both the original and PCA-transformed datasets. The objective is to assess whether applicants have a viable chance of securing admission to college. To gain insights into the classification models, explainable AI techniques such as Shapley values are utilized. It's important to note that the reported results of the classification algorithms in this study are based on the outcomes obtained after employing PCA. Additionally, for enhanced readability, this report presents the outcomes derived from the modified dataset. The classification results of the original dataset are available for reference in the Google Colab notebook.

Here's a revised breakdown of the remaining sections of the report: Section II introduces PCA with a concise overview, followed by Section III which provides insights into three classification algorithms. In Section IV, we delve into the description of the admission probability dataset, highlighting its key features. Section V presents a comprehensive discussion of the PCA results, shedding light on the insights gleaned from dimensionality reduction. Moving forward, Section VI offers a detailed analysis of the classification results, providing valuable interpretations and comparisons. Section VII delves into the intriguing world of explainable AI, focusing on the Shapley values and their significance in understanding model decisions. Finally, Section VIII encapsulates the findings and insights of the study, drawing a conclusive summary.

## II. PRINCIPAL COMPONENT ANALYSIS

Most of the real-world datasets contain high dimensionality. Processing and storing of these types of datasets are highly expensive. PCA helps to reduce the dimensionality of high-dimensional data sets into smaller ones by identifying the most important features, or principal components, that explain most of the variance in the original data. PCA is a widely used tool and technique for data analysis and visualization. It allows for the identification of the most vital features in the data and can help users gain insights into the underlying structure of complex data sets. PCA is a technique that can be utilized to reduce the dimensionality of data, identify key trends and patterns, find outliers, and improve the effectiveness of machine learning algorithms.

### *A. PCA algorithm*

Principal component analysis can be applied to a data matrix with N number of samples in rows and P number of variables in columns in the mentioned phases shown below.

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

1) **Standardization**: The primary objective of this stage is to standardize the initial variable to ensure that all of the variables contribute an equivalent amount to the study. First, we will need to determine the mean vector $\bar{x}$ for each column in the data collection. The mean vector is a p-dimensional vector, and its equation may be written as follows:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

After this, the data are normalized by taking the mean of each column and subtracting it from each individual item in the data matrix. The whole centered data matrix, denoted by the letter Y, may be stated as follows:

$$Y = H\,X$$

where $H$ represents the centering matrix.

2) **Covariance matrix computation**: Finding out how the different variables are related to one another is the purpose of this stage. There are situations when the variables are so tightly connected to one another that they include information that is already known. The covariance matrix is constructed to locate these relationships. The following equations are used to calculate the p × p covariance matrix:

$$S = \frac{1}{n-1} Y^T y$$

3) **Eigen decomposition**: It is possible to calculate S's eigenvalues and eigenvectors by using the eigen decomposition technique. Eigenvectors are used to describe the direction that each principal component (PC) will take, while eigenvalues are used to represent the amount of variance that will be collected by each PC. The following equation may be used to do the computation necessary to determine eigen decomposition:

$$S = A \Lambda A_T$$

where A means the p × p orthogonal matrix of eigen-vectors and Λ is the diagonal matrix of eigenvalues.

4) **Principal components:** It calculates the converted matrix Z, which has the dimensions n × p. The observations are represented by the rows of Z, while the PCs are represented by the columns of Z. The number of PCs in use is proportional to the size of the data matrix that was first used. The equation for Z may be expressed as follows:

$$Z = YA.$$

## III. MACHINE LEARNING BASE CLASSIFICATION ALGORITHMS

### A. Decision Tree:

A decision tree is a technique of non-parametric supervised learning that creates classification or regression models in the shape of a tree structure. Decision trees may be used to analyze data for both classification and regression. The purpose of this project is to develop a model capable of predicting the value of a target variable by the discovery and application of simple decision rules derived from the characteristics of the data. To categorize, it makes use of an if-then rule set that is both exhaustive and mutually exclusive. The training data are examined one at a time in order, and the rules are subsequently learnt. When a new rule is discovered, the tuples that it governs are taken out of circulation. On the training set, this procedure is repeated until one of the termination conditions has been fulfilled.

### B. K- nearest neighbor (KNN):

K-NN is a supervised classification technique that attempts to train a model by categorizing the samples in accordance to the training examples that are located in the feature space that are the closest to them [7]. The K-NN technique is often referred to as a "lazy learning algorithm" since it only makes approximations of the function in its immediate surroundings and delays all calculations until classification [8]. K-Nearest Neighbors is an example of a lazy learning algorithm because throughout the training phase, it just stores the data and conducts the calculation when it is time to classify the data. The K-Closest Neighbors method is one of the simplest classification algorithms available. In this technique, an item is allocated to the class that is the most frequent among its k numbers of nearest neighbors after being categorized by its neighbors based on a majority vote. K-Nearest Neighbors makes its model of the target function based on all the labelled training cases. To properly categorize a sample, the K-Nearest Neighbors algorithm first chooses the number of neighbors, then computes the Euclidean distance between each pair of neighbors, and then picks the K neighbors that are geographically closest based on this distance. In the last step, the K-Nearest Neighbors algorithm places the newly collected data in the category that already contains the most samples.

### C. Gaussian Naïve Bayes

The term "naive bayes classifier" refers to a family of straightforward probabilistic classifiers that are used in the field of machine learning. These classifiers are created by using Baye's theorem while making strong independent assumptions across the features.

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

The Naive Bayes classifier is often a parametric model. This model operates on the presumption that the existence of a certain characteristic in a class is independent to the presence of any other feature.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using terms associated with the Bayesian theory of probability, the previously mentioned equation may be expressed as,

$$Posterior = \frac{Prior \times Likelihood}{Evidence}$$

## IV. DATA SET DESCRIPTION

Kaggle served as the source for obtaining the admission probability dataset crucial for this project's objectives. This dataset provides insights into the likelihood of a student's admission to an institution, expressed as a percentage. Comprising seven distinct criteria, namely the GRE score, TOEFL score, university ranking, Statement of Purpose (SOP), Letter of Recommendation (LOR), percentage, and research experience, each criterion is backed by 500 submissions. Leveraging these comprehensive criteria, the system aims to accurately assess the student's probability of acceptance into the university.

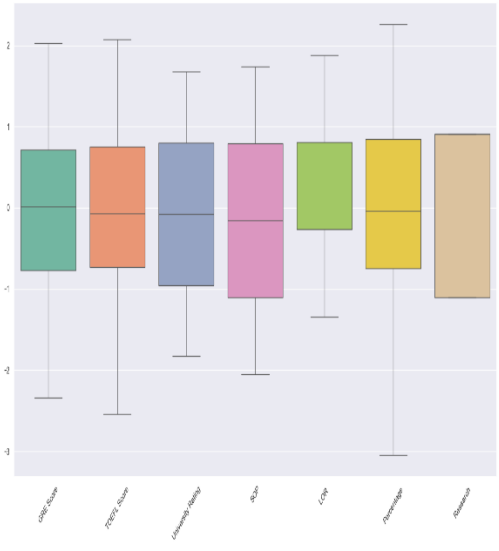

Fig-1: Box Plot

The distributions, central values, and variability of the characteristics were assessed with the use of box and whisker plots and their five number summaries on the dataset. The box plot of the characteristics of the chance of admission dataset may be seen in Fig-1. The image makes it clear that only two of the characteristics, the LOR, and the Percentage, include any outliers.

The correlation matrix for the normalized characteristics of the dataset is shown in Fig. 2. The GRE Score, the TOEFL Score, and the Percentage are the characteristics that have big positive values. This clearly suggests that these three characteristics have a high correlation with one another. The remaining characteristics have a weaker correlation with the features that are already included in the dataset. This finding is supported by the pair plot that may be seen in Figure 3. The highly correlated characteristics have a greater number of cells that increase in number in a regular pattern. On the other hand, the remaining characteristics demonstrate a weaker correlation amongst themselves.

## V. PCA RESULTS

PCA is applied to the dataset containing admission probabilities. There are two main approaches to implement PCA: constructing PCA from scratch using common Python libraries like numpy or utilizing a well-documented PCA library. Both methods are elaborated upon below, with implementation details provided in the Google Colab notebook. Opting for the PCA library affords users greater flexibility, as complex tasks can be achieved with just a single line of code, although the results are comparable across both methods. This paper includes figures and plots derived from the implementation conducted using the PCA library.

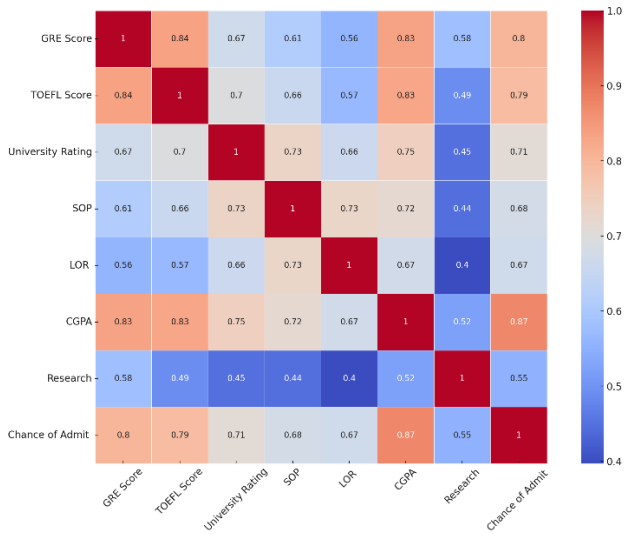


Fig-2: Correlation Matrix

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

By employing the stages of PCA, the original feature set comprising seven items can be condensed into a set of r features, where r is typically smaller than 7. This reduction is facilitated by the eigenvector matrix A, which transforms the original dataset with dimensions n × p. In this transformation, each column of the eigenvector matrix A represents a principal component (PC), akin to a virtual stand-in for a personal computer. Each PC encapsulates a specific amount of data, effectively determining the dimensionality (r) of the reduced dataset.

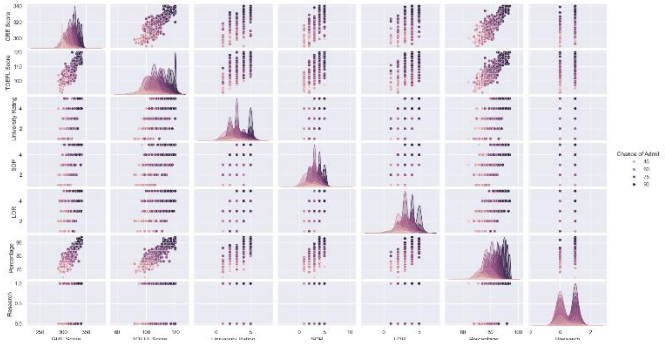

Fig-3: Pairplot

The following is an illustration of the produced eigenvector matrix (A) for the chance of admission dataset:

$$A=\begin{bmatrix} -0.40703207 & -0.29035438 & -0.31804042 & -0.17687917 & -0.00822785 & -0.22456993 & 0.75304975 \\ -0.4040102 & -0.13930849 & -0.43276995 & -0.17603167 & -0.03329763 & 0.72010999 & -0.28182405 \\ -0.38645209 & 0.24304508 & 0.02873674 & 0.56248101 & 0.68624024 & 0.0351194 & 0.04705463 \\ -0.37970846 & 0.36531726 & 0.11106807 & 0.43751573 & -0.71411451 & 0.03243295 & 0.08716203 \\ -0.33624154 & 0.47548901 & 0.48616841 & -0.62819397 & 0.12681732 & 0.08105421 & 0.08492371 \\ -0.42360628 & -0.03964817 & -0.23300861 & -0.14827499 & -0.02106426 & -0.64577544 & -0.57029609 \\ -0.29167838 & -0.68999862 & 0.63854978 & 0.11536276 & -0.0375769 & 0.07161687 & -0.10597288 \end{bmatrix}$$

and the corresponding eigen values are:

$$\lambda=\begin{bmatrix} 4.66230955 \\ 0.75754578 \\ 0.57950129 \\ 0.40515251 \\ 0.27975631 \\ 0.17923877 \\ 0.15052385 \end{bmatrix}$$

Figures 4 and 5 depict the scree plot and the Pareto plot of the principal components (PCs), respectively. These visualizations illustrate the extent to which each principal component contributes to explaining the variation in the dataset. To calculate the proportion of variation explained by the jth PC, the following formula can be utilized:

$j = \frac{\lambda_j}{\Sigma_j^P \lambda_j} \times 100, j = 1,2,, \ldots \ldots, p$ , where $\lambda_j$ represents the eigenvalue and the amount of variance of the j-the PC.

The number of PCs and the amount of variation that can be explained are shown in Figures 4 and 5. It is clear from looking at both figures that the variance of the first two principal components accounts for 86% of the total variance of the original dataset.

The principal components are as follows:

$Z_1$ = -0.40703207$X_1$-0.4040102$X_2$-0.38645209$X_3$ - 0.37970846$X_4$-0.33624154$X_5$-0.42360628$X_6$ - 0.29167838$X_7$

$Z_3$= -0.31804042$X_1$- 0.43276995$X_2$+0.02873674$X_3$+0.11106807$X_4$+ 0.48616841$X_5$-0.23300861$X_6$+0.63854978$X_7$

$Z_2$= -0.29035438$X_1$-0.13930849$X_2$+ 0.24304508$X_3$+0.36531726$X_4$+0.47548901$X_5$- 0.03964817$X_6$-0.68999862$X_7$

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

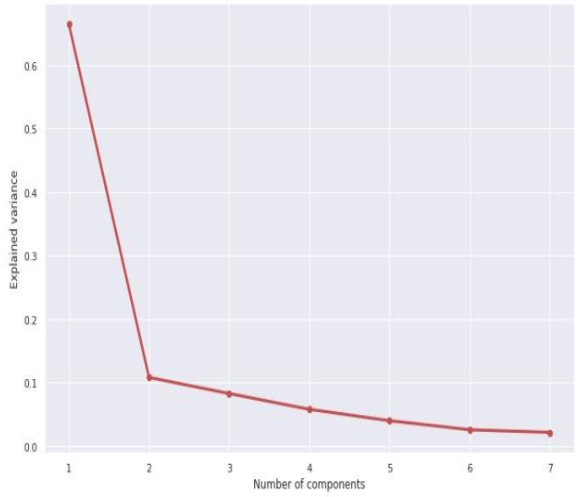


Fig-4: Scree Plot

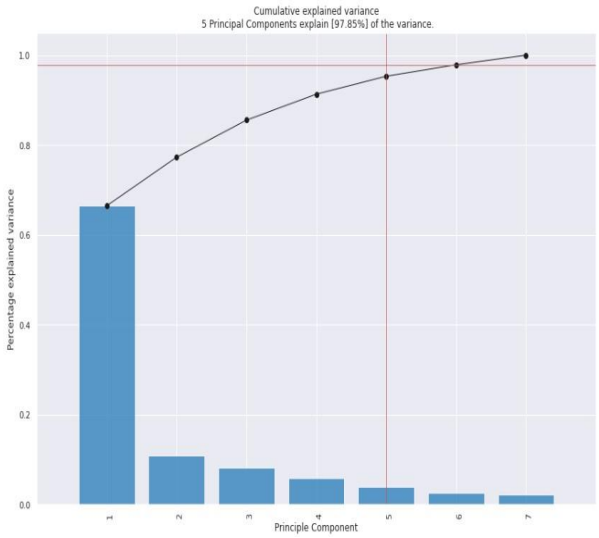


Fig-5: Pairplot

Figure 6 showcases the PC coefficient plot, offering a visual representation of the relative influence of each feature on the first two principal components (PCs). This graphical depiction corroborates the earlier PC calculations, highlighting that GRE Score, TOEFL Score, and Percentage exert the most significant impact on the first PC.

Another visualization, the Biplot in Figure 7, presents an alternative perspective on the first two PCs. Here, the axes represent these principal components, while vectors representing the rows of the eigenvector matrix are displayed below. Each observation in the dataset is depicted as a dot on the plot. Notably, the angles formed by the vectors representing GRE Score, TOEFL Score, and Percentage with the first PC are minimal, suggesting a substantial impact on this component. Conversely, the angles with the second PC are more pronounced, indicating a lesser influence. This observation aligns with the insights gleaned from the PC coefficient plot in Figure 6.

Moreover, the vectors representing the remaining features exhibit a contrasting pattern. Initially forming larger angles with the first PC, they subsequently show smaller angles with the second PC. This dynamic suggests a stronger association with the second PC, underscoring their varying degrees of influence across the principal components.

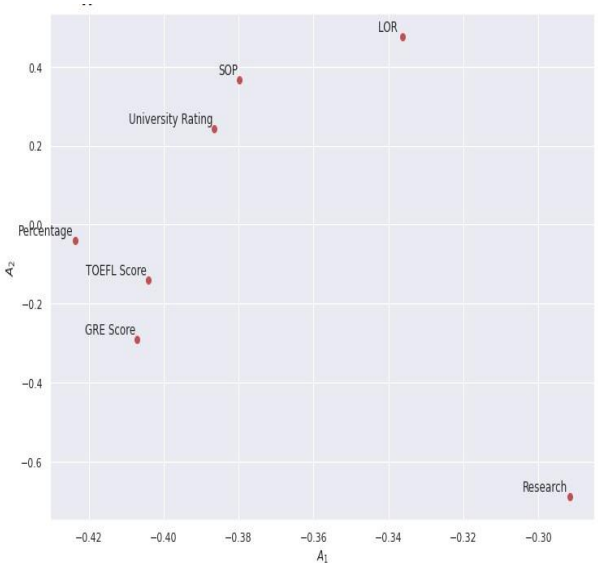


Fig-6: PC Coefficient Plot

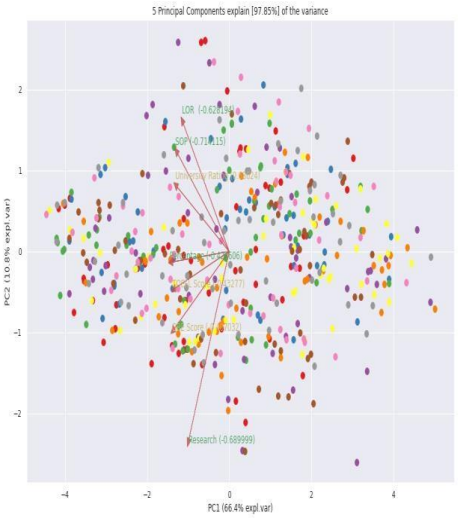


Fig-7: Biplot

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

## VI. CLASSIFICATION RESULTS

In this section of the report, we delve into the evaluation of the performance of three renowned classification algorithms using a dataset containing information on admission probabilities. These classification methods are applied to both the original dataset and the PCA-transformed dataset, utilizing three PCA components. This approach allows us to discern the influence of PCA on the admission probability dataset, shedding light on its impact. The categorization process is facilitated using Python's PyCaret package, streamlining the implementation. Subsequently, the initial dataset is partitioned to derive train and test sets, ensuring robust evaluation of the classification algorithms.

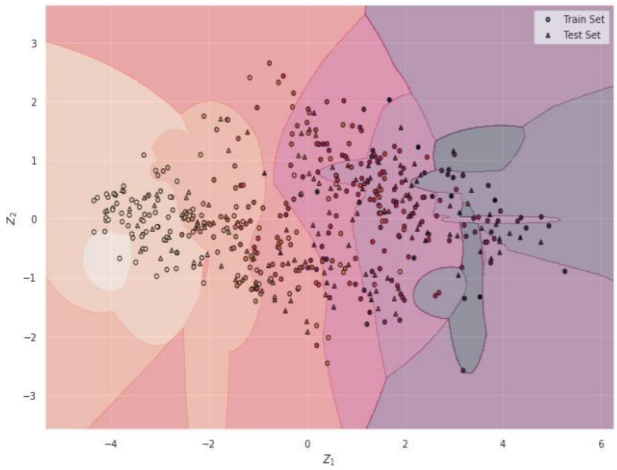


Fig-8: Gaussian Naive Bayes

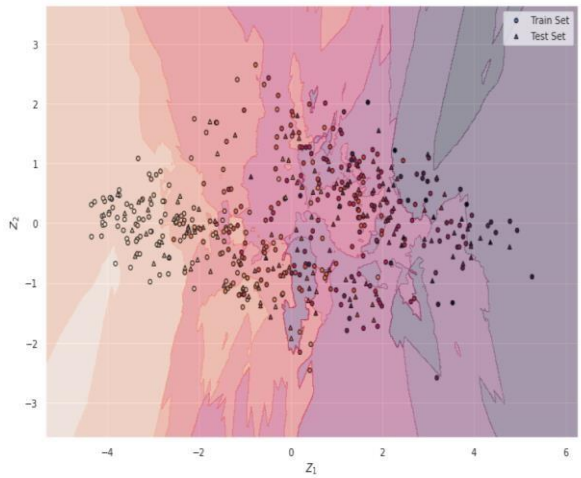


Fig-9: K- nearest neighbor

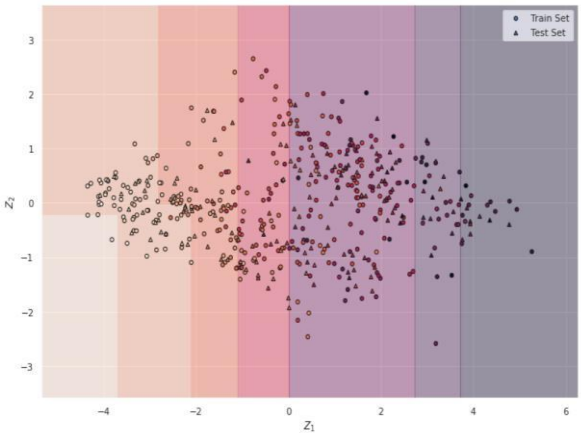


Fig-10: Decision Tree

In the subsequent stages of the experiment, these three algorithms serve as the foundation for our evaluations. We employ them to train, fine-tune, and evaluate both the original dataset and the transformed dataset. The Google Colab notebook contains detailed results from both experiments. However, this report concentrates solely on the insights gleaned from using PCA (transformed dataset).

Hyperparameter tuning is a crucial step in enhancing a model's performance. Leveraging PyCaret for this purpose entails three key steps: model creation, hyperparameter tuning, and performance analysis. Initially, a classification model is constructed based on the selected algorithm. Subsequently, the tune_model() method is employed to fine-tune the model with optimal hyperparameters. This function automatically evaluates the model using stratified K-fold cross-validation, tuning it with effective hyperparameters from a predefined search space. PyCaret defaults to using a 10-fold stratified K-fold validation across the three different techniques.

Based on the data presented above, it becomes evident that the metrics of the tuned KNN model surpass those of the basic model (prior to hyperparameter tuning).

Figures 8, 9, and 10 present visual representations of the decision boundaries generated by the model based on the transformed data. A decision boundary is a hyperplane that separates data points into distinct classes, marking the transition from one class to another. In these figures, the x-axis represents the first principal component (PC), while the y-axis represents the second principal component. Each dot on the graph represents an observation from various courses, illustrating the diverse decision boundaries produced by the algorithms and their variations. It's evident from the graphic that the KNN algorithm outperforms Gaussian Naive Bayes and Decision Tree in generating decision boundaries.

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

Given that the admission probability dataset involves binary classification, precision and recall metrics are employed to evaluate the performance of each class separately. Precision measures the proportion of relevant instances among all retrieved instances, while recall represents the fraction of retrieved instances among all relevant instances. These metrics provide insight into the effectiveness of classification. Figures 11, 12, and 13 display the confusion matrices generated by the three methods when applied to the transformed dataset. A confusion matrix summarizes the mix of expected and actual class occurrences, depicting accurate and inaccurate predictions for each class. You can refer to the notebook hosted on Google Colab for confusion matrices related to the original dataset. In these figures, the horizontal axis signifies the class prediction, while the vertical axis represents the actual label.

The F1-score is an additional measurement that is used in the performance assessment. The F1-score is a measure that takes the harmonic mean of a classifier's accuracy and recall values to provide a single score that combines the two. It is an excellent statistic to use when comparing the results obtained by the different classifiers. A good illustration of this would be the fact that classifier A has a greater recall whereas classifier B has a better precision. In this scenario, the F1-score is useful in determining which of two classifiers is superior. The following is one way to define the function of the F1score:

$$\text{F1-Score} = 2 \text{ X } \frac{Precision\ X\ Recall}{Precision+Recall}$$

The F1-Score of KNN has been improved after applying the PCA. All these results suggest possible benefits from using PCA and tuning hyperparameters.

## VII. BAR CHART

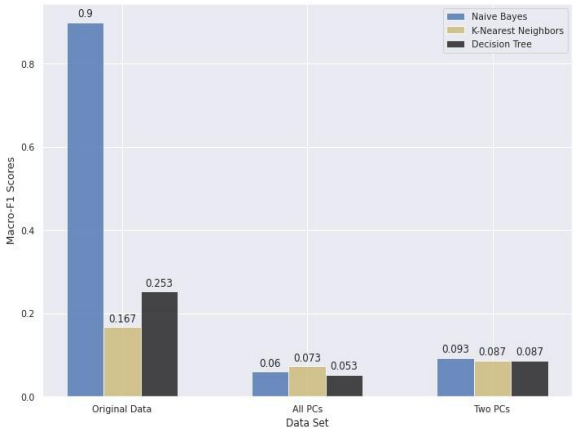


Fig-17: Bar Chart

From, the above chart we can say that after using the PCA, K-nearest neighbor algorithm is performing better than the Gaussian Naïve Bayes and the decision tree.

## VIII. CONCLUSION

In conclusion, we applied principal component analysis (PCA) alongside three widely used classification algorithms to the dataset representing the probability of admission. This dataset, referred to as "chance of admission," predicts students' likelihood of acceptance into a specific university. Initially, PCA was conducted on the primary dataset, revealing that the top three principal components (PCs) capture 85.6% of the total data variation. Subsequent rigorous testing on these three PCs, accompanied by comprehensive visualizations, provided insights from various perspectives.

Following the PCA analysis, we employed three classification methods - Gaussian Naive Bayes, K-Nearest Neighbor, and Decision Tree - on both the original and transformed datasets. Hyperparameters for each algorithm were fine-tuned, and their performance was evaluated using metrics such as ROC curves, F1-scores, and confusion matrices. Notably, after hyperparameter adjustment, the performance of all algorithms demonstrated significant improvement.

Ultimately, our analysis revealed that among the three algorithms, K-Nearest Neighbor emerged as the most effective. However, it's essential to note that all three systems exhibited the capability to accurately forecast whether an applicant would be accepted by the target university.

# Principal Component Analysis and Prediction of Admission using Machine Learning

**Vishnu Sharan Tupurani – 40271136**
**GitHub Link: https://github.com/VishnuSharan-T/INSE6220**

## IX. REFERENCES

Bibodi, J., Vadodaria, A., Rawat, A. and Patel, J. (n.d.). Admission Prediction System Using Machine Learning.

Eberle, W., Simpson, E., Talbert, D., Roberts, L. and Pope, A. (n.d.). Using Machine Learning and Predictive Modeling to Assess Admission Policies and Standards.

Mane, R. V. (2016). Predicting Student Admission decisions by Association Rule
Mining with Pattern Growth Approach, pp. 202–207

Mishra, S. and Sahoo, S. (2016). A Quality Based Automated Admission System for Educational Domain, pp. 221–223.

Waters, A. and Miikkulainen, R. (2013). G RADE : Machine