

Assessment Cover Page

Module Title:	Data Exploration & Preparation
Assessment Title:	CA1 Project
Lecturer Name:	Dr Muhammad Iqbal
Student Full Names:	Jefferson de Oliveira Lima Breno Silva Brito
Student Number:	2020373 2023462
Assessment Due Date:	03/12/2023
Date of Submission:	22/11/2023

Words count: 2875

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Table Content

Introduction: Exploring Crime Trends in Ireland	3
Data Overview	3
Data Cleaning	3
Descriptive Statistics	4
Categorical Variables:	4
Discrete Variables:	4
Bar chart to Visualize the total occurrences for each type of offence	5
Statistical Parameters (mean, median, minimum, maximum, and standard deviation)...	6
Mean Values:	7
Median Values:	7
Minimum Values:	7
Maximum Values:	7
Standard Deviation Values:.....	7
Min-Max Normalization, Z-score Standardization and Robust Scalar.....	8
Min-Max Normalization:.....	8
Z-score Standardization:	8
Robust Scalar:	8
Line, Scatter and Heatmaps.....	9
Line Plot:	9
Scatter Plot:	9
Heatmap Plot:	9
Data Exploratory Analysis	10
Data Filtering (EDA Step):.....	10
Aggregation (EDA Step):.....	10
Visualization (EDA Step):	10
Dummy Encoding.....	11
Create Dummy Variables:	12
Combine Dummy Variables with the Dataset:	12
View Column Names:.....	12
Principal Component Analysis (PCA)	12
Dimensionality Reduction	12
Conclusion	13
References.....	13
GitHub Link	13

Introduction: Exploring Crime Trends in Ireland

This research does a thorough inquiry to examine Ireland's criminal justice system in detail. It does this by analysing crime statistics in detail and identifying small patterns that highlight the complexity of criminal behaviour. We want to understand the complex relationship that exists between many social situations and the various forms of criminal activity. We believe that a comprehensive understanding of this relationship is essential to the development of successful policies and the general well-being of society.

Using a dataset, this investigation takes a comprehensive approach to exploring Ireland's historical criminal history. The analysis goes beyond simple numerical calculations in an effort to provide a clear picture of the complex nature of criminal activity. Our analyse, which makes use of R's features, involves developing plots, strategically applying filters depending on crime kinds and locations, and designing visualisations. This gives us knowledge of the geographic distribution of criminal episodes and allows us to discover common crimes as well as regional variances.

Data Overview

Our dataset provides a detailed look into crimes that were reported in Ireland from 2003 to 2022, which is helpful in attempting to comprehend the patterns of crime in Ireland. This collection of data provides a thorough analysis of criminal activity broken down by type, location, and law enforcement station.

A lot of information have been recorded in each entry of the dataset, such as the year the offence occurred, the station code, the Garda station identity, and the particular kind of offence. We find, for example, a wide range of criminal incidents when we review entries from any particular Garda Station.

The dataset classifies different types of crimes into 15 categories, which include "Offences against government, justice procedures, and organisation of crime", "Attempts/threats to murder, assaults, harassments, and related offences.", and so on.

Additionally, the dataset's temporal covers from the year 2003 to 2022, providing a thorough understanding of crime trends over time. With the use of visualisation and filtering techniques, we are getting ready to delve into the world of data exploration. Our goal is to uncover meaningful patterns that will further improve our comprehension of crime in Ireland's diverse environments.

Data Cleaning

The lack of missing values in our dataset is, in fact, a good place to start when it comes to cleaning of data. This opportune event simplifies our analytical procedure by removing the

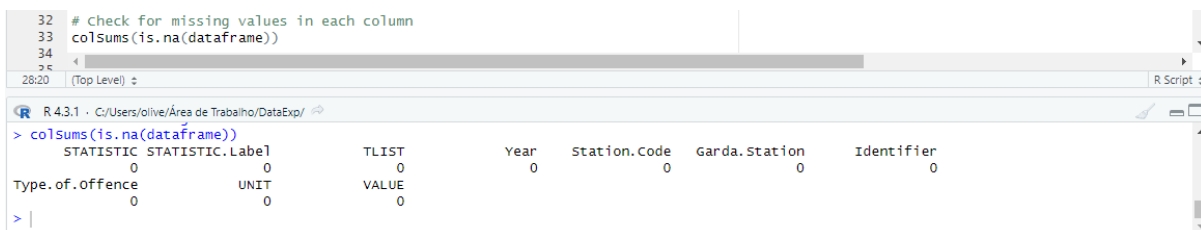
requirement for human value insertion. We ensure that our dataset is complete by carefully checking for missing values, which gives us confidence in the accuracy of the analyses we will perform later.

Although this component of the dataset is clean, it is important to recognise that missing values may have an impact on the accuracy of results. In our instance, the lack of missing values highlights the completeness and resilience of the dataset, putting us in a good position as we investigate crime trends in Ireland.

This guarantee not only indicates a solid basis for generating significant insights from the dataset, but it also improves the data cleaning procedure. We are able to explore the complexities of the data once missing values have been resolved and eliminated, concentrating on identifying patterns and trends that lead to a thorough understanding of crime occurrences in various Ireland regions and categories.

Following the code that checks for missing values in our dataset:

```
32 # Check for missing values in each column
33 colSums(is.na(dataframe))
34
```



STATISTIC	STATISTIC.Label	TLIST	Year	Station.Code	Garda.Station	Identifier
0	0	0	0	0	0	0
Type.of.offence	UNIT	VALUE				
0	0	0				

Descriptive Statistics

After analysing our dataset, it is clear that all of the variables are discrete or categorical, and there are no continuous variables to be found. This categorization is essential for adjusting our analytical strategies to the characteristics of every variable.

Categorical Variables:

STATISTIC, STATISTIC Label, Garda Station, Type of Offence and Unit are essential for labelling and segmenting statistical and geographic differences in the dataset. Pie charts and bar charts are two popular visualisation techniques that are useful for displaying how crime incidences are distributed throughout various statistics categories or Garda stations.

Discrete Variables:

TList, Year, Station Code, Identifier and Value. They each represent unique and different values. Interestingly, we classify "VALUE" as a discrete variable, which represents the total number of criminal incidents. The frequency and distribution of criminal episodes across time, station codes, identifiers, and the discrete count of incidents are all captured via histograms or bar charts.

By recognising "VALUE" as a discrete variable, we guarantee a proper representation of its attributes in our analyses. In addition to providing the framework for researching crime patterns in Ireland, this classification guarantees that our statistical techniques are in perfect alignment with the basic features of every variable, allowing precise and sensitive data analysis.

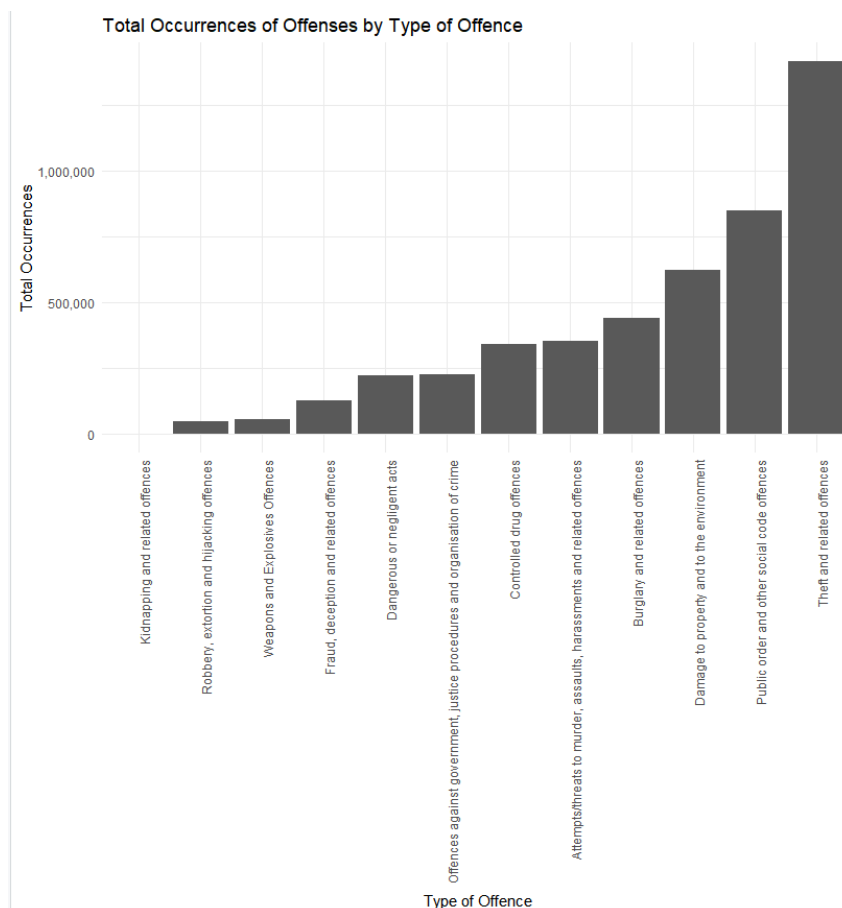
Bar chart to Visualize the total occurrences for each type of offence

```
# Create a bar chart to visualize the total occurrences for each type of offence

# Aggregate the VALUE variable by summing it up for each Identifier
aggregated_data <- dataframe %>%
  group_by(Type.of.offence) %>%
  summarise(Total_Value = sum(VALUE))

# Create a bar chart
ggplot(aggregated_data, aes(x = reorder(Type.of.offence, Total_Value), y = Total_Value)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Occurrences of Offences by Type of Offence",
       x = "Type of Offence",
       y = "Total Occurrences") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_y_continuous(labels = scales::comma)
```

This code generates a bar chart to visually represent the total occurrences for each type of offense in a given dataset. The "Value" variable for each distinct offence type is added to determine the overall count of occurrences. The data is initially aggregated by classifying



it according to the "Type.of.Offence" field. Using the ggplot2 package, a bar chart is then produced using the aggregated data that has resulted. Rearranged according to the total number of occurrences, the various offence kinds are shown on the x-axis. Each offence type's overall frequency is shown on the y-axis. In order to improve readability and clarity, the chart is further improved with a title, axis labels, and aesthetic adjustments such rotating the x-axis labels for improved visibility. The labels on the y-axis are formatted using the scale_y_continuous function.

When combined with our dataset, the provided code creates a bar chart that shows the total number of incidents divided into categories according to the kind of offence. The x-axis lists the various offence types from our dataset, and the y-axis shows the overall number of incidents over a period of years. Additionally, the graphic shows that, with over 1,800,000 instances, "Theft and related offences" predominates the Irish crime scene. On the other hand, crimes like "Robbery, extortion, and hijacking offences" have a far lower incidence rate—about 50,000 cases. Surprisingly, there are no documented cases of "Kidnapping and related offences" in the dataset. This graphic illustration offers a brief yet comprehensive summary of the frequency of different offences, giving a visual history of crime distribution in Ireland.

Statistical Parameters (mean, median, minimum, maximum, and standard deviation)

```
37 # Select numerical columns
38 numerical_columns <- subset(dataframe, select = sapply(dataframe, is.numeric))
39
40 # Calculate mean for each numerical column
41 mean_values <- sapply(numerical_columns, function(x) mean(x, na.rm = TRUE))
42 mean_values
43
44 # Calculate median for each numerical column
45 median_values <- sapply(numerical_columns, function(x) median(x, na.rm = TRUE))
46 median_values
47
48 # Calculate minimum for each numerical column
49 min_values <- sapply(numerical_columns, function(x) min(x, na.rm = TRUE))
50 min_values
51
52 # Calculate maximum for each numerical column
53 max_values <- sapply(numerical_columns, function(x) max(x, na.rm = TRUE))
54 max_values
55
56 # Calculate standard deviation for each numerical column
57 sd_values <- sapply(numerical_columns, function(x) sd(x, na.rm = TRUE))
58 sd_values
59
```

This code conducts a numerical analysis on the selected columns of the dataset, computing various descriptive statistics for each numeric variable.

Mean Values:

TLIST: Approximately 2012.5
Year: Approximately 2012.5
Station.Code: Around 35875.05
Identifier: Approximately 8.58
VALUE: Approximately 34.78

Median Values:

TLIST: 2012.5
Year: 2012.5
Station.Code: 34104.5
Identifier: 8.5
VALUE: 4.0

Minimum Values:

TLIST: 2003
Year: 2003
Station.Code: 11101
Identifier: 3
VALUE: 0

Maximum Values:

TLIST: 2022
Year: 2022
Station.Code: 512015
Identifier: 15
VALUE: 6523

Standard Deviation Values:

TLIST: Approximately 5.77
Year: Approximately 5.77
Station.Code: Around 25368.45
Identifier: Approximately 3.59
VALUE: Approximately 143.67

These results provide a comprehensive overview of the central tendency, spread, and variability within each numeric column, aiding in understanding the distribution and characteristics of the dataset's numerical variables.

Min-Max Normalization, Z-score Standardization and Robust Scalar

```
63 # Min-Max Normalization
64 min_max <- function(x) {
65   (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
66 }
67
68 normalized_data_min_max <- sapply(numerical_columns, min_max)
69 normalized_data_min_max
70
71 # Z-score Standardization
72 z_score <- function(x) {
73   (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
74 }
75
76 standardized_data_z_score <- sapply(numerical_columns, z_score)
77 standardized_data_z_score
78
79 # Robust Scalar
80 robust_scalar <- function(x) {
81   (x - median(x, na.rm = TRUE)) / IQR(x, na.rm = TRUE)
82 }
83
84 robust_scaled_data <- sapply(numerical_columns, robust_scalar)
85 robust_scaled_data
```

Three different feature scaling techniques—Min-Max Normalisation, Z-score Standardisation, and Robust Scaling—are used in this code to improve the dataset's interpretability and robustness.

Min-Max Normalization:

To ensure that all variables' values are rescaled inside the interval of 0 to 1, the `min_max` function is defined.

The dataset is subjected to Min-Max Normalisation using `sapply` on the numerical columns, producing `normalized_data_min_max`. When interpretability within a particular scale is important, this transformation is especially helpful.

Z-score Standardization:

Each variable's distribution is standardised by the `z_score` function, which aligns them to have a mean of 0 and a standard deviation of 1.

`Standardized_data_z_score` is the outcome of applying `sapply` on the numerical columns. This standardisation makes the dataset more consistent and comprehensible by facilitating the comparison of variables with various scales or units.

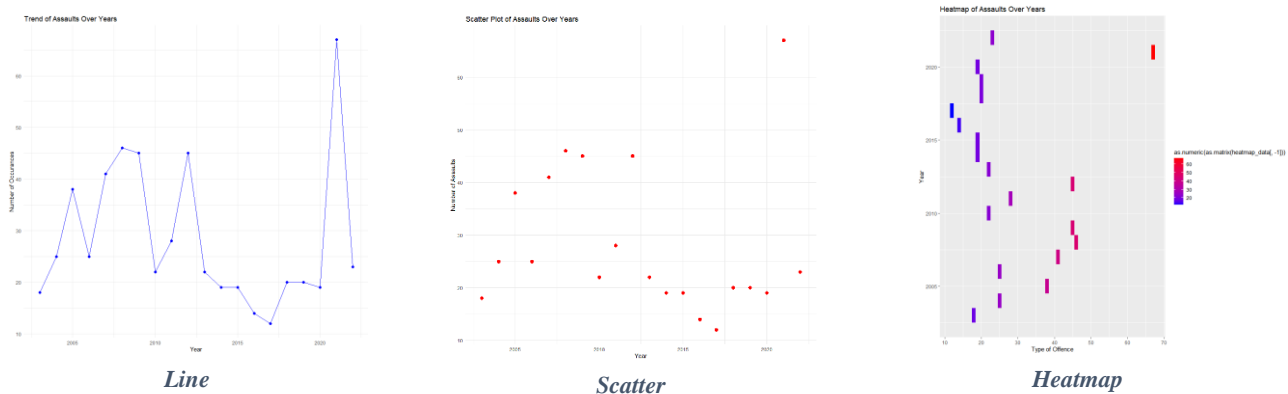
Robust Scalar:

The `robust_scalar` function takes the interquartile range (IQR) and median into account when designing it for robust scaling. When compared to Z-score Standardisation and Min-Max Normalisation, this method is less vulnerable to outliers.

`Robust_scaled_data` is the result of the numerical columns being transformed by the `sapply` function. This scaling technique ensures a more reliable representation of variable distributions, which is especially beneficial when there are outliers.

By using these feature scaling techniques, the dataset becomes more suitable for a wider range of analysis, providing a more standardised and normalised base for statistical tests and modelling projects. Clearer insights into the relative significance and behaviour of various variables within the dataset are made possible by these adjustments.

Line, Scatter and Heatmaps



The code performs exploratory data analysis and visualisation in order to identify and present patterns in the number of assaults that have occurred at a certain Garda station over time. Three distinct plot types are included in the analysis:

Line Plot:

The filtered data subset is created by selecting records for a specific police station (Garda.Station) and a particular type of offense (Type.of.Offence).

A line plot is generated using ggplot to showcase the trend of assaults over the years. Blue lines represent the trajectory of occurrences, with blue points marking individual data points.

Scatter Plot:

Another scatter plot is created to emphasize the distribution and variation in assault occurrences over the years. Red points represent the number of assaults each year, providing a clearer view of the data's dispersion.

Heatmap Plot:

The data is pivoted to wide format, creating heatmap_data for efficient visualization.

A heatmap is generated using ggplot, representing the intensity of assault occurrences for each year and type of offense. The colour gradient from blue to red indicates the

magnitude of incidents, with blue representing lower values and red indicating higher occurrences.

These graphics give a thorough overview of the dataset for the designated Garda station and offence type while also offering significant insights into the historical patterns and distribution of attacks. Combining line, scatter, and heatmap plots allows for a more thorough understanding of the underlying trends and addresses several facets of data analysis.

Data Exploratory Analysis

```
121
122 # Filter the data for "Theft and related offences" in Dublin
123 filtered_data <- subset(dataframe, Type.of.Offence == "Theft and related offences" & grepl("dublin", tolower(G
124
125 # Aggregate occurrences by year
126 aggregated_data <- aggregate(VALUE ~ Year, data = filtered_data, sum)
127
128 # Create a line plot
129 ggplot(aggregated_data, aes(x = Year, y = VALUE)) +
130   geom_line() +
131   labs(title = "Total Occurrences of Theft and Related Offences in Dublin",
132        x = "Year",
133        y = "Total Occurrences") +
134   theme_minimal()
135
```

Data Filtering (EDA Step):

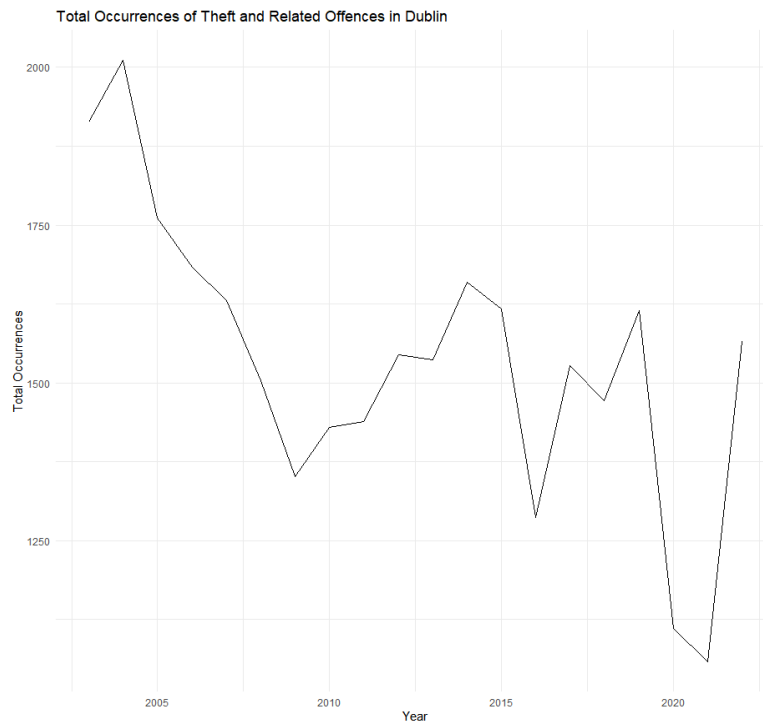
The initial step involves filtering the dataset to focus on a specific subset of interest. In this case, the data is narrowed down to records related to "Theft and related offences" in Garda Stations located in Dublin. This filtering is a crucial aspect of exploratory data analysis (EDA) as it allows for a targeted examination of a particular category within the dataset.

Aggregation (EDA Step):

Following data filtering, the next EDA step is aggregation. The occurrences of theft-related offenses are aggregated by year. Aggregation is a common technique in EDA to derive insights and obtain a summarized view of the data. By aggregating the data, patterns and trends over time become more apparent.

Visualization (EDA Step):

The final step includes the creation of a visual representation of the aggregated data. In this case, a line plot is generated. Visualization is a key component of EDA as it allows for the intuitive interpretation of data patterns. The line plot visually illustrates the trend of total occurrences of theft and related offenses over the years in Dublin, enabling a quick and clear understanding of the data.



Plot from the previous code provided

To summarise, the code provided aligns to the principles of exploratory data analysis by filtering through relevant data, collecting information to emphasise patterns, and plotting the results to visually display the results. Together, these actions help in the investigation and comprehension of patterns found in the dataset.

Dummy Encoding

```
R 4.3.1 - C:/Users/olive/Área de Trabalho/DataExp/
> # Create dummy variables using model.matrix
> dummy_variables <- model.matrix(~ 'type.of.offence' - 1, data = dataframe)
>
> # Combine the dummy variables with the dataset
> dataframe <- cbind(dataframe, dummy_variables)
>
> # View column names
> names(dataframe)
[1] "STATISTIC"
[2] "STATISTIC.Label"
[3] "TLIST"
[4] "Year"
[5] "Station.Code"
[6] "Garda.Station"
[7] "Identifier"
[8] "type.of.offence"
[9] "UNIT"
[10] "VALUE"
[11] "type.of.offenceAttempts/threats to murder, assaults, harassments and related offences"
[12] "type.of.offenceBurglary and related offences"
[13] "type.of.offenceControlled drug offences"
[14] "type.of.offenceDamage to property and to the environment"
[15] "type.of.offenceDangerous or negligent acts"
[16] "type.of.offenceFraud, deception and related offences"
[17] "type.of.offenceKidnapping and related offences"
[18] "type.of.offenceOffences against government, justice procedures and organisation of crime"
[19] "type.of.offencePublic order and other social code offences"
[20] "type.of.offenceRobbery, extortion and hijacking offences"
[21] "type.of.offenceTheft and related offences"
[22] "type.of.offenceWeapons and Explosives offences"
>
```

Create Dummy Variables:

The code utilizes the `model.matrix` function to create dummy variables based on the `Type.of.Offence` column in the dataset.

Explanation: Dummy variables are binary (0 or 1) representations of categorical variables. In this case, the `Type.of.Offence` column, which likely contains categorical data, is converted into a set of binary columns (dummy variables). Each unique category in `Type.of.Offence` gets its own column, and for each observation, the corresponding column is set to 1 if the observation belongs to that category and 0 otherwise.

Combine Dummy Variables with the Dataset:

The dummy variables created in the previous step are added to the original dataset.

Explanation: By combining the dummy variables with the original dataset, the analysis can incorporate the binary representations of the categorical variable. This is often done when working with machine learning models or statistical analyses that require numerical input.

View Column Names:

The code concludes by displaying the column names of the modified dataset.

Explanation: This step is likely included for verification purposes. It allows the user to inspect the names of the columns in the dataset to confirm that the dummy variables have been successfully added. Checking column names is a good practice to ensure that the data manipulation steps were executed as intended.

To summarise, this code creates dummy variables from a categorical variable (`Type.of.Offence`), integrates them back into the dataset, and then shows the names of the columns in the updated dataset. Categorical data are frequently prepared for analysis using this procedure, especially when statistical modelling or machine learning are involved.

Principal Component Analysis (PCA)

In data analysis, principal component analysis, or PCA, is a statistical method for reducing dimensions. The process converts a high-dimensional dataset into a new coordinate system and highlights the directions (principal components) where the data exhibits the greatest variation. PCA minimises the dimensionality of the dataset by keeping the principal components that represent the largest variation, making analysis easier to understand while maintaining crucial information. This technique is widely used in many different industries to help with the processing of signals, machine learning modelling, and pattern recognition as well as noise reduction and effective feature selection. (*Brennan Whitfield, Mar 29, 2023*)

Dimensionality Reduction

Dimensionality reduction reduces the amount of characteristics or variables in a dataset while keeping important information, which helps to simplify complicated datasets. In machine learning applications, where high-dimensional data can result in higher processing

demands, this approach is crucial for improving computational efficiency. Models become less vulnerable to overfitting and more computationally simple when the number of dimensions is decreased. Effective data visualisation is made possible by dimensionality reduction, which also makes it possible to explore patterns in two or three dimensions. In addition, it improves model interpretability, helps reduce noise by eliminating unnecessary characteristics, and helps build stronger, more broadly applicable models. All things considered, dimensionality reduction is a useful method for raising the effectiveness, interpretability, and generalizability of studies and models across a range of disciplines. (Jason Brownlee, June 30, 2020)

Conclusion

To sum up, the investigation of Ireland's criminal data has been a complex undertaking that has made use of advanced data analytic methods to uncover complex patterns and trends. This study explored into criminal behaviour by utilising the powers of R programming, going beyond simple quantitative measurements. The lack of missing values in the dataset made data cleaning easier and guaranteed the validity of our findings. Descriptive statistics provided light on the features of the dataset and offered insightful information about the distribution and central tendencies of different types of criminal activity. Subsequently, the discernment of discrete, continuous, and categorical variables, in conjunction with perceptive visualisations, establishes the groundwork for more profound analysis. The use of methods like Principal Component Analysis is indicative of a persistent effort to manage dataset dimensionality while extracting valuable information. This research adds to the body of knowledge in data science and provides insightful viewpoints for guiding evidence-based choices in the areas of policy creation and crime prevention.

References

- Brennan Whitfield (2023). A Step-by-Step Explanation of Principal Component Analysis (PCA) <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>
- Jason Brownlee (2020). Introduction to Dimensionality Reduction for Machine Learning <https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/>

GitHub Link

<https://github.com/JeffOlima/DataExp>