FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Universidade de Coimbra

Pattern Recognition

Turma:PL2

Final Report

Rafael Correia Molter
Yandra Vinturini Vieira Dantas
Beatriz Negromonte
May/22

Contents

Abstract

According to the World Health Organization, 18 million people die annually due to a heart disease, making it one of the leading causes of deaths worldwide. Therefore, correct diagnosis is crucial to prevent those fatalities. The present work aims to detect patterns in data gathered from the Centers for Disease Control and Prevention (CDC), predicting whether a person has a heart disease or not. Data is processed through techniques of feature selection and dimensionality reduction, and used to train several models of classifiers, all that implemented using Python programming language.

1 Introduction

Cardiovascular diseases are physical conditions that manifest due to heart or blood vessels damages, and nowadays are responsible for 32% of annual deaths in the world, about 18 million people. Most heart diseases can be prevented from happening through adoption of a healthy lifestyle, with a balanced diet, regular physical exercise and moderate consume of alcohol. The detection of those conditions in early stages is crucial, as treatment might still take place depending on the disease and prevention of death can occur. However, the medical community has issues to diagnose patients early, as most of the symptoms are also symptoms of other simple diseases or are mistaken with aging signals [1].

The present report stems from the project assignment "Heart Disease prediction", for Pattern Recognition course of University of Coimbra, academic year 2021/2022. In this work, pattern recognition methods were implemented in order to develop several classifiers, namely: Fisher's Linear Discriminant (FLD) , Minimum Distance Classifiers (MDC) with Euclidean and Mahalanobis distances, Naive Bayes, K-Nearest Neighbors (KNN) and Support Vector Machines (SVM).

Along the work we had mainly three goals, which from now on we are going to refer as scenario A, scenario B and scenario C. In the first scenario, we used our classifiers to detect whether a person in our dataset had coronary heart disease (CHD) or not. For scenario B, the objective was to distinguish which heart disease the patient had: CHD or myocardial infarction (MI), commonly known as a heart attack. The latter case was a multi-class problem where the first class is someone who has a heart disease and is living in a situation of comorbidity, i.e., having more than one condition. In our case, it includes kidney disease and/or skin cancer. Second class is a person who has either one of heart diseases but no other condition, and third class includes people who have no heart disease.

The work is structured as follows: in the methods section we present how the data was cleaned and pre-processed, as well as the feature selection, dimensionality reduction and classifying techniques used. We then present the results for each model trained with the different pipelines applied. The fourth section is a discussion of the results obtained, followed by our conclusions in the last section.

2 Methods

2.1 Dataset

Our dataset is a filtered version of a survey made by the CDC with United States residents, with questions regarding the health status of the respondents. It contains 318958 entries with 4 dependent variables - Coronary Heart Disease, Myocardial Infarction, Kidney Disease and Skin Cancer - and 15 independent variables - body mass index (BMI), smoker, alcohol drinking, stroke, physical health, mental health, difficulty of walking, sex, age category, race, diabetic, physical activity, general health, sleep time and asthma.

The original data is organized in the following way:

Dependent variables:

- CoronaryHeartDisease/MyocardialInfarction/KidneyDisease/SkinCancer: 2.0 for negative; 1.0 for positive

Independent variables:

- Smoking - whether or not they have smoked at least 100 cigarettes in life: 2.0 for no; 1.0 for yes
- Stroke - whether or not they have had a stroke: 2.0 for no; 1.0 for yes
- DiffWalking - whether or not they have serious difficulties walking or climbing stairs: 2.0 for no; 1.0 for yes
- Asthma - whether or not they have asthma: 2.0 for no; 1.0 for yes
- PhysicalActivity - whether or not they have done physical activity in the past 30 days: 2.0 for no; 1.0 for yes
- AlcoholDrinking - whether or not they are heavy drinkers: 1.0 for no; 2.0 for yes
- Sex - 2.0 female; 1.0 male
- AgeCategory - 80 or older: 13 / 75 - 79: 12 / 70 - 74: 11 / 65 - 69: 10 / 60 - 64: 9 / 55 - 59: 8 / 50 - 54: 7 / 45 - 49: 6 / 40 - 44: 5 / 35 - 39: 4 / 30 - 34: 3 / 25 - 29: 2 / 18 - 24: 1
- Race - 1 White; 2 Black; 3 Asian; 4 American indian/Alaskan native; 5 Hispanic; 6 Other
- Diabetic - 1 Yes; 2 Yes, during pregnancy; 3 No; 4 Borderline diabetes

- GenHealth - person's general health: 1 Excellent; 2 Very good; 3 Good; 4 Fair; 5 Poor
- SleepTime - average number of hours sleeping in a 24-hours period

2.2 Data Preprocessing

Data cleaning and subsets

The dataset has both qualitative and quantitative features. The qualitative data was changed from float to integer, and the features that originally had negative as 2.0 and positive as 1.0 was changed to 0 and 1, respectively. The dataset was also validated for any missing data, returning a result that there was no missing data. Since we had three different scenarios, we divided our data into subsets. For scenario A, the only column used as labels was CoronaryHeartDisease. For scenario B, we discarded columns KidneyDisease and SkinCancer. Furthermore, the only samples taken into consideration was the ones that had a positive value for CoronaryHeartDisease or - exclusive - MyocardialInfarction, since we wanted to distinguish between those two conditions. For scenario C, all dependent variables are taken into account.

Data Visualization

In order to gain a better understanding of the distribution of our quantitative features' values, the data was visualized through histograms and box plots (Figures 1, 2, 3 and 4)

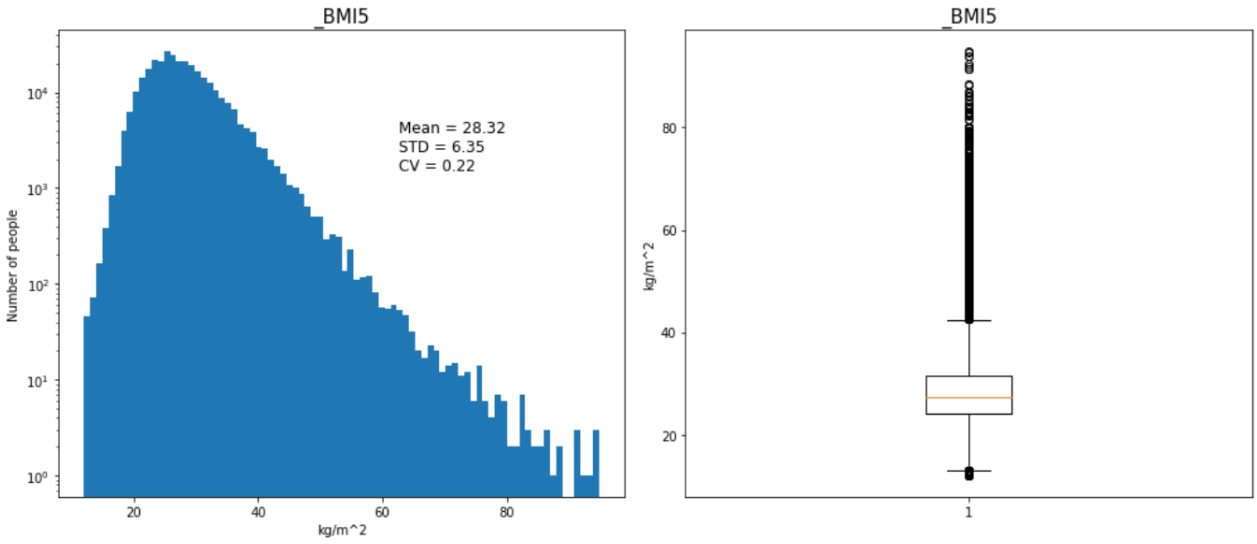

Figure 1: Body Mass Index

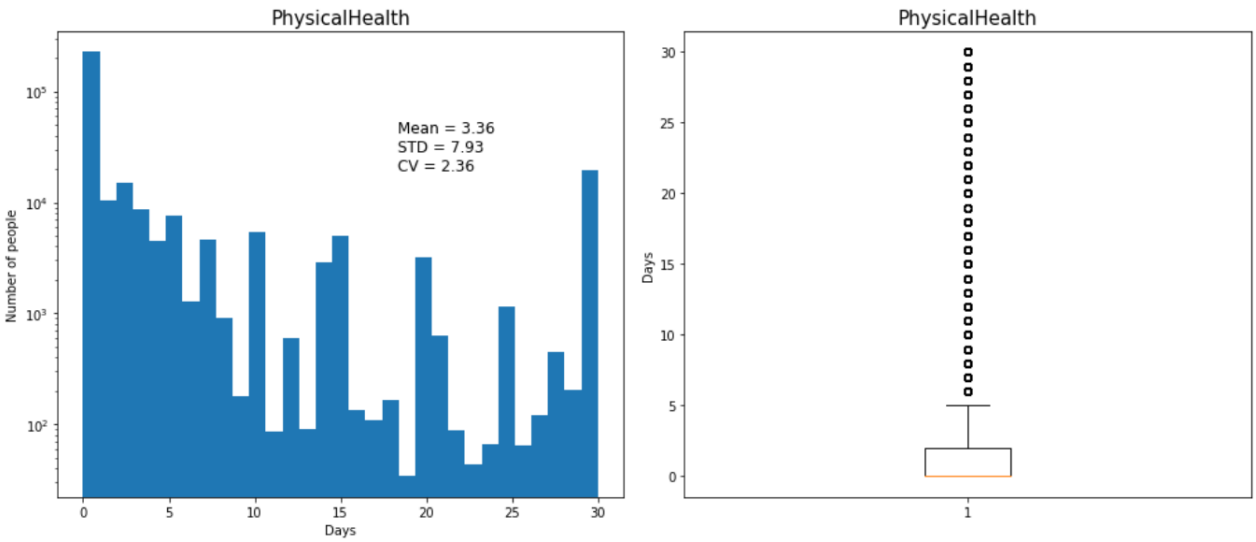

Figure 2: Physical Health

Multiple outliers can be seen in all features, especially for the body mass index (BMI) feature, compared to the others. Besides that, the standard deviation is high for both Physical and Mental health, which is determined by the coefficient of variation above 1. Also, all 4 features seem to have a non-gaussian distribution.

Resampling

Our dataset is highly unbalanced towards the healthy class. As the algorithm receives more examples from one class, it will likely be biased towards that particular class, failing to understand the underlying patterns that allow us to distinguish classes. Considering that, we made use of a naive resampling technique such as random oversampling and undersampling in the data untill balance between classes was achieved. After that, we could test if the classifiers have poor prediction on an unbalanced dataset [2][3]

In scenario A, more than 90% of the labels are negative for CHD. We oversampled the minority class to 80% of majority class, based on the available data. After that we undersampled majority class so that the final distribution is 50% for each class.

For Scenario B, the classes already had a pretty close distribution, so resampling it would not have much impact and therefore we chose to save computational power and not do it.

Finally, Scenario C has 91,68% of class 0 (no heart diseases), 5,99% of class 1 (heart disease but no comorbidities) and 2,33% of class 2 (heart disease and comorbidities). We started by oversampling classes 1 and 2 to 80% of the majority class. After, class 0 was undersampled to match the same number of other classes.

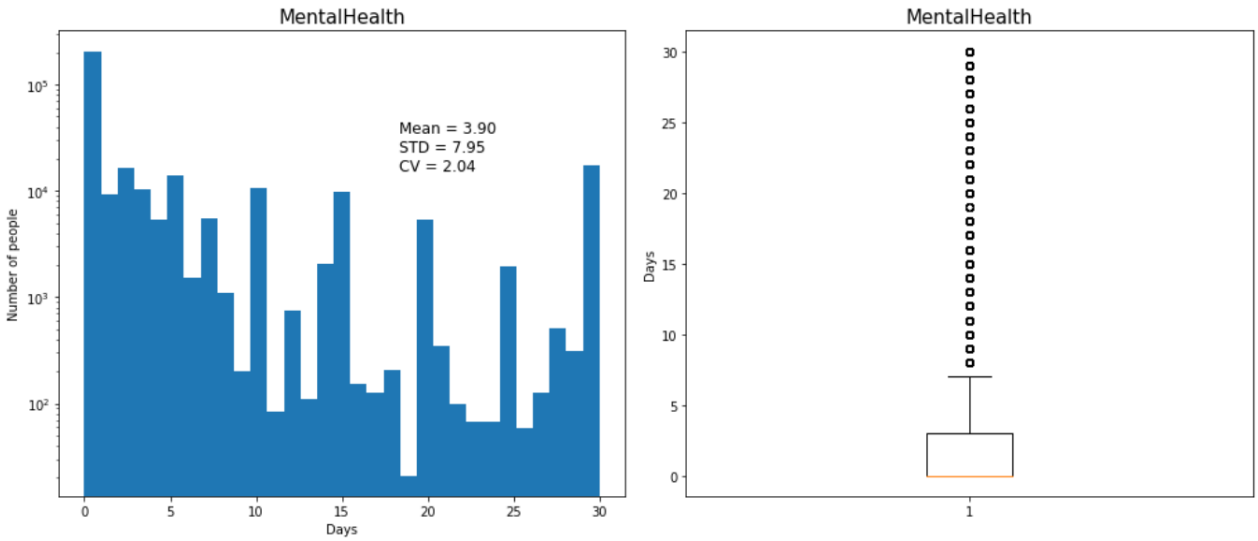

Figure 3: Mental Health

Scaling

Before applying Principal Component Analysis and Linear Discriminant Analysis, we normalized our features with StandardScaler function, dividing them first into train and test sets to avoid data leakage, with approximately the same proportion of classes on both sets. The StandardScaler function standardizes features through Equation 1, where X is the value of the sample's feature, u is the mean of that feature across all samples and s is the standard deviation [4][5].

$$z = \frac{(x - u)}{s} \tag{1}$$

After that, the features correlation were analysed through the covariance matrix seen in Figure 5. The color heatmap goes from blue for a negative correlation and red for a positive correlation. The most uncorrelated features are shown in gray. With that, We can see that the most correlated features are PhysicalHealth and DiffWalking, PhysicalHealth and GenHealth, and DiffWalking and GenHealth.

2.3 Feature selection

To avoid overfitting, improve model prediction and make it computationally more efficient, feature selection was done. The method chosen was Kruskal Wallis, which is a filter method that only takes into account the intrinsic properties of the data and calculates a score value for each feature [6]. In our case, we used the H score to rank the best features according to the higher score. Being that filter techniques are independent from the classification algorithms [6], we only had to

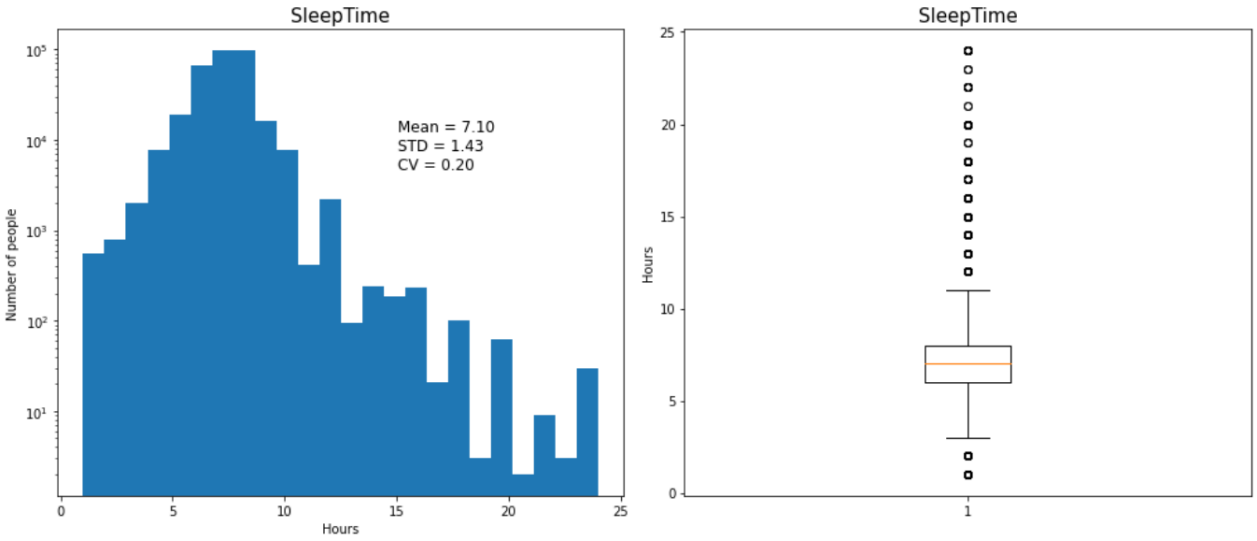

Figure 4: Sleep

do feature selection once and could apply it for every classifier. Kruskal Wallis is a non-parametric statistical test, i.e., it does not assume any particular distribution of the data. The null hypothesis to the test is that the groups of data have the same mean. To select features based on that, one has to separate each feature in C groups, being C the number of classes. The test is then applied to see if features are significantly different between classes [7], based on the p-value and H-score, and the K best features are chosen.

2.4 Dimensionality reduction

Principal Component Analysis

We apply Principal Component Analysis in order to reduce the dimensionality, through the performance of a orthonormal transformation to retain only the significant eigenvectors and consequently, the eigenvalues of those corresponding eigenvectors. The goal is to obtain the directions that have greater variance of the data in a given direction [8][9][10], and they are obtained through the eigenvectors and eigenvalues of the covariance matrix in Figure 5.

The accumulated variance against the number of components in PCA was plotted in graphs for each scenario. The dimension chosen is according to an observed knee of the curve, meaning that choosing one more dimension does not add much more variance to the data. If the graph did not show any knee point, we chose a sufficient number of dimensions so as not to remove much variance.

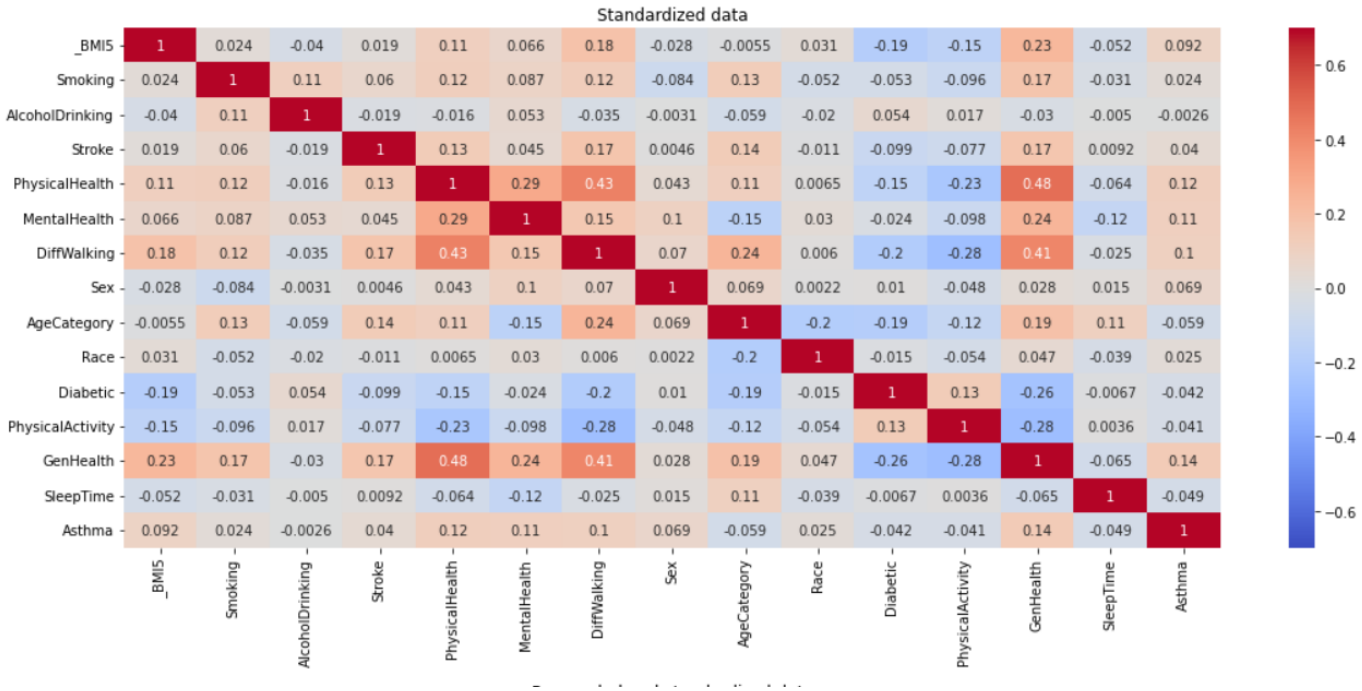

Figure 5: Covariance matrix for standardized features

Linear Discriminant Analysis

Other than the Principal Component Analysis, we also test the Linear Discriminant Analysis (LDA) technique to reduce the features number, where the focus here to maximize the separability of the class means presented while reducing dimensions at the same time [11].

LDA always uses a number of components that is the minimum between the number of classes minus one and the number of features. In our case, since we have 15 features and 2 classes for Scenarios A and B, LDA will choose the minimum between 15 and 1. Therefore, the dataset is reduced to only one dimension. For Scenario C it is reduced to two dimensions, since it has three classes. In our code, we implement the LDA with Eigen decomposition as solver, using sklearn.

2.5 Classification

Fisher's Linear Discriminant

The Fisher linear discriminant provides the possibility to use the classification information of the training set to produce an optimised mapping to a lower dimensional space. The goal is to maximize the class separability through the choice of a direction in the feature space where the distance of the means relative to the within-class variance is maximum [8].

The LDA model used for dimensionality reduction already has the decision function calculated within. Thus, we then just apply the predict function on the test set for each scenario. For the sets reduced using PCA, we first need to also apply LDA to create a model that has the decision function calculated.

Minimum Distance Classifier - Euclidean and Mahalanobis

We applied two methods for the minimum distance classifier: Euclidean and Mahalanobis.

For the Euclidean distance classifier, there's a generalization for any d-dimensional feature vector x and any number of classes, represented by their prototype m_k, to represent the shortest distance between two points. The square of the euclidean distance between the feature vector x and the prototype m_k is given by Equation 2:

$$d_k^2(x) = ||x - m_k||^2 = (x - m_k)'(x - m_k) = x'x - m_k'x - x'm_k + m_k'm_k \tag{2}$$

The Mahalanobis metric is a generalization of the Euclidean metric, for unequal variances and correlated features, for obtaining a linear discriminant function in the form of hyperplanes passing through the middle point of the line segment linking the means. Those hyperplanes that will be separating the 2 classes are orthogonal to the vector [8]. After calculating both metrics, we are then able to classify the samples in the sets reduced using PCA and LDA.

Naive Bayes

Naive Bayes classifier is a probabilistic algorithm based on the Bayes theorem, rewritten in the context of classes as shows Equation 3.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{3}$$

That means that we can find the probability of y happening being that X has occurred, i.e., the probability of the given sample X belonging to class y. This algorithm makes to assumptions, which is the reason why it is called naive: it assumes that the features are totally independent from each other, and one does not have more influence over the other [12]. The class where the observation belongs is the one that has the greater probability based on the Bayes formula.

K-Nearest Neighbors

K-nearest neighbors is a supervised machine learning algorithm used for classification and regression problems. It is based on the assumption that points belonging to the same class are found

near one another in the space dimension. For each new observation where we don't know the label, the point's distance to its neighbors is calculated and is assigned the label which is closer to it [13]. For this algorithm, we need to select the K number of neighbors, which in our case 100 was chosen.

Support Vector Machines

A Support Vector Machine is an algorithm that calculates a hyperplane that best separates the classes of the dataset. The hyperplane should have the highest possible margin - distance between the nearest point from each class and the plane. Nevertheless, the algorithm always gives preference to the hyperplane that has the minimum prediction error, even if there is other plane with higher margin [14]

If the problem is not linearly separable, SVM uses a mathematical function called kernel to transform the data into a higher dimensional space, making the observations separable [14]. We here used the *rbf* and*linear* kernels, with tunable hyperparameters *C*, *gamma* and *tolerance*.

2.6 Graphic User Interface

Since there are many parameters to choose as we have different pre-processing techniques and classifiers being used, we develop a Graphic User Interface (GUI) to facilitate user interaction, pipeline understanding and application of numerous tests.

The GUI is coded inside our Google Colaboratory notebook, using the library *ipywidgets*. Using a set of different widgets, it will permit our user to choose between various parameters to make the classifier model.

3 Results

For all scenarios, we selected the 8 best features according to the Kruskal Wallis test. The best features for each scenario were:

- Scenario A - GenHealth, AgeCategory, DiffWalking, Diabetic, PhysHealth, Stroke, Smoking and PhysActivity.
- Scenario B - _BMI5, AgeCategory, Stroke, SleepTime, GenHealth, PhysHealth, Smoking and MenHealth.
- Scenario C - GenHealth, AgeCategory, DiffWalking, Stroke, Diabetic, PhysHealth, _BMI5 and Smoking.

For the Principal Component Analysis, we selected 7 components with all scenarios, were we observed to have a knee in the curve (even if it was very slight) as demonstrated in the example for

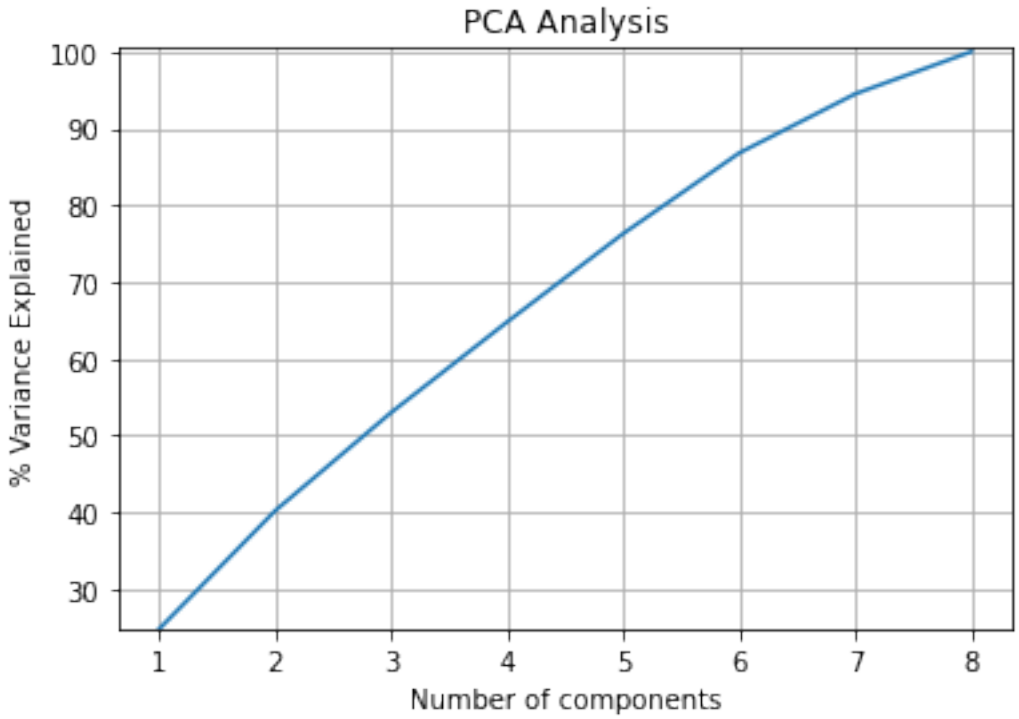

Figure 6: PCA graph for scenario B

scenario B in Figure 6.

The results after feature selection and dimensionality reduction (with PCA and LDA) for every classifier in each scenario can be seen in the tables and matrix confusions in the attached document. The metrics calculated were the specificity, accuracy, sensitivity, precision and F1 score.

4 Conclusion

We were able to conclude the proposed work of implementing different classification models for the three different scenarios A, B and C. Overall, the results were reasonable and have potential to be improved with a more thorough pipeline and data pre-processing, as well as tuning more parameters, although that would require a greater computational power. Moreover, the GUI, although it has its code written, there are still some errors during code running and we still could not debug it successfully.

Table 1: Metrics for scenario A with 5 features and 4 components k=47

Metric	Fisher LDA	Euclidean	Mahalanobis	NaiveBayes	KNN	SVM
Accuracy	74.98%	73.23%	75.00%	73.09%	75.67%	75.17%
Sensitivity	78.93%	69.10%	78.71%	70.69%	83.29%	77.76%
Specificity	71.03%	77.36%	71.28%	75.48%	68.05%	72.59%
Precision	73.15%	75.32%	73.27%	74.25%	72.27%	73.94%
F1 score	0.76	0.72	0.76	0.67	0.77	0.76

Table 2: Metrics for scenario A with 10 features and 8 components KNN

Metric	k=1	k=3	k=8	k=15
Accuracy	94.87%	92.70%	88.45%	82.50
Sensitivity	98.15%	99.29%	98.82%	95.95%
Specificity	91.59%	86.11%	78.09%	69.04%
Precision	92.11%	87.73%	81.85%	75.61%
F1 score	0.95	0.93	0.90	0.85

Table 3: Metrics for scenario A with 10 features and 8 components

Metric	Fisher LDA	Euclidean	Mahalanobis	NaiveBayes	KNN	SVM
Accuracy	74.75%	72.58%	74.77%	71.30%	94.87%	74.72%
Sensitivity	76.47%	66.94%	76.42%	65.38%	98.15%	76.21%
Specificity	73.03%	78.21%	73.12%	77.22%	91.59%	73.23%
Precision	73.93%	75.45%	73.98%	74.16%	92.11%	74.01%
F1 score	0.75	0.71	0.75	0.69	0.95	0.75

Table 4: Metrics for scenario B with 5 features and 4 components(k=467)

Metric	Fisher LDA	Euclidean	Mahalanobis	NaiveBayes	KNN	SVM
Accuracy	56.45%	56.34%	56.51%	57.09%	56.84%	56.67%
Sensitivity	78.86%	68.37%	66.36%	80.10%	81.76%	80.67%
Specificity	30.81%	42.58%	45.24%	30.75%	28.33%	29.21%
Precision	56.60%	57.67%	58.10%	56.96%	56.62%	56.60%
F1 score	0.66	0.63	0.62	0.67	0.67	0.67

References

[1] W. H. O. (WHO), "Cardiovascular diseases (cvds)." `https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)`, 2021. [Online; accessed 22-April-2022].

[2] G. Lahera, "Unbalanced Datasets & What To Do About Them." `https://medium.com/`

Table 5: Metrics for scenario B with 10 features and 8 components(k=460)

Metric	Fisher LDA	Euclidean	Mahalanobis	NaiveBayes	KNN	SVM
Accuracy	56.87%	56.62%	57.17%	56.51%	56.07%	56.84%
Sensitivity	75.76%	58.81%	54.58%	77.26%	83.15%	75.76%
Specificity	35.25%	54.11%	54.58%	32.76%	27.07%	35.19%
Precision	57.24%	59.46%	59.96%	56.80%	55.95%	57.22%
F1 score	0.65	0.59	0.60	0.65	0.67	0.65

Table 6: Metrics for scenario C with 5 features and 4 components

Metric	Fisher LDA	Euclidean	Mahalanobis	NaiveBayes	KNN	SVM
Accuracy	74.98%	73.23%	75.00%	73.09%	75.67%	75.17%
Sensitivity	78.93%	69.10%	78.71%	70.69%	83.29%	77.76%
Specificity	71.03%	77.36%	71.28%	75.48%	68.05%	72.59%
Precision	73.15%	75.32%	73.27%	74.25%	72.27%	73.94%
F1 score	0.76	0.72	0.76	0.67	0.77	0.76

`strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd`, 2019. [Online; accessed 14-April-2022].

[3] K. Pykes, "Oversampling and undersampling. a technique for imbalanced... —." `https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf`, 2020. [Online; accessed 15-April-2022].

[4] J. Brownlee, "How to transform data to better fit the normal distribution." `https://machinelearningmastery.com/how-to-transform-data-to-fit-the-normal-distribution/`, 2018. [Online; accessed 17-April-2022].

[5] K. Chouhbi, "Preprocessing data: Feature scaling." `https://towardsdatascience.com/preprocessing-data-feature-scaling-cc28c508e8af`, 2020. [Online; accessed 14-April-2022].

[6] Y. Saeys, I. A. Inza, and P. L. Aga, "A review of feature selection techniques in bioinformatics," *BIOINFORMATICS REVIEW*, vol. 23, pp. 2507–2517, 2007.

[7] S. Vora and H. Yang, "A comprehensive study of eleven feature selection algorithms and their impact on text classification," in *2017 Computing Conference*, pp. 440–449, 2017.

[8] J. P. M. d. Sa, *Pattern recognition : concepts, methods, and applications / J.P. Marques de Sa.* Berlin ;: Springer, 2001 - 2001.

[9] E. Tavares, "Principal component analysis (pca) in python using scikit-learn." https://etav.github.io/python/scikit_pca.html, 2017. [Online; accessed 20-April-2022].

[10] J. VanderPlas, "In depth: Principal component analysis." https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html, 2019. [Online; accessed 23-April-2022].

[11] P. Sarkar, "What is lda: Linear discriminant analysis for machine learning." https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning, 2019. [Online; accessed 20-April-2022].

[12] R. Gandhi, "Naive bayes classifier." https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c, 2018. [Online; accessed 24-May-2022].

[13] IBM, "What is the k-nearest neighbors algorithm?." https://www.ibm.com/topics/knn. [Online; accessed 24-May-2022].

[14] S. Ray, "Understanding support vector machine(svm) algorithm from examples (along with code)." https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. [Online; accessed 24-May-2022].