PRINCIPAL COMPONENT ANALYSIS AND MACHINE LEARNING TO PREDICT UNIVERSITY RATING

GUSTAVO DAVID ORNA VILLALTA 40222999

GitHub Link: https://github.com/gusorna/INSE_6210_PROJECT/blob/main/PROJECT6210_PyCaret_University.ipynb

***Abstract*—For this project, Principal Component Analysis (PCA) is applied on a dataset of required elements to apply for admission in universities to predict the university rating based on the requirements for admission. Next, three multiclass classification models are selected to be applied to the dataset and to the altered dataset after the PCA analysis. In this case the models selected are the Decision Tree Classifier (DTC), the K-Nearest Neighbor Classifier (K-NN) and the Logistic Regression (LR). Then, all the models are tuned with hyperparameters to achieve better performance metrics. Next, the models are shown with their decision boundaries to show how each module fits on the dataset. Finally, an experiment is performed to make an interpretation of the model employing the explainable AI (Artificial Intelligence) with Shapley Values.**

Keywords—Principal Component Analysis, multiclass classifications, decision tree, K-nearest neighbor, logistic regression, hyperparameters

I. INTRODUCTION

Admission for universities has always been a struggle for students applying for them. Students need to take into consideration many variables that universities evaluate to determine eligibility for admission. It is very common that the universities that are considered "better" are the ones demanding more from applicants to just be considered for admission. An example of this would be the University of Chicago having an acceptance rate of less than 6% [1]. This is a good reason for a student to check many times the requirements needed to apply for a specific university. A way to figure out this issue is to determine the quality of each university to prepare the necessary documents and grades needed and elaborate a list of possible universities to apply. There are already some criteria that are evaluated to establish a university ranking like for example the percentage of alumni donating money to their former university or the graduation rate average [2]. Some ranking models already exist to establish this information and make it easier for students to choose a university to apply like the US News and World Report ranking model that focuses more on the performance of a student during the time at their university [3]. Seeing how complex is the process of determining the rating of a university depending on so many variables, it is proposed in this paper a different approach by trying to predict a university rating based on the criteria that is normally requested to be granted admission at a university.

Lately, the use of machine learning algorithms for prediction on a variety of subjects has increased due to the reliability of these prediction methods and their way of saving time to perform this task. Nowadays there are pre-established packages with these algorithms ready to use so a person without a deep knowledge of these algorithms can use them to facilitate their work.

For this project, the first step is to apply Principal Component Analysis (PCA) on the dataset of requirements for admission at universities to extract the most relevant information and reduce the dimension of the dataset. Then, the three classification models chosen, the Decision Tree Classifier (DTC), the K-nearest neighbor (KNN) and the logistic regression (LR) are applied to the dataset after PCA was executed and to the original dataset. The reason to apply these multiclassification models is to find out the rating of the universities between the 5 levels specified: (1) =>Poor, (2) => Fair, (3) =>Satisfactory, (4) => Good, (5) => Superior. Finally, to understand the classification models, an experiment is made with an explainable AI with Shapley Values. All the results from the classification methods are from the dataset after it was applied the PCA methodology. All the results, including the ones from the original dataset can be found on the Google Colab notebook from the link provided.

The report will follow the structured proposed of: Section II – description of the PCA methodology with its algorithm, Section III – description of the three chosen classification models, Section IV – description of the dataset of variables requested for admission, Section V – the PCA applied and discussion of results, Section VI – analysis of the results from the three classification methods, Section VII – discussion of the results from the interpretation of the AI Shapley Values ,and for Section VIII – conclusion explained from the results obtained.

II. PRINCIPAL COMPONENT ANALYSIS

Datasets for this type of project often have the problem of dimensionality as they tend to be very large as they usually need a lot of information to make predictions or decisions on their own. Part of the problem is the time and resources needed to review these datasets as the information can be very extensive and complex. PCA is then used as a solution for this problem as it helps to reduce the dimensions of a dataset by extracting the main information from them and converting the variables into a new set of variables with less information than the originals. PCA is a multivariate technique that analyzes a data table and extracts the important information from the table to represent it as a set of new variables called principal components [4].

A. PCA algorithm

To apply the PCA algorithm, first it is needed to define a matrix X which would contain the data for the analysis. Then it would require following the next steps [5]:

1) **Standarization:** In this step the main concern is to standarize all the values of the variables from the data table to have all the values under the same scale. To do this is necessary to calculate the mean ($\bar{x}$) for all the columns in the dataset, generating a mean vector can be calculated by:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} xi \quad (1)$$

The data is now standardized by subtracting the mean vector, the mean of each column from each value from the dataset. Then the centered data matrix Y can be calculated with the following equation:

$$Y = H\,X \quad (2)$$

Where H is the matrix that centers X matrix of the data to obtain Y matrix.

2) **Covariance matrix:** In this step is computed the covariance matrix of the centered data matrix to determine the relationship between the variables in the dataset. The covariance matix is a *p x p* dimension matrix and is determined by the following equation:

$$S = \frac{1}{n-1} Y'Y \quad (3)$$

3) **Eigen-decomposition:** After obtaining the covariance matrix S, eigen-decomposition is used to compute the eigenevectors and eigenvalues. Where the eigenvectos indicate the direction of the principal components (PC) coefficients and the eigenvalues indicate the amount of variance represented in each principal component. This shows the importance of each principal component. These eigenvectors and eigenvalues can be computed using the following equation:

$$S = A\,\Lambda\,A' \quad (4)$$

Where A is the *p x p* orthogonal matrix whose columns are the eigenvectors of S and Λ is a p x p diagonal matrix whose elements are the eigenvalues of S.

4) **Principal component scores:** In this step the matrix Z of size n x p is computed as its columns contain the PC scores and the rows the observations. The number of PCs should be equal to the dimension of the original data. Important to be noted that the first PC score contains as much variation from the original data as possible. The matrix Z is calculated according to the next equation:

$$Z = Y\,A \quad (5)$$

III. Multiclass Classification models

A. *Decision Tree Classifier (DT)*

The decision tree classifier is one of the most used models for machine learning purposes because it possesses reasonable accuracy and is a very cheap model to compute [6]. It helps to solve classification problems and has a structure of a tree with internal nodes that manifest the features of the dataset used, branches that indicate the decision rules inside the model and leaf nodes that express the outcome from where it cannot divide any further [7]. Basically, how it works is that each internal node is a feature from the dataset and if the data falls in this category it becomes a leaf node as the outcome does not divide into another subtree. If the data does not fall for this category, it divides into another subtree into another decision node where the process will continue to happen until the last leaf node is achieved. In other words, when all the data has been completely classified.

B. *K-Nearest Neighbor (KNN)*

This algorithm of classification is very well known for its simplicity of use and as it requires no process of the training data before performing this method. It classifies the samples from a dataset by the most voted label within the k nearest neighbors in the dataset [8]. The performance of the model depends directly on the parameter k, this being the most important parameter on a knn classification algorithm. After it has being set, the prediction can be made depending on the class labels of these k nearest neighbors. To calculate this first the number of k neighbors is set, then the distance between these neighbors is calculated with the Euclidean metric for distance and finally it assigns the samples to the class with the most samples [8].

C. *Logistic Regression (LR)*

The Logistic Regression is a mathematical model approach to describe the relationship between many independent variables (features) and a dependent variable (class) [9]. The logistic model ensures that the output of the logistic function stays between 1 and 0. The logistic function that determines the label of the value of the sample is shown as:

$$f(z) = \frac{1}{1+e^{-z}} \quad (6)$$

For these function f(z), z is the input to the function and the possible outcomes of the function are between the values 0 and 1. Now this function depicts the simplicity of the model for cases in which there are only two solutions but, in this case, the logistic model can be applied to multiple class classification like the problem at hand. Z can be expressed as:

$$Z = \alpha + \beta 1X1 + \beta 2x2 + \cdots + BkXk \quad (7)$$

Where Xs are the independent variables of interest, in this case the features, and α, β are unknown parameters to the function. If we replace the last formula with the logistic function, we will get the following model formula:

$$P(X) = \frac{1}{1+e^{-(\alpha+\sum \beta iXi)}} \quad (8)$$

Where α and β are known parameters and Xi to Xk are defined.

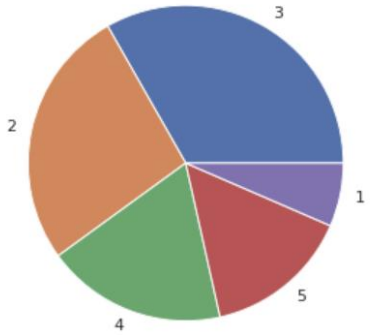

Fig. 1: Pie chart

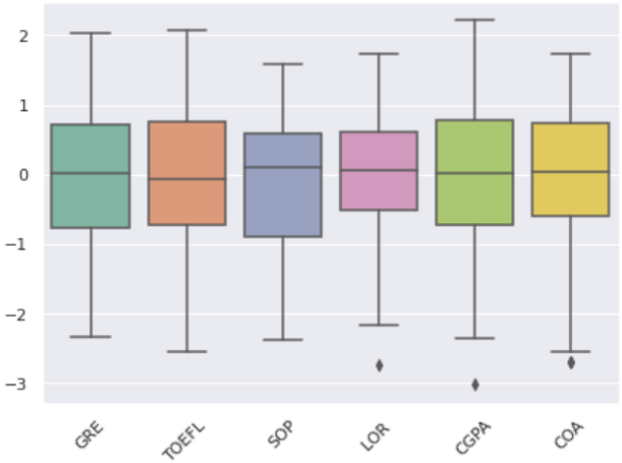

Fig. 2: Box plot

IV. Data Set Description

This dataset for admission for university was obtained from Kaggle and is the one used for the development of this project. This dataset provides the admission requirements that universities usually ask for to grant admission to new students. The class "UR" or University Rating is the column chosen for class prediction using the other features from the dataset. The University Rating has 5 possible values for prediction which have been defined before as a 1-5 rating.

From the dataset it is illustrated in Fig. 1 that some universities with the rating provided are distributed among the 400 entries of the dataset. In Fig. 1 it is shown that the distribution of university rating among the dataset is almost balanced between the options, demonstrating that there is not one type of rating that is more dominant than the others among the entries. This will be useful during the prediction process of training because there is sufficient information for all the types of rating.

The dataset has other features aside from the "University Rating" that are going to be used to make the prediction of this class. These are the 6 features from the dataset for prediction: "GRE scores (GRE)", "TOEFL scores (TOEFL)", "Statement of Purpose (SOP)", "Letter of Recommendation (LOR)", "Undergraduate GPA (CGPA)", "Chance of Admit (COA)".

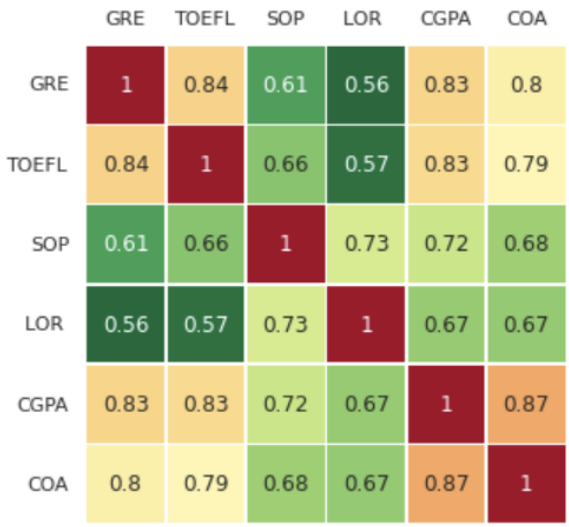

Fig. 3: Correlation matrix

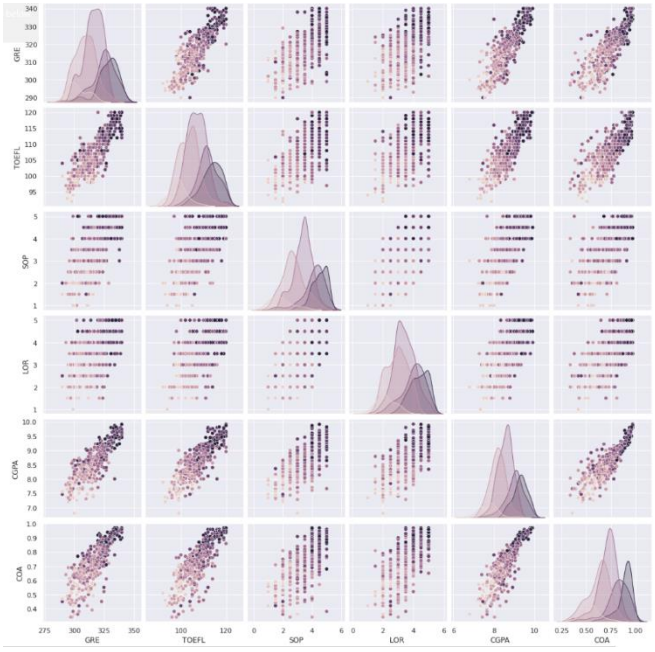

Fig. 4: Pair plot

From Fig. 2 it is shown the box plot used to illustrate the features from the data for admission dataset. The box and whiskers plot are used in this case to measure the distributions, variability, and central values of the data for the six features in the dataset. In Fig. 2, the distribution of the six features seems like a normal distribution with long whiskers in each case, there are even outliers present in the last 3 features of the dataset. Not exactly a normal distribution but tends to that type of distribution. The whiskers seem similar on both sides, but they are slightly more negatively skewed.

A correlation matrix was also used to show the values of how each feature is correlated with the other features in the database. Fig. 3 is shown with the correlation matrix for the standardized features of the dataset. From this table, it is evident that the features with the highest positive values are GRE, TOEFL, CGPA and COA. This clearly shows that

these features are highly correlated to each other. A pair plot is also used to show graphically how the features relate to each other and what kind of relation they have. Fig. 4 shows these relation ship confirming the values of the correlation matrix. The features GRE, TOEFL, CGPA and COA are the ones that have more cells in Fig. 4 and tend to a positive line that the other features with less correlation. The other 2 features do not seem to have a very defined correlation.

V. PCA Results

The first step after having standardized the dataset is to apply PCA to the dataset of admission for university to reduce the dimensionality and get the principal components to analyze them. The implementation of PCA to the data using Python can be done in two ways: to formulate the PCA analysis from scratch using common libraries or to use a predetermined PCA python library. Both methods are implemented in the Google Colab notebook file but for more precise and flexible results regarding PCA, the data and graphs shown in this report are from the PCA library.

Applying the PCA analysis, the six features from the dataset get reduced to an *r* number of features that should be less than six which is the current number of features. The original dataset of *n x p* dimensions is reduced using a matrix A of eigenvectors. Each column in matrix A is represented by a principal component (PC). Each principal component column has some variability information which is key to determine the new dimension *r* after PCA is applied to the dataset. The matrix A of eigenvectors, obtained from the data for admission dataset is the following:

$$A = \begin{bmatrix} -0.413 & -0.416 & -0.385 & -0.366 & -0.437 & -0.427 \\ -0.408 & -0.339 & 0.488 & 0.663 & -0.144 & -0.136 \\ -0.025 & 0.246 & 0.736 & -0.521 & -0.088 & -0.342 \\ 0.351 & 0.457 & -0.187 & 0.389 & -0.334 & -0.606 \\ 0.721 & -0.659 & 0.151 & -0.038 & -0.014 & -0.143 \\ 0.139 & 0.080 & 0.114 & -0.024 & -0.817 & 0.541 \end{bmatrix}$$

And the corresponding eigenvalues from the dataset are the following:

$$\lambda = \begin{bmatrix} 4.634 \\ 0.621 \\ 0.282 \\ 0.199 \\ 0.158 \\ 0.118 \end{bmatrix}$$

In the next part of the python code are some graphs that really help to get a better understanding of the amount of variability each principal component has from the PCA. In Fig. 5 is shown the scree plot and in Fig. 6 is shown the pareto chart. Both graphs illustrate the amount of variance explained by each principal component of the altered dataset. Even though on the graphs is stated that 98% of the variance is explained by the first four principal components, the first two still contain much of the variance of the data by themselves. These first two principal components contribute 87.3% of the amount of variance from the dataset. PC1 contains 77% of the variance ($\iota 1 = 77\%$) and PC2 contain 10.3% of the variance ($\iota 2 = 10.3\%$). The scree plot shows that there is a change in the form of the graph at the point of

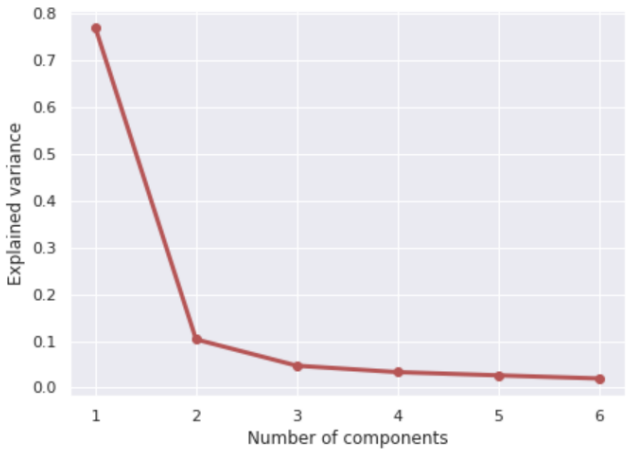

Fig. 5: Scree plot

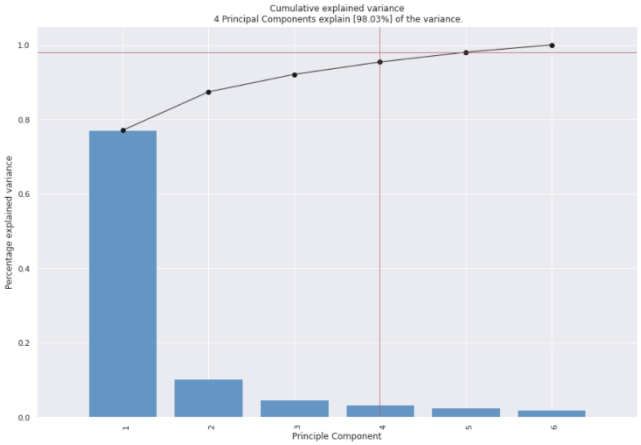

Fig. 6: Pareto chart

PC2, indicating that the dimension of features after PCA is reduced to two components (r=2).

The principal component PC1 can then be expressed as Z1 in the following equation:

$$Z1 = -0.413X_1 - 0.408X_2 - 0.025X_3 + 0.351X_4 + 0.721X_5 + 0.139X_6 \quad (9)$$

From this first principal component equation it is visible that X1 (GRE), X2 (TOEFL) and X5 (CGPA) are the ones that most contribute to the variance in PC1. The contribution from X3 (SOP) is very small and therefore can be ignored for the first PC. Therefore, Z1 can be expressed as:

$$Z1 = -0.413X_1 - 0.408X_2 + 0.351X_4 + 0.721X_5 + 0.139X_6 \quad (10)$$

The second principal component PC2 can be expressed as Z2 in the next equation:

$$Z2 = -0.416X_1 - 0.339X_2 + 0.246X_3 + 0.457X_4 - 0.659X_5 + 0.080X_6 \quad (11)$$

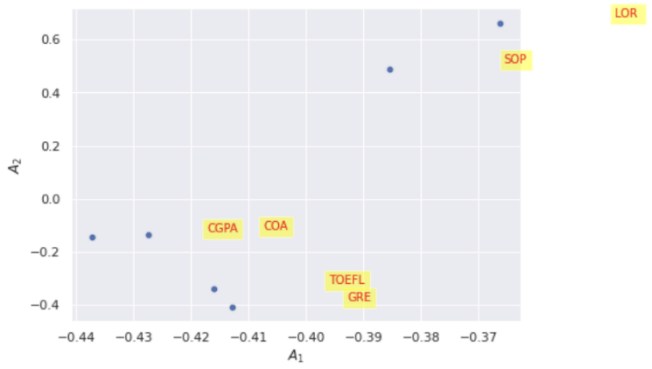

Fig. 7: Coefficient plot

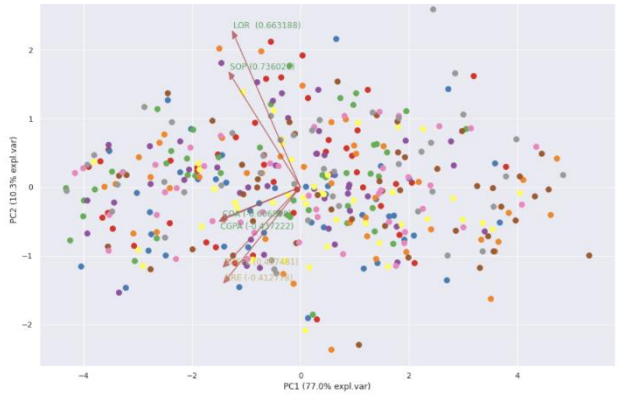

Fig. 8: Biplot

From the second principal component is evident that X1 (GRE), X4 (LOR) and X5 (CGPA) contribute the most to the second component PC2. In the other hand, X6 (COA) has a very little contribution to the variance and hence can be removed from the equation. Then, Z2 can be expressed as:

$$Z2 = -0.416X_1 - 0.339X_2 + 0.246X_3 + 0.457X_4 - 0.659X_5 \quad (12)$$

In Fig. 7 is shown the coefficient plot for the principal components where it is illustrated the amount of variability that each feature from the dataset provide to the first two PCs. The figure clearly shows the relationship of the features as expressed in the equations of Z1 and Z2, being the features GRE and TOEFL the features that contribute the most for the first two components. In the case of Z1, the major contribution is from the feature CGPA and for the Z2 component the feature of LOR as shown in the figure.

For Fig. 8 is the biplot that shows a representation of the entries and features in terms of the first two components. The axes of the graph are represented by PC1 and PC2. The eigenvectors of the matrix A are shown as vectors among the 400 entries in the dataset. It is displayed in this graph that the vectors of some features are more related to a certain principal component, and this can be measured by the angle of the vectors with each axis, in this case the PCs. This confirms the data seen in Fig. 7 gives a clearer sense of these vectors. Since some features themselves relate to the first two components it is not so clear to determine which features are more related to the components. This is also observed

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.5898	0.8399	0.4987	0.5650	0.5656	0.4403	0.4490	0.245
lda	Linear Discriminant Analysis	0.5858	0.8391	0.4876	0.5687	0.5598	0.4336	0.4427	0.020
nb	Naive Bayes	0.5778	0.8278	0.5879	0.5522	0.5469	0.4384	0.4487	0.020
et	Extra Trees Classifier	0.5778	0.8273	0.4896	0.5631	0.5592	0.4255	0.4310	0.206
gbc	Gradient Boosting Classifier	0.5502	0.8031	0.4942	0.5540	0.5420	0.3956	0.4000	0.421
ridge	Ridge Classifier	0.5388	0.0000	0.4098	0.4323	0.4756	0.3560	0.3698	0.019
lightgbm	Light Gradient Boosting Machine	0.5380	0.8014	0.4798	0.5451	0.5302	0.3799	0.3852	0.094
knn	K Neighbors Classifier	0.5298	0.7821	0.4245	0.5331	0.5093	0.3557	0.3650	0.027
qda	Quadratic Discriminant Analysis	0.5178	0.7891	0.4528	0.4925	0.4922	0.3492	0.3559	0.018
lr	Logistic Regression	0.5142	0.7955	0.4200	0.4581	0.4714	0.3299	0.3415	0.324
dt	Decision Tree Classifier	0.4583	0.6338	0.4355	0.4650	0.4461	0.2715	0.2762	0.017
ada	Ada Boost Classifier	0.4458	0.6399	0.3550	0.4178	0.4031	0.2375	0.2533	0.110
dummy	Dummy Classifier	0.3546	0.5000	0.2000	0.1259	0.1858	0.0000	0.0000	0.019
svm	SVM - Linear Kernel	0.2837	0.0000	0.2294	0.1085	0.1499	0.0413	0.0618	0.025

Fig. 9: Comparison table of classification models before PCA is applied

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.5811	0.8272	0.4864	0.5420	0.5504	0.4093	0.4189	0.033
lda	Linear Discriminant Analysis	0.5803	0.8346	0.5401	0.5654	0.5552	0.4189	0.4300	0.018
nb	Naive Bayes	0.5605	0.8215	0.4690	0.5160	0.5289	0.3853	0.3944	0.018
qda	Quadratic Discriminant Analysis	0.5600	0.8240	0.4905	0.5246	0.5314	0.3915	0.4004	0.018
lightgbm	Light Gradient Boosting Machine	0.5389	0.7958	0.5294	0.5449	0.5263	0.3805	0.3870	0.072
et	Extra Trees Classifier	0.5337	0.7957	0.5021	0.5505	0.5289	0.3698	0.3745	0.191
ridge	Ridge Classifier	0.5232	0.0000	0.3693	0.4258	0.4632	0.3174	0.3302	0.015
gbc	Gradient Boosting Classifier	0.5197	0.7836	0.4825	0.5138	0.5094	0.3413	0.3449	0.358
rf	Random Forest Classifier	0.5139	0.8017	0.4823	0.5159	0.5016	0.3419	0.3481	0.237
knn	K Neighbors Classifier	0.5026	0.7810	0.4490	0.5112	0.4949	0.3166	0.3213	0.033
svm	SVM - Linear Kernel	0.4742	0.0000	0.4022	0.3585	0.3931	0.2673	0.2977	0.020
dt	Decision Tree Classifier	0.4724	0.6443	0.4572	0.4921	0.4600	0.2902	0.2981	0.017
ada	Ada Boost Classifier	0.3647	0.6711	0.3905	0.3984	0.3327	0.1934	0.2216	0.106
dummy	Dummy Classifier	0.3539	0.5000	0.2000	0.1255	0.1853	0.0000	0.0000	0.016

Fig. 10: Comparison table of classification models after PCA is applied

with the values of variances for the components, the first two components add up to more than 85% of the variance in the dataset but there is still some contribution from the other components until the PC4 which rounds about 98% of the total variance of the data. These values can also be seen in Fig. 8 of the biplot to confirm the total percentage of variance for the first two principal components.

VI. Classification Results

For this section, the three classification models selected are analyzed for its performance on the data for admission dataset. To get a better understanding of how important it is to reduce dimensionality on a dataset by applying PCA, the multi class classification models are applied to the original dataset and the dataset altered by PCA with 2 PCA components. This classification is used with the predefined python library of PyCaret. Before applying the models, the dataset before PCA is separated into two groups used for classification: training set and test set which have a proportion of 70% and 30% for each group respectively. The session is also set to 123 to be able to reproduce the same results when this session is joined.

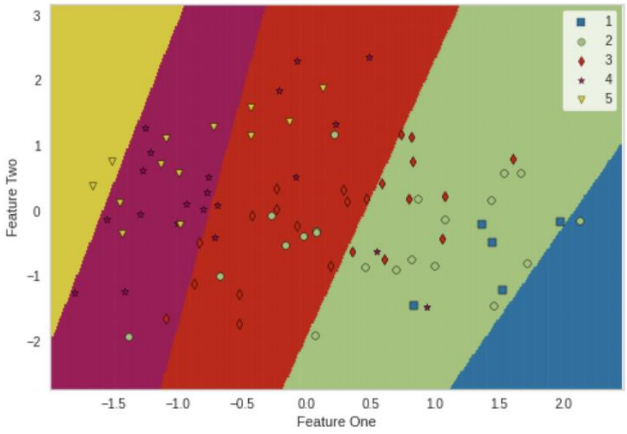

Fig. 11: Decision boundary

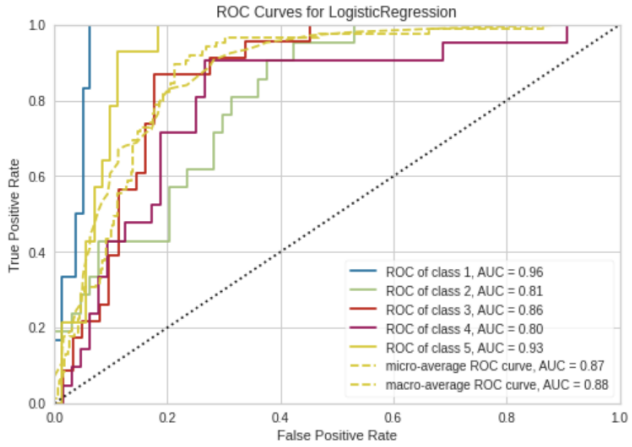

Fig. 12: ROC curve for LR model

In the PyCaret library there is a way to make a comparison between the models available on the python library and choose the best option, in terms of accuracy, to apply on the dataset. In Fig. 9 it is visible that the top model before applying PCA on the dataset is Random Forest Classifier (RF) with the highest accuracy among the available models.

After applying PCA to the dataset, the best models in terms of accuracy are the ones listed in Fig. 10. This is also a list generated from the PyCaret library. In Fig. 10 is clear that the best model for the dataset in this project is Logistic Regression (LR). For the other two models chosen to be applied, the Decision Tree Classifier (DT) and K-Neighbors Classifier (KNN), these models were chosen for having a better understanding of how these models work rather than being on top of the list of models. Since accuracy also depends on the number of entries to train the dataset, these models were chosen on biased for this project. However, since the library does give a list of the best models are PCA is applied it is also considered to use the model of Logistic Regression, being this model the best model available in accuracy.

To improve the accuracy of the models, tuning using hyperparameters is applied to the models. There is function in the PyCaret library that allows to use these hyperparameters, which work in the following manner: first create the model, then tune the model and lastly evaluate the performance of the tuned model. After each model is created for the types chosen for classification, the function tune_model() is used to tune the models using hyperparameters. This function tunes the model using effective hyperparameters on a search space predefined on the library and then uses stratified K-fold cross validation. It can be seen on the Google Colab notebook that the metrics of the models after using the hyperparameters tuning method are better than the metrics on the models before tuning them.

In Fig. 11 is shown the decision boundary of the model Logistic Regression on the dataset altered after PCA was applied. The decision boundary displays a hyperplane that is divided into different decision regions that represent the classes of the feature that is being predicted [10]. The axis represents the principal components of the data: the horizontal axis is PC1, and the vertical axis is PC2. The different types of dots scattered on the different regions of the plot represent the 1-5 different classes of the University Rating feature that is being tried to predict. It was chosen to use the decision boundary of the Logistic Regression model for being the most accurate model among the models used for this project and it shows how the different features are separated more accurately among the regions.

Another analysis that can be done in Fig. 12 is the receiver operating characteristic curve (ROC) curve for the Logistic Regression model. This graph displays the performance of a classification model at the threshold of all features. It contains two parameters: True Positive Rate and False Positive Rate. Each parameter is represented in the graph as the axes: true positive rate is the vertical axis and false positive rate is the horizontal axis. With these parameters it is shown how accurate the model is to predict a specific class of the feature trying to predict. The LR model is chosen to depict the ROC curve as it clearly shows the accuracy this model possesses over the other 2 models chosen. The ROC curves for the other models can be found on the Google Colab notebook. As shown on the graph, this model predicts the features with 88% accuracy. So, it is possible to say that the LR model is accurately classifying the 1-5 ratings for the university rating (UR) using the other features as data for prediction.

VII. Explainable AI with Shapley Values

Shapley Values help to get a better understanding of a model and to see the amount of importance each feature in the dataset used has. This helps to measure the proportion from each feature and was used to make a prediction. To see this relevant information Shapley Values were used. For this project was used a python library that contained SHAP values to make this measurement.

Shapley values is a concept based on cooperative game theory. Like in a game, the Shapley value helps to determine the distributed collective value of the individuals in a team game [11]. Like in this example, the Shapley values help to determine the grade of importance each feature in the data for admission must have in the prediction. This is known to be a very important aspect of Machine Learning. The players of a team can be the features of the dataset. These Shapley values are a tool to accurately distribute the percentage of prediction each feature possesses.

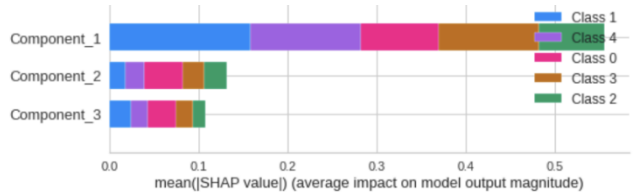

Fig. 13: Shapley Summary Plot

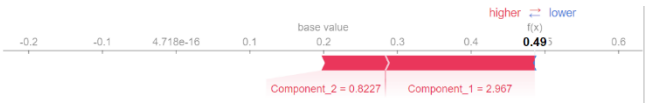

Fig. 14: Force plot for the 32nd observation

There is an inconvenience with usage of the SHAP python library, as it is an open source and still in development stage, only the models based on tree classification models are supported. This would include the Decision Tree Classifier used om the classification section but since the Random Forest Classifier (RF) is also part of the supported models and have a higher accuracy than the DT model, as seen in the list of the Fig. 10, this model is used to perform Shapley analysis. Like the process that the other models were subjected to, the RF model was created and tuned with hyperparameters. Then the tuned model is subjected as input for the SHAP library to get the graphs for interpretation.

In Fig. 13 is displayed the summary graph of the SHAP values. The summary plot contains the overall feature importance in terms of the principal components. Each bar for the principal components is divided into segments of different colors that represent the amount of importance each for each value in the class predicted. The Shapley values are represented in the horizontal axis and the principal components PCs are on the vertical axis. Since the first PCs are the ones that contain the most variability, they are the ones represented with the Shapley values on the graph. The PCs are vertically ordered in terms of importance. This order is supported by the PCA since the majority of variance is stored on the first components. This can also be seen from the scree plot in Fig. 5 and the pareto chart in Fig. 6. From the summary plot, it can be inferred that the lower the PC value the higher is the impact on the university rating prediction and vice versa. In other words, the PC values are negatively correlated with the classes of the university rating variable.

In Fig. 14, is the force plot from a single observation. This type of plot can only be made for one observation to be analyzed. In this case, the 32nd observation was selected to be displayed. This plot shows the contribution each feature must make to the output of the model. The base value is 0.2 in the graph. Since both components in the observation are the same color red, means that both components push the model for the same class output. But this graph does not contain the information for the whole model since it is only the prediction for a single observation.

The last graph for this analysis is the force plot for all principal components that can be seen in Fig. 15. This plot sums up all the force plots from the model and shows them horizontally. There are 120 points in the dataset therefore

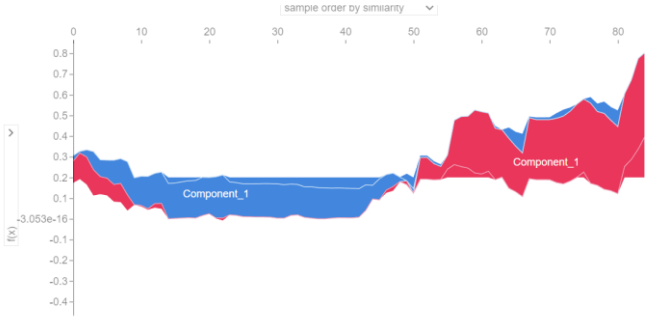

Fig. 15: Combined force plot

120 observations in the combined force plot. In this graph is illustrated how much the principal components influence on the prediction of the rating. The areas with red mean a negative influence the PCs have on the prediction and blue areas the positive influence on the prediction.

VIII. Conclusion

In conclusion, the PCA and the three classification models were applied to the dataset to make predictions about the ratings of universities with a certain degree of accuracy. The dataset of data for admission has various features that were used to define a rating from 1-5 to universities. At the PCA applied to the dataset, it was found that the first two principal components contain 87.3% of the total variance of the data. Therefore, the dimension of the dataset was reduced from 6 to 2 PCs. Some calculations and graphs were produced to validate the results obtained from the PC analysis. Then, the three classification models selected, LR, DT, KNN, were applied to the dataset after the PCA was conducted. Each algorithm was then tuned with hyperparameter to increase its performance to get better results on the ROC curves and decision boundaries to determine the accuracy to classify of these models. After PCA was applied to the dataset, the accuracy and overall performance of the models was increased. Finally, AI Shapley Values were used to get a better understanding of the model used for this section and the graphs were produced to make an interpretation of the model's prediction. In the end, the three proposed models were applied correctly to make an accurate prediction of the ratings of universities.

References

[1] K. Moon, "Forbes," 3 October 2019. [Online]. Available: https://www.forbes.com/sites/kristenmoon/2019/10/03/the-strategy-behind-building-your-college-lists/?sh=3855f7d85c77. [Accessed 30 Noviembre 2022].

[2] R. Morse, "USNews," 11 September 2022. [Online]. Available: https://www.usnews.com/education/best-colleges/articles/ranking-criteria-and-weights. [Accessed 30 November 2022].

[3] T. J. Chuanfu Ding, "The Relationship Between University Ranks and Outcomes Measurement," *College Teaching Methods & Styles Journal (CTMS)*, pp. 1-9, 2007.

[4] L. J. W. Herve Abdi, "Principal Component Analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. p. 433-459, 2010.

[5] A. B. Hamza, Advanced Statistical Approaches to Quality, Unpublished.

[6] Z. Z. Wenliang Du, "Building Decision Tree Classifier on Private Data," *Electrical Engineering and Computer Science*, no. 8, 2002.

[7] S. Jaiswal, "JavaTpoint," [Online]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. [Accessed 2 December 2022].

[8] R. H. Shiliang Sun, "An Adaptive k-Nearest Neighbor Algorithm," in *Seveenth International Conference on Fuzzy Systems and Knowledge Discovery*, Shanghai, 2010.

[9] M. K. David G. Kleinbaum, Logistic Regression, Atlanta: Springer.

[10] Z. W. S. C. G. L. Ningyuan Gong, "Decision Boundary Extraction of Classifiers," *Journal of Physics: Conference Series*, no. 1651, 2020.

[11] L. W. P. B. O. K. Benedek Rozemberczki, "Cornell University," 26 May 2022. [Online]. Available: https://arxiv.org/pdf/2202.05594.pdf. [Accessed 15 December 2022].