\documentclass[sigconf,authorversion,nonacm]{acmart}

\settopmatter{printacmref=false} % Removes citation information below abstract

\pagestyle{plain} % Removes running headers

\fancyfoot{} % Clears footer

\usepackage{graphicx} % Required for inserting images

\usepackage{wrapfig}

\usepackage{enumerate}

%Author Information

\title{A Guide to Principal Component Analysis (IAPT)}

\author{Matthias Bartolo\\ B.Sc. IT (Hons) Artificial Intelligence Student}

\email{matthias.bartolo.21@um.edu.mt}

\affiliation{%

\institution{University of Malta}

\city{Msida}

\country{Malta}

\postcode{MSD 2080}

}

\date{1st May 2023}

\begin{document}

\maketitle

\section{Introduction}

\subsection{Definition and Practical Application of PCA}

Principal Component Analysis (PCA) is an incredibly useful, and widely used multivariate algorithm in Machine Learning. Moreover, such algorithm is also extremely helpful in the analysis of huge datasets, whilst effectively undertaking Dimensionality Reduction and Feature Selection. Furthermore, PCA is used to ensure that data scientists may load and utilise large datasets on less powerful machines, which could not support the size of the full dataset. Additionally, PCA also provides cleaner data visualisation through the envisioning of the key data features in the full dataset, which hold the largest degree of information. [1-2]

\subsection{History}

PCA has a long, and illustrious history that goes back more than a century. The algorithm was pioneered by Karl Pearson, who in 1901 launched this system with the aim of undertaking data analysis and dimensionality reduction. The current PCA's design was first pioneered by Harold Hotelling in the 1930s, who led the way for the method to truly take shape. Hotelling was instrumental in formulating the concept of variance maximization, and the use of orthogonal projections to find the Principal Components. [1-2]

Further improvements to the PCA algorithm were developed in the 1960s, in part due to the emergence of Singular Value Decomposition (SVD), which offered an alternate method for calculating the eigenvalues and vectors, necessary to perform the PCA algorithm. The growing adoption of PCA at that time was largely triggered by the need for dimensionality reduction and the widespread growth of computers. Consequently, the method gained a lot of popularity in the 1970s and later on when data scientists and researchers fully comprehended the effectiveness of such technique in dealing with enormous and complex datasets which were becoming more and more prevalent in industries such as banking, engineering, and medicine. [1-2]

\subsection{Aims and Objectives}

Nevertheless, such algorithm’s behaviour may not always be comprehensible, thus triggering the need for the development of a visual tool, which allow users the possibility of visualising the algorithm's stages and data transformations, whilst offering a better understanding on the modified data. Consequently, the programmed solution also effectively portrays the PCA process as a simple convenient methodology which may be explained to students who have just completed a Linear Algebra or AI Numerical Methods course. Additionally, the developed Jupyter Notebook which outlines the aforementioned process, conveys to the students the necessary information to understand better such algorithm, whilst providing them with essential tools, to experiment and expand their knowledge. Furthermore, the notebook’s characteristics of being robust and responsive, allow students to interact with the visual plots through the plot’s minimising and maximising tools. The Jupyter Notebook which incorporates the developed solution, will be complemented with various famous datasets utilised by the machine learning community such as the Iris dataset, with the aim of making the students familiar with such datasets. One must note that the datasets were chosen for their distinct properties, to allow students to evaluate different experiments and infer new knowledge.

\subsection{Summary of Results}

The noteworthy key points attained from the creation of the Jupyter Notebook are listed hereunder:

\begin{enumerate}[(1)]

\item Principal Component Analysis is designed to be utilised on linearly separable data i.e., the variables in the dataset need to be linked together through a linear relationship. Consequently, Kernel PCA can be used to resolve the above-mentioned limitation [3].

\item The PCA algorithm is intended to be used on continuous data values. The approach taken to cater for discrete value features in the dataset, included changing the discrete values to continuous values, is through different encoding techniques. Moreover, such discrete columns which could also be discarded before the calculation of the PCA, thus resulting in more accurate data projections, at the cost of losing the discrete data columns.

\item There are different Libraries which may be used to implement PCA, for example the NumPy and the scikit-learn library. Furthermore, sometimes the NumPy library does not support the singular value decomposition of large datasets and would crash. On the other hand, the scikit-learn library enables the implementation of PCA without terminating abruptly as it utilises random singular value decomposition. Furthermore, Randomized SVD can approximate the whole SVD with a substantially lower computation cost by randomly selecting a fraction of the matrix's rows or columns [4].

\item It is imperative that the data which is fed to the PCA algorithm is normalized, so as to avoid the PCA from loading on the high variance data. [5]. Additionally, the created artefact utilises Z-Score Normalization.

\item Understanding the PCA algorithm mathematically can be quite a difficult task, as the process includes multiple mathematical calculations such as SVD or Covariance Matrix evaluation. Nevertheless, the developed artefact provides the need-to-know basis for such algorithm, in a well explained format.

\end{enumerate}

\section{Background and Methodology}

Mathematically, PCA enables the conversion of linear continuous data into a new coordinate system, characterized by new axis (Principal Components) which are ordered in accordance with the features in the new coordinate system. This enables that the best principal components are plotted on different dimensional graphs, thus presenting a satisfactory visualisation of a large dataset. Unfortunately, such method may have some minimal data reduction, however visualising an n dimensional feature dataset on a 3D plot is quite a benefit. The PCA's main characteristics of decreasing the dimensionality of data, whilst retaining salient information, lead to it being the most effectively ranked data analysis and machine learning technique [1-2].

It is worth pointing out that, the designed implementation provides an in-depth description of such algorithm, whilst categorizing the explanation in the following ordered sections:

\subsection{Libraries Utilised}

Figure 1 illustrates the various libraries which were used in the construction of the aforementioned notebook.

\begin{figure}[h]

\centering

\includegraphics[width=0.55\linewidth]{Figure1.png}

\caption{Libraries utilised.}

\label{fig:figure1}

\end{figure}

\subsection{Loading the Data}

A key step before the initiation of the PCA Algorithm, entails the selection of a relevant dataset which will be analysed by such algorithm. The designed implementation enables students interacting with the notebook, the choice to select any of the default datasets, and explore how the PCA algorithm will function on such datasets. Students are also given the option to load their preferred dataset. In addition, the selected default datasets are characterised by different attributes, so as to allow students in carrying out different experiments, and to facilitate a comparative analysis of the results obtained through varying the datasets.

\textbf{\\The following are the default datasets (Obtained from [6-12]):}

\begin{enumerate}[(1)]

\item \textbf{country\\_wise\\_latest.csv} - This dataset has a small Size, a large number of Features, and a few numbers of Discrete Columns.

\item \textbf{diabetes.csv} - This dataset has a small Size, a small number of Features, and no Discrete Columns.

\item \textbf{FIFA-2014.csv} - This dataset has a small Size, a small number of Features, and one Discrete Column.

\item \textbf{IRIS.csv} - This dataset has a small Size, a large number of Features, and one Discrete Column.

\item \textbf{Salary\\_Dataset\\_with\\_Extra\\_Features.csv} - This dataset has a large Size, a small number of Features, and a reasonable number of Discrete Columns.

\item \textbf{spotify.csv} - This dataset has a large Size, a large number of Features, and a reasonable number of Discrete Columns.

\item \textbf{wine-quality-white-and-red.csv} - This dataset has a large Size, a large number of Features, and one Discrete Column.

\end{enumerate}

The aforementioned functionality of loading the chosen dataset into a pandas data frame can be illustrated in Figure 2. The presented code snippet lists a Menu and depending on the choice to load a preferred dataset or load a default dataset, the user will be given a relevant message to input the file path or name respectively. In case that the user selects an invalid option, the program will continue to loop, until the user has successfully inputted a valid input. In case that the user chooses to load a default dataset, the user is presented with the list of default csv files present in the Datasets folder. Consequently, retrieving such dataset names is executed dynamically through the os.listdir function, thus enabling adding another default dataset to the Datasets folder relatively simple. On the other hand, if the user decides to load a preferred dataset, a relevant check is run to ascertain that the specified file exists. Finally, the specified file name is loaded through the pd.read\\_csv function, and stored in a pandas data frame. In case that the specified csv file has less than 3 columns, a Warning message is displayed to the user, advising him that the dataset will present Errors in the data visualisations later on. Additionally, calculating the PCA algorithm on a dataset which has less than three columns presents unsatisfactory results. Furthermore, applying the concept of dimensionality reduction on a dataset which already has reduced size, is quite redundant.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure2.png}

\caption{Code required for Loading the chosen dataset into a pandas data frame.}

\label{fig:figure2}

\end{figure}

\subsection{Dataset Feature Selection}

Another key step when performing a data analysis or a machine learning study, pertains to observing the type and number of different Genes/Features, which the dataset possesses. This step is highly critical, as sometimes processing a huge number of features in the dataset may cause memory allocation issues or prolong the processing time of algorithms. Consequently, in this stage students are given the option to choose which features to retain from the dataset, through the form of a user input menu. Additionally, in the eventuality that the student’s selection represents two columns or less, the program will automatically choose the first three columns which will be added to the filtered dataset. This was applied as a fail-safe measure, to ensure that the filtered dataset, would have enough features for visualisation in the upcoming sections.

The code snippet illustrated in Figure 3 portrays the aforementioned functionality, whereby the program first creates a new copy of original data frame, and then proceeds to loop through all of the new data frame’s columns. For each column iteration, the program probes the user with the choice of keeping the current column or discarding it. In case that the user opts to drop the current column, the pd.drop method is actioned in removing such feature from the new data frame. At the end of such snippet, a while loop is being used to ensure that the new data frame has at least three columns. Notably, for every iteration in the while loop, the pd.insert method is being used to add a column from the old data frame.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure3.png}

\caption{Code required for Feature Selection.}

\label{fig:figure3}

\end{figure}

\subsection{Dealing with Discrete Data}

As previously mentioned, PCA is designed to be utilised on continuous data [13]. This is a cardinal feature and dictates the need to transform discrete data into continuous data before using PCA on a dataset. This is critical, as discrete data lacks a continuous range of values and cannot be represented in the same way as continuous data for this cause.

There are various ways how discrete data can be Transformed/Encoded to continuous data, in order to be examined by the PCA. On the other hand, in case that students would prefer to discard such columns, and focus only on the continuous data, they could opt to remove such discrete columns from the data frame in the previous section. Additionally proper error checking was implemented, in case that the data frame does not include any discrete columns, thus ensuring the robustness of the developed solution.

\textbf{\\The following are different types of Encoders, which were implemented in the artefact:}

\begin{enumerate}[(1)]

\item \textbf{One-Hot Encoding}

\item \textbf{Label Encoding}

\item \textbf{Ordinal Encoding}

\item \textbf{Count Encoding}

\item \textbf{Word Embeddings Model}

\end{enumerate}

\subsubsection{One-Hot Encoding}

One-hot Encoding is a data preparation technique used to transform discrete variables into a format that enables the examination by machine learning algorithms. Consequently, this encoding algorithm works by creating a binary vector for each possible category in the data. Additionally, each binary vector would have a value of 1 or 0 symbolising the presence or absence of each category respectively. [14-16]

The encoding technique featured in the program, is quite explosive, as the number of different Genes/Features obtained after applying One-Hot encoding on a single column, will greatly increase the number of columns depending on the number of distinct features in each column. For an algorithm which aims to reduce dimensionality, such approach to turn discrete data into continuous data is quite inefficient, notwithstanding the increase in memory and time complexity.

One might think whether this binary vector can be transformed back to decimal. Note that such encoding algorithm exists and is known as Binary to Decimal Decoding. The aforementioned algorithm effectively transforms the binary vector back into a decimal value, thus reducing the size of the Genes/Features to their original number [17]. Essentially such encoding would take relatively more time whilst achieving the same results as Label Encoding or Ordinal Encoding.

In this section students are presented with an application of such encoding algorithm on the filtered data frame, and given a detailed explanation, justifying the inefficiency of the combined use of One-Hot Encoding with the PCA algorithm. As illustrated in the code snippet of Figure 4, the program loops through all the filtered data frame columns and proceeds to check whether the current column has an Object type i.e., is a discrete column. In case the current column has such type, then the pd.get\\_dummies function is applied on such column, and the result is stored inside a new data frame, whilst exiting the loop. Following this, the user is presented with a message displaying the difference in size between the encoded column and the original data frame.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure4.png}

\caption{Code required for applying One-Hot Encoding on the first Discrete Column.}

\label{fig:figure4}

\end{figure}

\subsubsection{Label Encoding}

Label Encoding is another data preparation technique which facilitates the transformation of discrete variables into a format that is easily readable by machine learning algorithms. Such encoder works by giving each distinct category a unique numeric value or code [14,15,18]. For instance, taking the list of categories [“hat”,”apple”,”cap”] will be encoded as [3,1,2] (as numeric values).

As can be depicted in code snippet of Figure 5, the program first creates a new copy of filtered data frame, and then proceeds to loop through all of the new data frame’s columns. For every column iteration, the program checks whether the current column has an Object type. In the affirmative, the pd.factorise function is used to apply Label Encoding on such column. Furthermore, for this encoding implementation the sort flag was set to True in the pd.factorise function, as to enable the transformed data to be assigned numerical values based on the sorted strings.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure5.png}

\caption{Code required for applying Label Encoding on the filtered data frame.}

\label{fig:figure5}

\end{figure}

\subsubsection{Ordinal Encoding}

A similar data preparation technique to Label Encoding is Ordinal Encoding. Such encoder works by giving each distinct category a unique numeric value or code, based on the order which the category appeared first [14,15,19]. For instance, taking the list of categories ["hat","apple","cap"] will be encoded as [1,2,3] (as numeric values, and encoded in the order which they appeared).

As can be seen in the code snippet of Figure 6, Ordinal Encoding is being implemented similarly to the Label Encoding implementation depicted in Figure 5. Notably the difference between both figures is clearly denoted by the pd.factorise function used to apply Ordinal Encoding, which has the sort flag applied to False. This was intentionally programmed in such manner so as to facilitate the assignment of numerical values based on the order in which the word appeared. Through the implementation of Label Encoding and Ordinal Encoding, which utilise the same function with some minor tweaks, students are given further opportunities to test out different encoding techniques, on the same dataset.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure6.png}

\caption{Code required for applying Ordinal Encoding on the filtered data frame.}

\label{fig:figure6}

\end{figure}

\subsubsection{Count Encoding}

Another data preparation technique is Count Encoding. This encoder works by encoding each distinct category, with the number of times such category appeared [14-16]. For instance, if the category "hat" appeared 5 times, then "hat" will be encoded by the number 5.

The code snippet seen in Figure 7, depicts the implementation of the aforementioned Count Encoding technique, which utilises the same logic of realising discrete columns as the previously mentioned encoding techniques. Furthermore, the value\\_counts function, is being used to count the number of times, each discrete value appears in the current discrete column. Afterwards, the pd.map function is being used to map the acquired array of discrete value frequencies to the current column’s discrete values, ultimately transforming the discrete values to the number of times which they have appeared in the discrete column.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure7.png}

\caption{Code required for applying Count Encoding on the filtered data frame.}

\label{fig:figure7}

\end{figure}

\subsubsection{Word Embeddings Model}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure8.png}

\caption{Code required for applying Word Embeddings Model on the filtered data frame.}

\label{fig:figure8}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure9.png}

\caption{Student Choice for the encoding technique to utilise.}

\label{fig:figure9}

\end{figure}

\subsection{Filtered Dataset Visualisations}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure10.png}

\caption{Student Choice for the features to utilise for visualisation.}

\label{fig:figure10}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure11.png}

\caption{Code required to visualise a 3D Plot.}

\label{fig:figure11}

\end{figure}

\subsection{Normalizing Data}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure12.png}

\caption{Z-Score Normalization Formula.}

\label{fig:figure12}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure13.png}

\caption{Standard Deviation Formula.}

\label{fig:figure13}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure14.png}

\caption{Z-Score Normalization Implementation.}

\label{fig:figure14}

\end{figure}

\subsection{Normalized Dataset Visualisations}

\subsection{loremIpsum}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure15.png}

\caption{Taking a small subset of the dataset.}

\label{fig:figure15}

\end{figure}

\subsection{Understanding PCA - SVD Approach}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure16.png}

\caption{SVD Calculation Formula.}

\label{fig:figure16}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure17.png}

\caption{Code required for PCA Calculation via SVD Approach.}

\label{fig:figure17}

\end{figure}

\subsection{Understanding PCA - Covariance Matrix Approach}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure18.png}

\caption{Covariance Matrix Calculation.}

\label{fig:figure18}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure19.png}

\caption{Code required for PCA Calculation via Covariance Matrix Approach.}

\label{fig:figure19}

\end{figure}

\subsection{A note on Calculating the Variance Ratio and Visualising ratio in Scree Plot}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure20.png}

\caption{Variance Ratio Calculation.}

\label{fig:figure20}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure21.png}

\caption{Code required for Variance Ratio Calculation.}

\label{fig:figure21}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure22.png}

\caption{Scree Plot example.}

\label{fig:figure22}

\end{figure}

\subsection{Comparisons Between Approaches}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure23.png}

\caption{2D Plot of PCA utilising SVD Approach.}

\label{fig:figure23}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure24.png}

\caption{2D Plot of PCA utilising Covariance Matrix Approach.}

\label{fig:figure24}

\end{figure}

\subsection{Working out PCA on the Entire Dataset}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure25.png}

\caption{PCA algorithm through the scikit-learn library.}

\label{fig:figure25}

\end{figure}

\subsection{PCA Visualisations}

\subsection{Conclusions and Limitations of PCA}

\section{Evaluation}

\begin{figure}[h]

\centering

\includegraphics[width=0.5\linewidth]{Figure26.png}

\caption{Showcasing the Dataset Selection Menu.}

\label{fig:figure26}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure27.png}

\caption{Displaying the Contents of the Requested Dataset.}

\label{fig:figure27}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=0.7\linewidth]{Figure28.png}

\caption{Showcasing the Feature Selection Menu.}

\label{fig:figure28}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure29.png}

\caption{Displaying the Filtered Dataset with the Requested Features.}

\label{fig:figure29}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=0.3\linewidth]{Figure30.png}

\caption{Displaying the result obtained after applying One-Hot Encoding on the first Discrete Column.}

\label{fig:figure30}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure31.png}

\caption{Displaying the comparison between the number of features between the Filtered Dataset and the One-Hot Encoded Data.}

\label{fig:figure31}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure32.png}

\caption{Displaying the result obtained after applying Label Encoding.}

\label{fig:figure32}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure33.png}

\caption{Displaying the result obtained after applying Ordinal Encoding.}

\label{fig:figure33}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure34.png}

\caption{Displaying the result obtained after applying Count Encoding.}

\label{fig:figure34}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure35.png}

\caption{Displaying the result obtained after utilising the Word Embeddings Model.}

\label{fig:figure35}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=0.3\linewidth]{Figure36.png}

\caption{Showcasing the Encoding Technique Selection Menu (In this case Ordinal Encoding was selected).}

\label{fig:figure36}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=0.7\linewidth]{Figure37.png}

\caption{Showcasing the Feature Visualisation Selection Menu (In this case Features 2,3 and 4 were chosen as the Features to be plotted on the respective axis).}

\label{fig:figure37}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure38.png}

\caption{Displaying the 3D Representation of the Filtered Dataset, with the chosen features.}

\label{fig:figure38}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure39.png}

\caption{Displaying the 2D Representation of the Filtered Dataset, with the chosen features.}

\label{fig:figure39}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=0.3\linewidth]{Figure40.png}

\caption{Displaying the Calculated Mean for each Column.}

\label{fig:figure40}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=0.4\linewidth]{Figure41.png}

\caption{Displaying the Calculated Standard Deviation for each Column.}

\label{fig:figure41}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure42.png}

\caption{Displaying the Normalized Data frame.}

\label{fig:figure42}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure43.png}

\caption{Displaying the 3D Representation of the Normalized Dataset, with the chosen features.}

\label{fig:figure43}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure44.png}

\caption{Displaying the 2D Representation of the Normalized Dataset, with the chosen features.}

\label{fig:figure44}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure45.png}

\caption{Displaying the resultant U Matrix obtained after the SVD Decomposition.}

\label{fig:figure45}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure46.png}

\caption{Displaying the resultant Sigma Matrix obtained after the SVD Decomposition.}

\label{fig:figure46}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure47.png}

\caption{Displaying the resultant V transpose Matrix obtained after the SVD Decomposition.}

\label{fig:figure47}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure48.png}

\caption{Displaying the Principal Components obtained through the Matrix multiplication of U by Sigma.}

\label{fig:figure48}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure49.png}

\caption{Displaying Scree Plot in the Understanding PCA - SVD Approach section.}

\label{fig:figure49}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure50.png}

\caption{Displaying the 3D Representation of the plotted Principal Components in the PCA - SVD Approach section.}

\label{fig:figure50}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure51.png}

\caption{Displaying the 2D Representation of the plotted Principal Components in the PCA - SVD Approach section.}

\label{fig:figure51}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure52.png}

\caption{Displaying the calculated Covariance Matrix.}

\label{fig:figure52}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure53.png}

\caption{Displaying the Principal Components obtained through the Matrix multiplication of the Normalized Data frame with the Sorted Eigenvectors of the Covariance Matrix.}

\label{fig:figure53}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure54.png}

\caption{Displaying Scree Plot in the Understanding PCA – Covariance Matrix Approach section.}

\label{fig:figure54}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure55.png}

\caption{Displaying the 3D Representation of the plotted Principal Components in the PCA – Covariance Matrix Approach section.}

\label{fig:figure55}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure56.png}

\caption{Displaying the 2D Representation of the plotted Principal Components in the PCA – Covariance Matrix Approach section.}

\label{fig:figure56}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure57.png}

\caption{Displaying Scree Plot pertaining to the Principal Components obtained by utilising the Entire Dataset.}

\label{fig:figure57}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure58.png}

\caption{Displaying the 3D Representation of the plotted Principal Components obtained by utilising the Entire Dataset.}

\label{fig:figure58}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure59.png}

\caption{Displaying the 2D Representation of the plotted Principal Components obtained by utilising the Entire Dataset.}

\label{fig:figure59}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Figure60.png}

\caption{Displaying the 1D Representation of the plotted Principal Components obtained by utilising the Entire Dataset.}

\label{fig:figure60}

\end{figure}

\section{Conclusion}

\section{Video Demonstration}

\section{References}

[1] S. Mishra et al., "Multivariate Statistical Data Analysis-Principal Component Analysis," Int. J. Livest. Res., vol. 1, pp. 1-6, 2017. [Online]. Available: https://www.researchgate.net/publication/316652806\\_Principal\\_Component\\_Analysis. [Accessed: 18-Apr-2023].

\noindent[2] D. Li and S. Liu, "4.2.3.1 Principal Component Analysis," in Water Quality Monitoring and Management: Basis, Technology and Case Studies, 1st ed., S. K. Gupta and R. Kumar, Eds. Amsterdam, Netherlands: Elsevier, 2019. [Online]. Available: https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/principal-component-analysis. [Accessed: 18-Apr-2023].

\noindent[3] N. B. Subramanian, "Types of PCA", aiaspirant.com. [Online]. Available: https://aiaspirant.com/types-of-pca/. [Accessed: 18-Apr-2023].

\noindent[4] N. Halko, P. G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” arXiv preprint arXiv:0909.4061, 2009. [Online]. Available: https://arxiv.org/abs/0909.4061. [Accessed: 18-Apr-2023].

\noindent[5] Stack Exchange. "Why do we need to normalize data before Principal Component Analysis (PCA)?", Cross Validated, May 26, 2014. [Online]. Available: https://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-principal-component-analysis-pca. [Accessed: 18-Apr-2023].

\noindent[6] DEVAKUMAR K. P., "COVID-19 Dataset", Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/imdevskp/corona-virus-report?select=country\\_wise\\_latest.csv. [Accessed: 18-Apr-2023].

\noindent[7] UCI MACHINE LEARNING, "Pima Indians Diabetes Database", Kaggle, 2016. [Online]. Available: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. [Accessed: 18-Apr-2023].

\noindent[8] S. BANERJEE, "FIFA - Football World Cup Dataset", Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/iamsouravbanerjee/fifa-football-world-cup-dataset?select=FIFA+-+2014.csv. [Accessed: 18-Apr-2023].

\noindent[9] MATHNERD, "Iris Flower Dataset", Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/arshid/iris-flower-dataset. [Accessed: 18-Apr-2023].

\noindent[10] S. BANERJEE, "Software Industry Salary Dataset - 2022", Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/iamsouravbanerjee/software-professional-salaries-2022. [Accessed: 18-Apr-2023].

\noindent[11] R. Holbrook and A. Cook, "Principal Component Analysis, spotify.csv", Kaggle. [Online]. Available: https://www.kaggle.com/code/ryanholbrook/principal-component-analysis/data?select=spotify.csv. [Accessed: 18-Apr-2023].

\noindent[12] RUTHGN, "Wine Quality Data Set (Red \& White Wine)", Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/ruthgn/wine-quality-data-set-red-white-wine. [Accessed: 18-Apr-2023].

\noindent[13] V. Karthik, "PCA for categorical features", Stack Overflow, Dec. 2016. [Online]. Available: https://stackoverflow.com/questions/40795141/pca-for-categorical-features\#:~:text=PCA\%20is\%20designed\%20for\%20continuous,yes\%2C\%20you\%20can\%20use\%20PCA. [Accessed: 18-Apr-2023].

\noindent[14] Datagy. "Pandas get\\_dummies (One-Hot Encoding) Explained," Datagy.io, Feb. 2021. [Online]. Available: https://datagy.io/pandas-get-dummies/. [Accessed: 18-Apr-2023].

\noindent[15] DataCamp. "Dealing with Categorical Data". DataCamp, 2021. [Online]. Available: https://www.datacamp.com/tutorial/categorical-data. [Accessed: 18-Apr-2023].

\noindent[16] B. Roy, "All about Categorical Variable Encoding," Towards Data Science, Jul. 2, 2019. [Online]. Available: https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02. [Accessed: 18-Apr-2023].

\noindent[17] T. Crosley, "What is the binary to decimal decoder?", Quora, May 8, 2018. [Online]. Available: https://www.quora.com/What-is-the-binary-to-decimal-decoder. [Accessed: 18-Apr-2023].

\noindent[18] Pandas. "pandas.factorize()". pandas 1.4.0 documentation, Jan. 07, 2022. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.factorize.html. [Accessed: 18-Apr-2023].

\noindent[19] J. Brownlee, "One-Hot Encoding for Categorical Data," Machine Learning Mastery, Aug. 17, 2020. [Online]. Available: https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/. [Accessed: 18-Apr-2023].

\noindent[20] Vatsal, "Word2Vec Explained", Towards Data Science, Jul. 29, 2021. [Online]. Available: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71. [Accessed: 18-Apr-2023].

\noindent[21] R. Sharma. "What is Normalization in Data Mining and How to Do It?", UpGrad, Sep. 22, 2022. [Online]. Available: https://www.upgrad.com/blog/normalization-in-data-mining/\#:~:text=Project\%20Ideas\%20\%26\%20Topics-,Z\%2DScore\%20Normalization,up\%20to\%20\%2B3\%20standard\%20deviation. [Accessed: 18-Apr-2023].

\noindent[22] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular Value Decomposition and Principal Component Analysis," in Learning from Data: Concepts, Theory, and Methods, vol. 2, Springer, Boston, MA, 2007, pp. 151-176, doi: 10.1007/0-306-47815-3\\_5. [Online]. Available: https://www.researchgate.net/publication/2167923\\_Singular\\_Value\\_Decomposition\\_and\\_Principal\\_Component\\_Analysis. [Accessed: 18-Apr-2023].

\noindent[23] CUEMATH, "Covariance Matrix", CUEMATH. [Online]. Available: https://www.cuemath.com/algebra/covariance-matrix/. [Accessed: 18-Apr-2023].

\noindent[24] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," in The Data Deluge: Can Libraries Cope with E-Science? Proceedings of a Conference Held at the Royal Society, London, UK, 4-5 November 2004, vol. 463, Royal Society Publishing, 2016, pp. 21-36. doi: 10.1098/rsta.2015.0202.[Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202. [Accessed: 18-Apr-2023].

\noindent[25] K. Guillaumier, "Linear Algebra in Data Science and PCA"

\noindent[26] S. Mangale, "Scree Plot," Medium, Aug. 28, 2020. [Online]. Available: https://sanchitamangale12.medium.com/scree-plot-733ed72c8608. [Accessed: 18-Apr-2023].

\noindent[27] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular Value Decomposition and Principal Component Analysis," in Learning from Data: Concepts, Theory, and Methods, vol. 2, Springer, Boston, MA, 2007, pp. 151-176, doi: 10.1007/0-306-47815-3\\_5. [Online]. Available: https://www.researchgate.net/publication/2167923\\_Singular\\_Value\\_Decomposition\\_and\\_Principal\\_Component\\_Analysis. [Accessed: 18-Apr-2023].

\noindent[28] Scikit-learn, “sklearn.decomposition.PCA", scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. [Accessed: 18-Apr-2023].

\noindent[29] M. Kumar, "Memory error in NumPy SVD," in IEEE, 2014. [Online]. Available: https://stackoverflow.com/questions/21180298/memory-error-in-numpy-svd. [Accessed: 18-Apr-2023].

\noindent[30] N. B. Subramanian, "Types of PCA", aiaspirant.com. [Online]. Available: https://aiaspirant.com/types-of-pca/. [Accessed: 18-Apr-2023].

\end{document}