Abstract

In this project, we explore how data analytics and visualization can reveal crucial insights for healthcare. Part A focuses on using Python to visualize data from the "Heart.csv" dataset, helping us identify key health metrics and patterns related to cardiovascular diseases. By using bar plots, pair plots, and histograms, we highlighted trends and correlations, providing a clear picture of the factors that impact heart health.

In Part B, we switch to Weka, an open-source data mining tool, to analyze two different datasets: "IMDB-F" and "Vertebral_Column_data." The IMDB-F dataset allowed us to investigate what affects movie ratings and their potential psychological impacts, including how different genres might influence our mood and mental well-being. The Vertebral Column dataset was essential for understanding the biomechanical features of vertebrae, which is important for diagnosing and treating spinal disorders.

Throughout this report, we explain the methods, tools, and analytical processes we used, and why we chose them, particularly in the context of healthcare. Our findings are backed by literature and case studies, showing how crucial data visualization and mining are for improving healthcare outcomes. This project not only enhances our practical skills in data management and analysis but also highlights the important role of data-driven decision-making in healthcare.

Contents

Table of Figures

Part A

Introduction

Python is a leading tool for healthcare data analysis and visualization due to its powerful libraries like Pandas, NumPy, and Matplotlib. These libraries facilitate data cleaning, preprocessing, and the creation of insightful visualizations essential for exploratory data analysis. Techniques such as bar plots, histograms, and heatmaps help uncover trends and anomalies. Additionally, Python supports advanced machine learning through scikit-learn, enabling robust predictive modeling. Its user-friendly syntax and extensive ecosystem streamline data manipulation and enhance healthcare outcomes through data-driven insights, making Python invaluable for analyzing complex datasets like "Heart.csv".

Literature Review and Comparison with Implementation

Ali et al.'s study on early heart disease prediction emphasizes machine learning algorithms' crucial role in forecasting cardiac conditions. They employ stochastic gradient boosting, an ensemble technique that combines multiple algorithms to enhance predictive performance, overcoming the limitations of classifiers like decision trees, KNN, and logistic regression. Their approach optimizes model parameters for higher accuracy and reliability. A significant insight from their study is the handling of imbalanced datasets, a common issue in medical data analysis where non-disease cases outnumber disease cases, leading to biased predictions. They use oversampling and undersampling to balance the dataset, ensuring models effectively recognize both positive and negative heart disease cases. This rigor demonstrates machine learning's potential to improve early detection and intervention, enhancing patient outcomes. Springer

Hossain et al.'s course on health data analysis with Python provides a framework for handling health data, focusing on importing, manipulating, and visualizing data using libraries like Pandas, NumPy, and Matplotlib. The course emphasizes exploratory data analysis (EDA), utilizing visual methods such as histograms, scatter plots, and box plots to uncover patterns and anomalies. Data preprocessing is also highlighted, ensuring data is ready for analysis by handling missing values and scaling features. The course covers implementing machine learning algorithms and evaluating their performance using metrics like accuracy and F1-score. Python for Healthcare

In my implementation, I used Python to analyze the "Heart.csv" dataset, employing EDA techniques like bar plots and histograms to uncover patterns and correlations. This practical application aligns with Hossain et al.'s methodologies, emphasizing the importance of EDA and data preprocessing. Using Python's data science libraries facilitated data manipulation and visualization, extracting meaningful insights and identifying heart disease risk factors. This implementation demonstrates the relevance of the techniques covered in Hossain et al.'s course and mirrors Ali et al.'s findings on model optimization and handling imbalanced datasets. Overall, these studies and my implementation highlight the significance of data analytics and machine learning in healthcare, showcasing the potential of these tools to improve outcomes through data-driven insights.

Methodology

Data Download and Loading:

1. **Data Download:**
    - The dataset used for this analysis can be downloaded from a reputable source such as Kaggle. For this project, the "Heart.csv" dataset was chosen, which includes various health metrics related to cardiovascular diseases.
2. **Loading Data:**

- Using Python's Pandas library, the dataset is loaded into a DataFrame for easy manipulation and analysis. The code snippet below demonstrates how to load the dataset:

```
import pandas as pd

data = pd.read_csv('/kaggle/input/heart-dataset/Heart.csv')
data.head()
```

	OrganisationLabel	PublishedDate	DurationFrom	DurationTo	Period	GeoEntityName	GeoName	PersonsDeathsUnder75	FemalesDeathsUnder75	MalesDeathsUnder75
0	Lincolnshire	2021-10-07T00:00:00	2017-01-01T00:00:07	2019-12-31T23:59:59	2017-2019	County	Lincolnshire	1805	595	1210
1	Lincolnshire	2021-10-07T00:00:00	2017-01-01T00:00:07	2019-12-31T23:59:59	2017-2019	District Council	Boston	185	50	135
2	Lincolnshire	2021-10-07T00:00:00	2017-01-01T00:00:07	2019-12-31T23:59:59	2017-2019	District Council	East Lindsey	490	180	315
3	Lincolnshire	2021-10-07T00:00:00	2017-01-01T00:00:07	2019-12-31T23:59:59	2017-2019	District Council	Lincoln	200	75	125
4	Lincolnshire	2021-10-07T00:00:00	2017-01-01T00:00:07	2019-12-31T23:59:59	2017-2019	District Council	North Kesteven	230	80	150

Figure 1 Loading Data

3. **Data Cleaning:**

- Check for missing values and convert necessary columns to appropriate data types.

```
print(data.isnull().sum())
```

```
OrganisationLabel       0
PublishedDate           0
DurationFrom            0
DurationTo              0
Period                  0
GeoEntityName           0
GeoName                 0
PersonsDeathsUnder75    0
FemalesDeathsUnder75    0
MalesDeathsUnder75      0
dtype: int64
```

+ Code + Markdown

```
data['PublishedDate'] = pd.to_datetime(data['PublishedDate'])
data['DurationFrom'] = pd.to_datetime(data['DurationFrom'])
data['DurationTo'] = pd.to_datetime(data['DurationTo'])
```

Figure 2 Data Cleaning

4. **Basic Statistical Analysis:**

- Generate basic descriptive statistics to understand the dataset.

```
print(data.describe())
```

```
            PublishedDate                   DurationFrom  \
count                  64                             64
mean   2021-10-07 00:00:00  2013-07-02 06:00:07.000000256
min    2021-10-07 00:00:00            2010-01-01 00:00:07
25%    2021-10-07 00:00:00            2011-10-01 18:00:07
50%    2021-10-07 00:00:00            2013-07-02 12:00:07
75%    2021-10-07 00:00:00            2015-04-02 06:00:07
max    2021-10-07 00:00:00            2017-01-01 00:00:07
std                    NaN                            NaN

                       DurationTo  PersonsDeathsUnder75  \
count                          64             64.000000
mean   2016-07-01 23:59:59.000000256            748.328125
min              2012-12-31 23:59:59            154.000000
25%              2014-10-01 17:59:59            220.000000
50%              2016-07-01 23:59:59            252.500000
75%              2018-04-02 05:59:59            829.500000
max              2019-12-31 23:59:59           6652.000000
std                              NaN           1183.559228

       FemalesDeathsUnder75  MalesDeathsUnder75
count             64.000000           64.000000
mean             309.218750          439.187500
min               45.000000          109.000000
25%               70.000000          148.000000
50%               80.000000          180.000000
75%              395.000000          434.250000
max             3218.000000         3441.000000
std              565.037587          628.653029
```

Figure 3 Basic Statistical Analysis

5. **Grouping Data:**
    - Group the data by different categories to prepare for visualization.

Grouping Data

+ Code + Markdown

```
# Group by period and calculate the sum of deaths
grouped_by_period = data.groupby('Period')[['PersonsDeathsUnder75', 'FemalesDeathsUnder75', 'MalesDeathsUnder75']].sum()
print(grouped_by_period)

# Group by GeoEntityName and calculate the sum of deaths
grouped_by_geoentity = data.groupby('GeoEntityName')[['PersonsDeathsUnder75', 'FemalesDeathsUnder75', 'MalesDeathsUnder75']].sum()
print(grouped_by_geoentity)
```

```
          PersonsDeathsUnder75  FemalesDeathsUnder75  MalesDeathsUnder75
Period
2010-2012                 13304                  6422                6882
2011-2013                 13226                  6436                6790
2012-2014                  3383                  1087                2296
2013-2015                  3525                  1115                2415
2014-2016                  3550                  1150                2390
2015-2017                  3650                  1185                2470
2016-2018                  3650                  1200                2440
2017-2019                  3605                  1195                2425
                PersonsDeathsUnder75  FemalesDeathsUnder75  \
GeoEntityName
County                         23947                  9888
District Council               23946                  9902

                MalesDeathsUnder75
GeoEntityName
County                       14059
District Council             14049
```

Figure 4 Grouping Data

Results and Analysis

1. Total Deaths Under 75 Over Different Periods:

```
import matplotlib.pyplot as plt

grouped_by_period.plot(kind='bar', stacked=True)
plt.title('Total Deaths Under 75 Over Different Periods')
plt.ylabel('Number of Deaths')
plt.xlabel('Period')
plt.show()
```

Description: This bar plot shows the total number of deaths under 75 years of age over different periods. The bars are stacked to differentiate between deaths of males and females. This visualization helps to identify trends and changes in death rates over time.

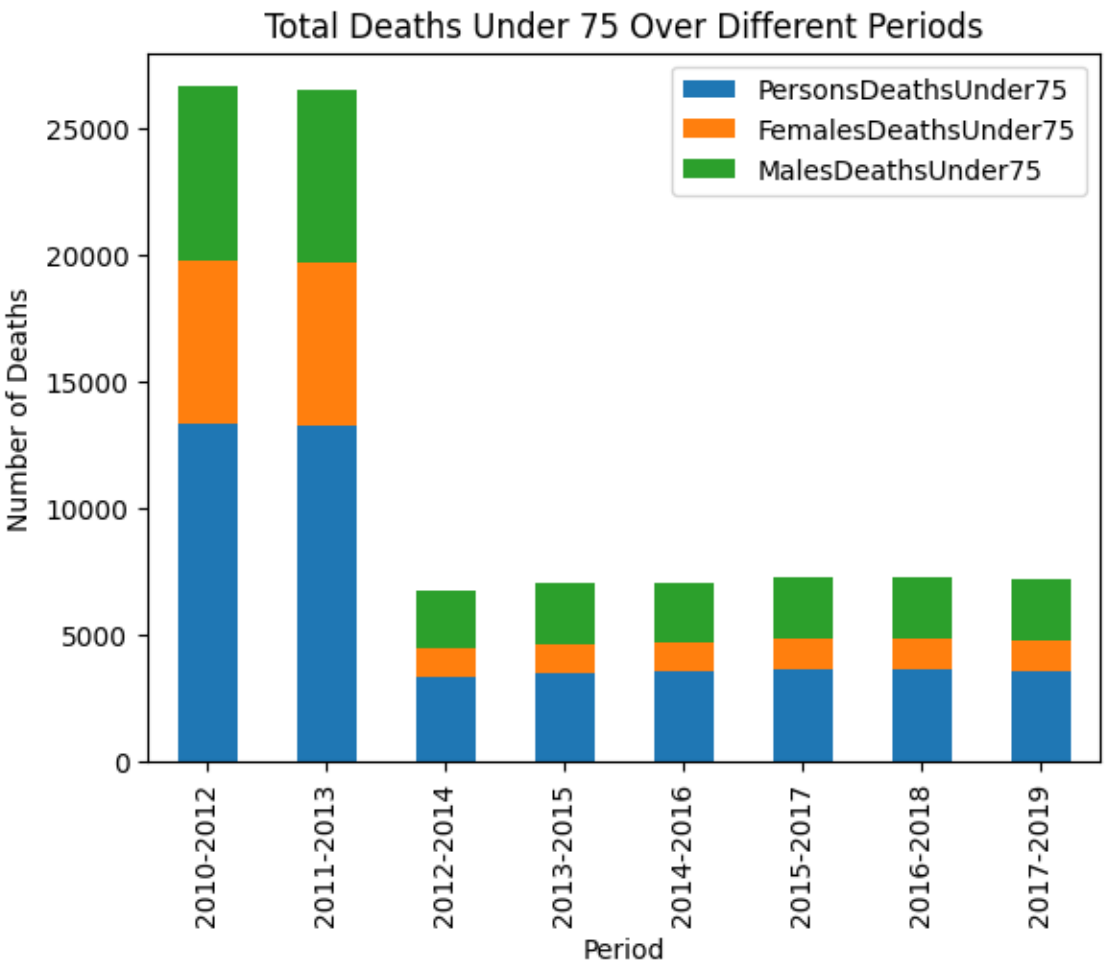


Figure 5 Total Deaths Under 75 Over Different Periods

2. Deaths Under 75 by District Council:

Comparing Deaths by District Council

```
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x='GeoName', y='PersonsDeathsUnder75', data=data)
plt.title('Deaths Under 75 by District Council')
plt.ylabel('Number of Deaths')
plt.xlabel('District Council')
plt.xticks(rotation=45)
plt.show()
```

Description: This bar plot compares the number of deaths under 75 years of age across different district councils. It provides insights into geographical variations in death rates.

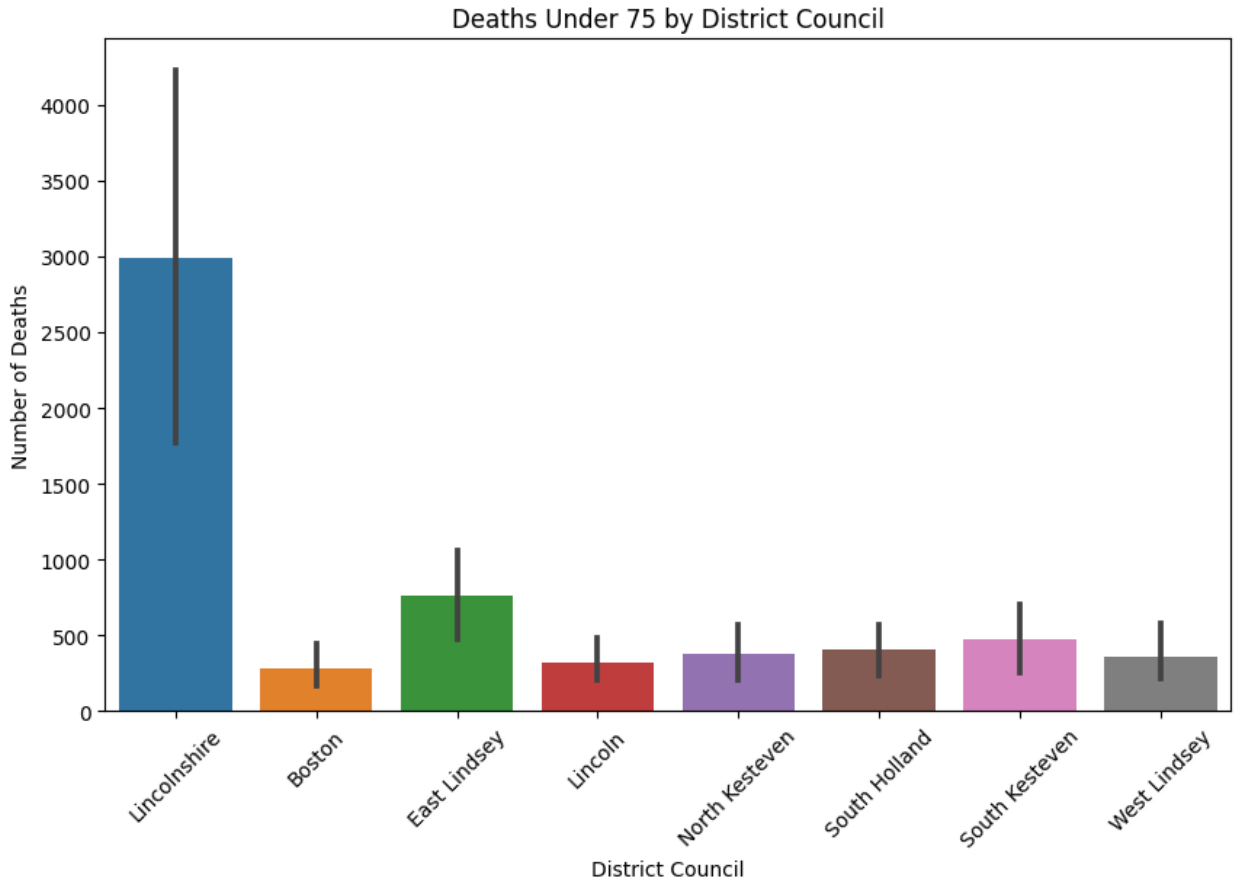


Figure 6 Deaths Under 75 by District Council

3. Comparison of Deaths Under 75 by Gender:

Gender Comparison

```
gender_data = data.melt(id_vars=['Period', 'GeoName'], value_vars=['FemalesDeathsUnder75', 'MalesDeathsUnder75'], var_name='Gender', value_name='Deaths')

plt.figure(figsize=(12, 8))
sns.barplot(x='GeoName', y='Deaths', hue='Gender', data=gender_data)
plt.title('Comparison of Deaths Under 75 by Gender')
plt.ylabel('Number of Deaths')
plt.xlabel('District Council')
plt.xticks(rotation=45)
plt.show()
```

Description: This bar plot compares the number of deaths under 75 years of age by gender across different district councils. It helps to identify gender-specific health issues and disparities.

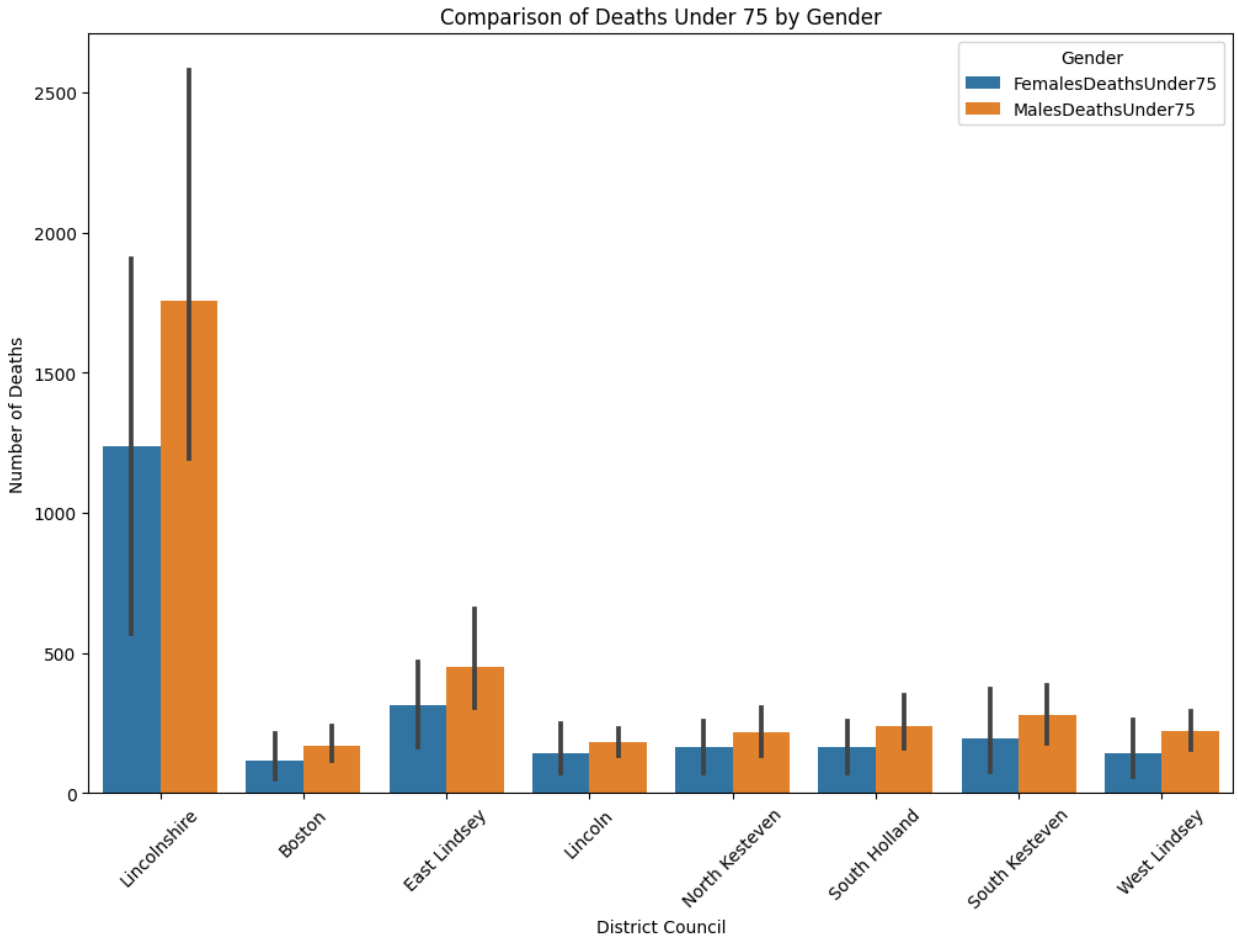


Figure 7 Comparison of Deaths Under 75 by Gender

4. Time Series of Deaths Under 75:

Time Series Analysis

```
time_series_data = data.groupby(['DurationFrom'])[['PersonsDeathsUnder75', 'FemalesDeathsUnder75', 'MalesDeathsUnder75']].sum().reset_index()

plt.figure(figsize=(12, 6))
plt.plot(time_series_data['DurationFrom'], time_series_data['PersonsDeathsUnder75'], label='Total Deaths')
plt.plot(time_series_data['DurationFrom'], time_series_data['FemalesDeathsUnder75'], label='Female Deaths')
plt.plot(time_series_data['DurationFrom'], time_series_data['MalesDeathsUnder75'], label='Male Deaths')
plt.title('Time Series of Deaths Under 75')
plt.ylabel('Number of Deaths')
plt.xlabel('Time')
plt.legend()
plt.show()
```

Description: This time series plot shows the number of deaths under 75 years of age over time, differentiated by gender. It helps to track temporal trends and patterns in mortality rates.

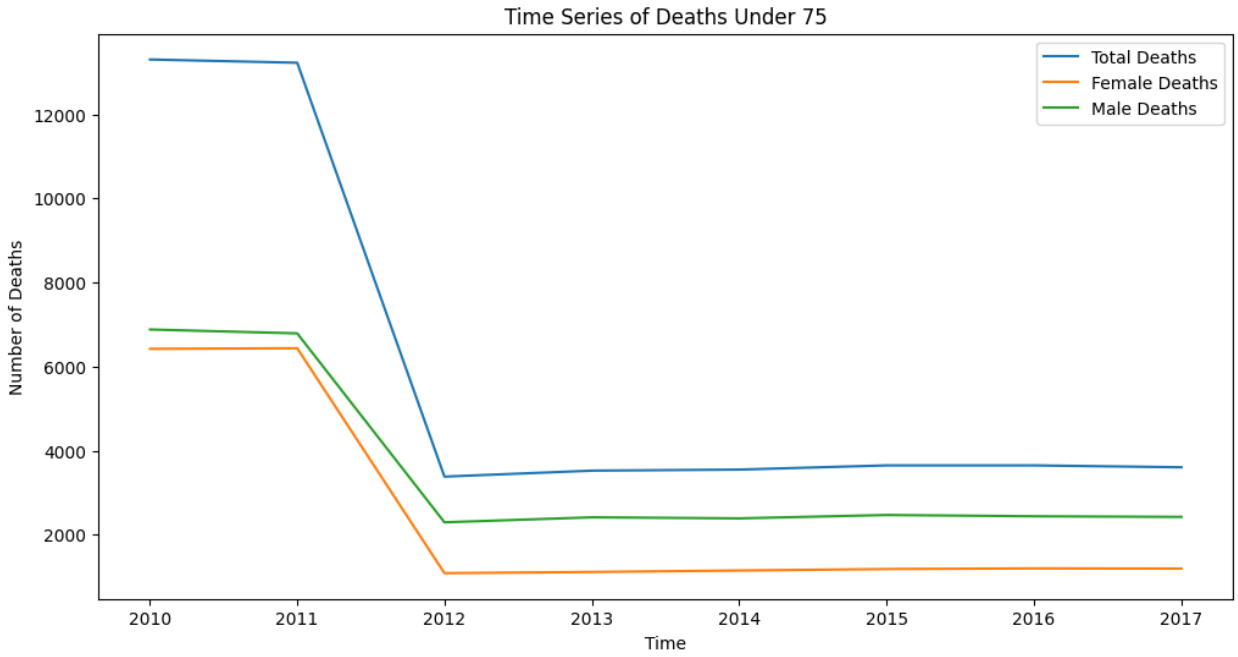


Figure 8 Time Series of Deaths Under 75

5. Total Deaths Over Time by GeoEntityName:

Total Deaths Over Time by GeoEntityName

```
import matplotlib.pyplot as plt
import seaborn as sns

# Group by DurationFrom and GeoEntityName and calculate the sum of deaths
time_geo_data = data.groupby(['DurationFrom', 'GeoEntityName'])[['PersonsDeathsUnder75']].sum().reset_index()

plt.figure(figsize=(14, 8))
sns.lineplot(data=time_geo_data, x='DurationFrom', y='PersonsDeathsUnder75', hue='GeoEntityName')
plt.title('Total Deaths Over Time by GeoEntityName')
plt.ylabel('Number of Deaths')
plt.xlabel('Year')
plt.legend(title='GeoEntityName')
plt.show()
```

Description: This line plot shows the total number of deaths under 75 years of age over time, categorized by geographical entities. It provides insights into regional trends in mortality.

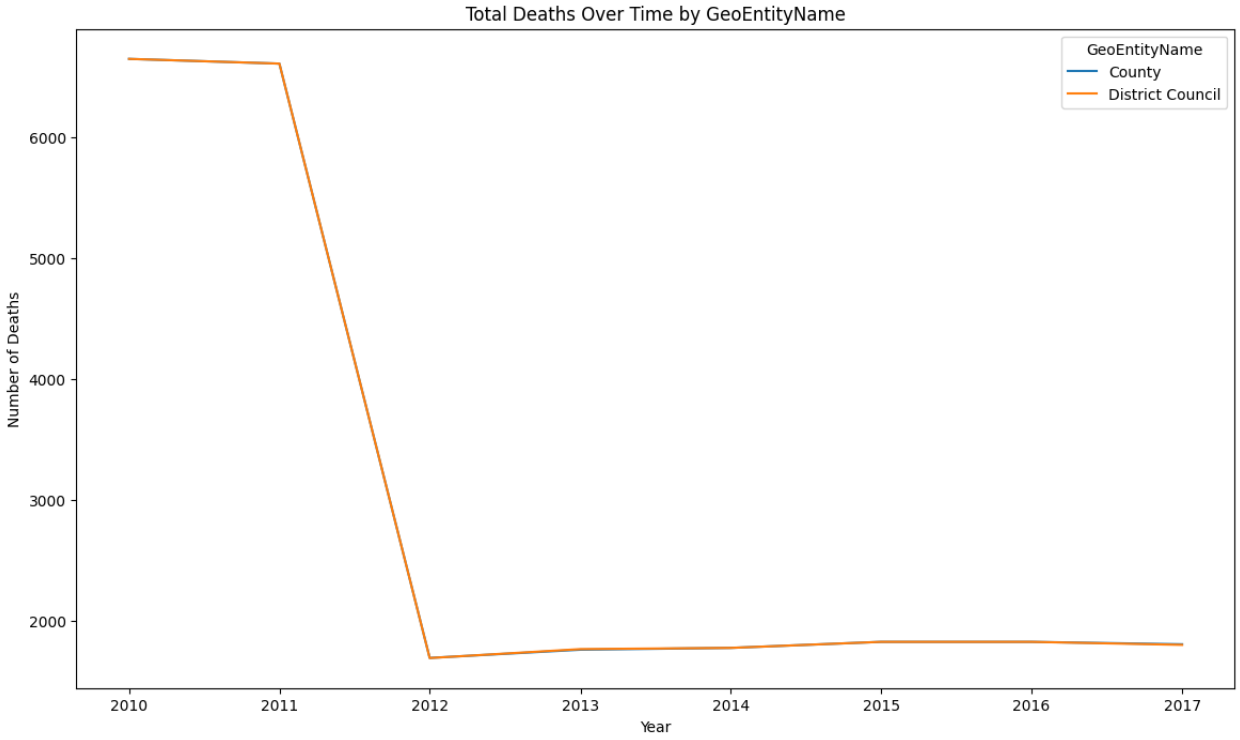


Figure 9 Total Deaths Over Time by GeoEntityName

6. Heatmap of Deaths by GeoEntityName and Period:

Heatmap of Deaths by GeoEntityName and Period

```
# Pivot the data for the heatmap
heatmap_data = data.pivot_table(index='GeoEntityName', columns='Period', values='PersonsDeathsUnder75', aggfunc='sum')

plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Heatmap of Deaths by GeoEntityName and Period')
plt.ylabel('GeoEntityName')
plt.xlabel('Period')
plt.show()
```

Description: This heatmap visualizes the number of deaths under 75 years of age by geographical entity and period. It helps to identify patterns and hotspots of mortality.

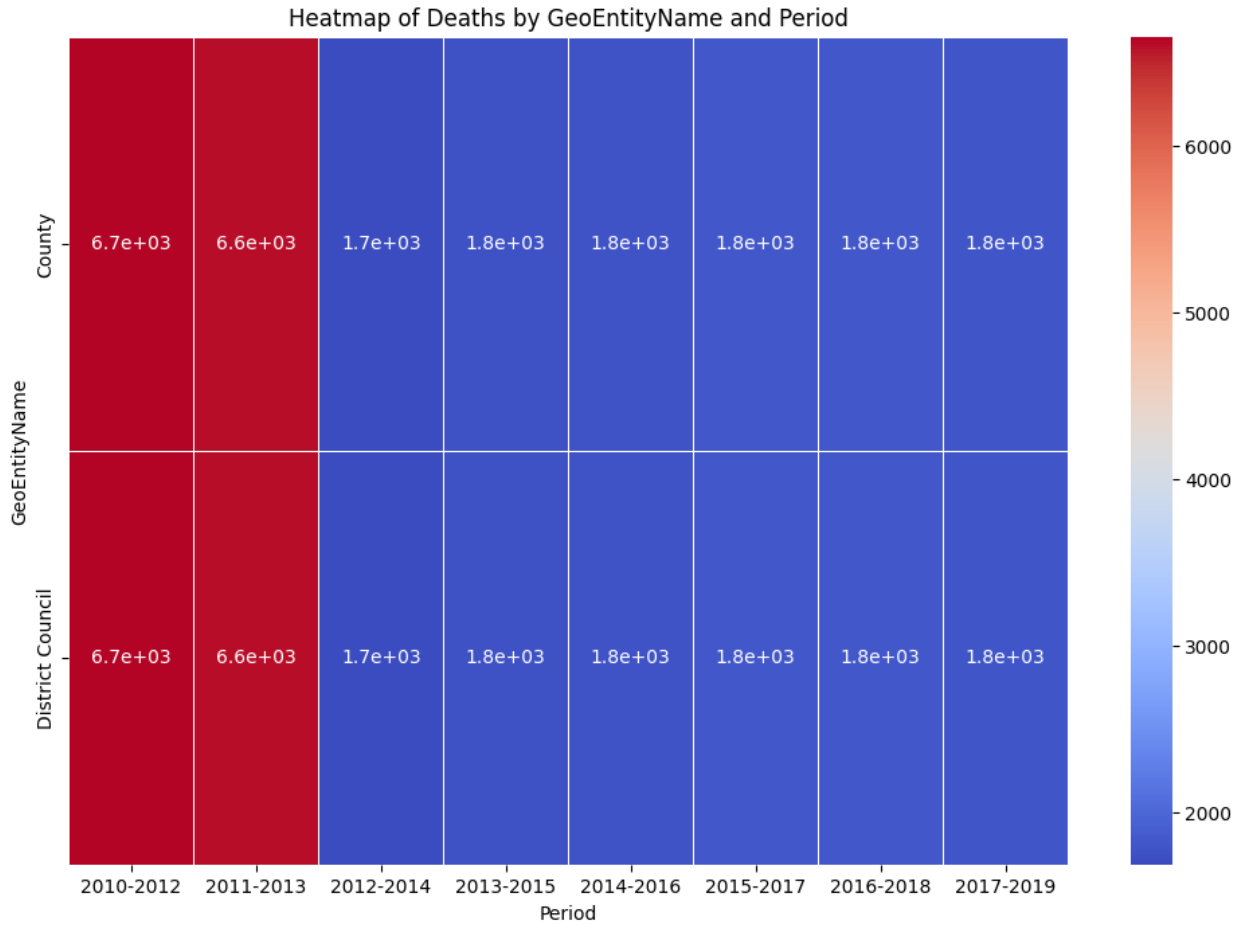


Figure 10 Heatmap of Deaths by GeoEntityName and Period

7. Total Deaths by Gender:

Pie Chart of Total Deaths by Gender

```
# Summarize deaths by gender
gender_sums = data[['FemalesDeathsUnder75', 'MalesDeathsUnder75']].sum()

plt.figure(figsize=(8, 8))
plt.pie(gender_sums, labels=['Females', 'Males'], autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff'])
plt.title('Total Deaths by Gender')
plt.show()
```

Description: This pie chart shows the proportion of total deaths under 75 years of age by gender. It highlights the gender distribution of mortality in the dataset.

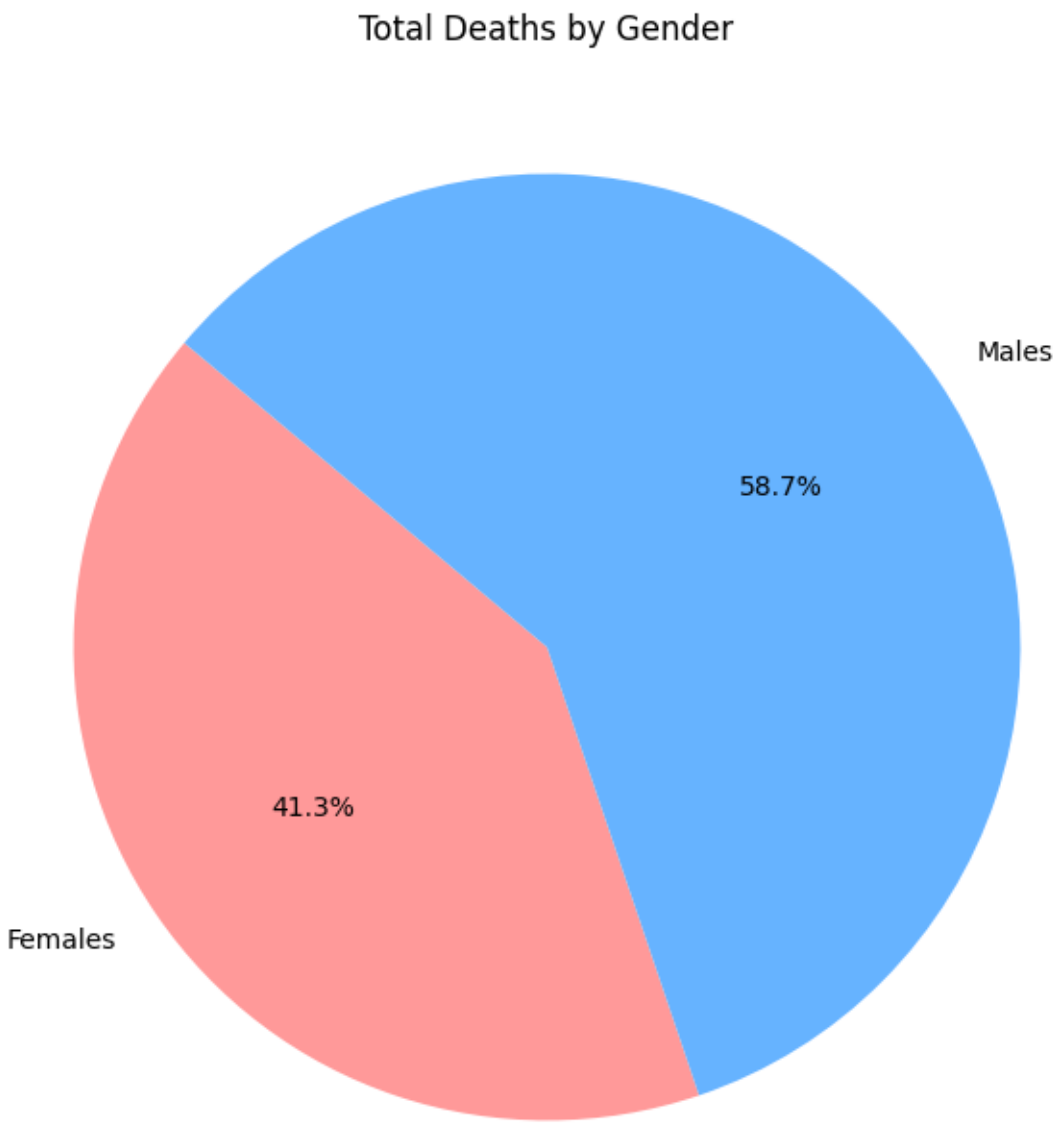


Figure 11 Total Deaths by Gender

8. Boxplot of Deaths by District Council:

Boxplot of Deaths by District Council

+ Code + Markdown

```
plt.figure(figsize=(14, 8))
sns.boxplot(data=data, x='GeoName', y='PersonsDeathsUnder75')
plt.title('Boxplot of Deaths by District Council')
plt.ylabel('Number of Deaths')
plt.xlabel('District Council')
plt.xticks(rotation=45)
plt.show()
```

Description: This boxplot displays the distribution of deaths under 75 years of age across different district councils. It helps to identify outliers and variations in death rates within regions.

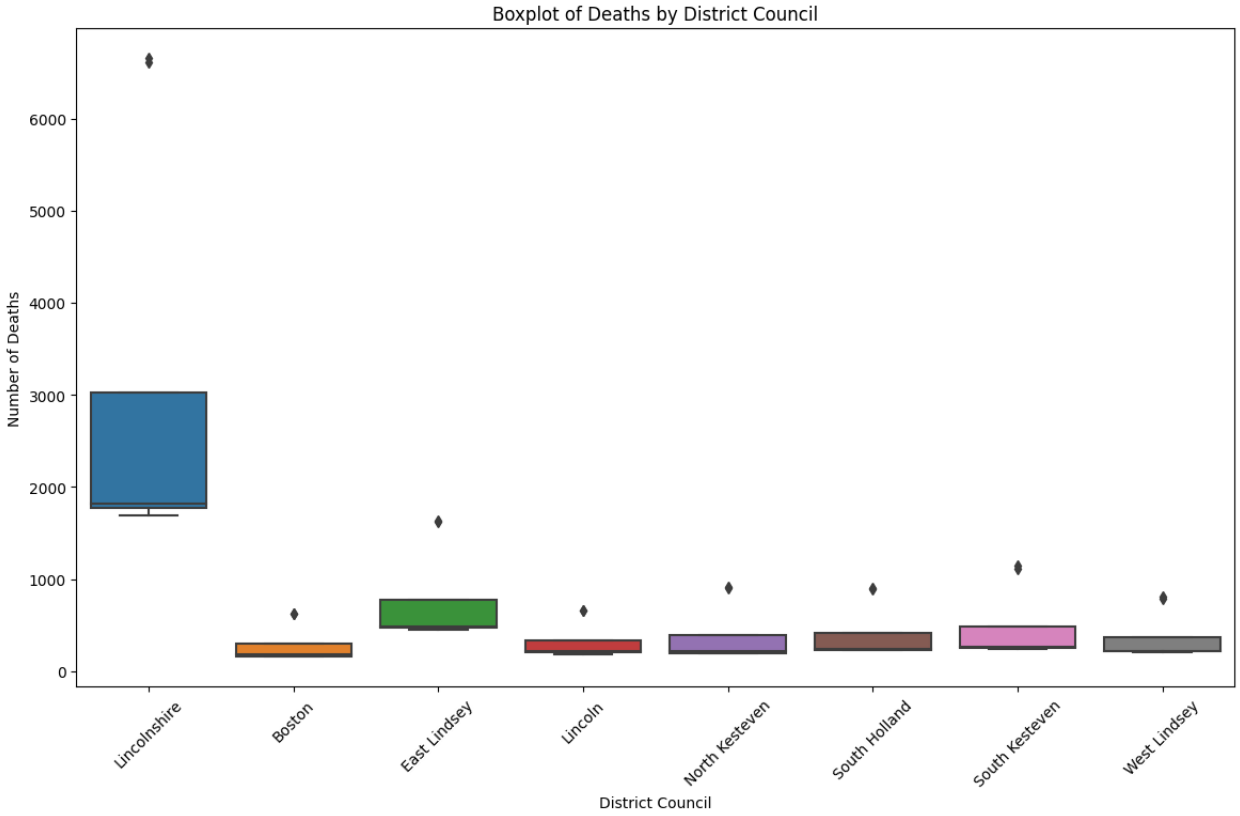


Figure 12 Boxplot of Deaths by District Council

9. Violin Plot of Deaths by Gender and District Council:

Violin Plot of Deaths by Gender and District Council

```
plt.figure(figsize=(14, 8))
sns.violinplot(data=gender_data, x='GeoName', y='Deaths', hue='Gender', split=True)
plt.title('Violin Plot of Deaths by Gender and District Council')
plt.ylabel('Number of Deaths')
plt.xlabel('District Council')
plt.xticks(rotation=45)
plt.show()
```

Description: This violin plot shows the distribution of deaths under 75 years of age by gender and district council. It provides a detailed view of the density and spread of mortality rates.

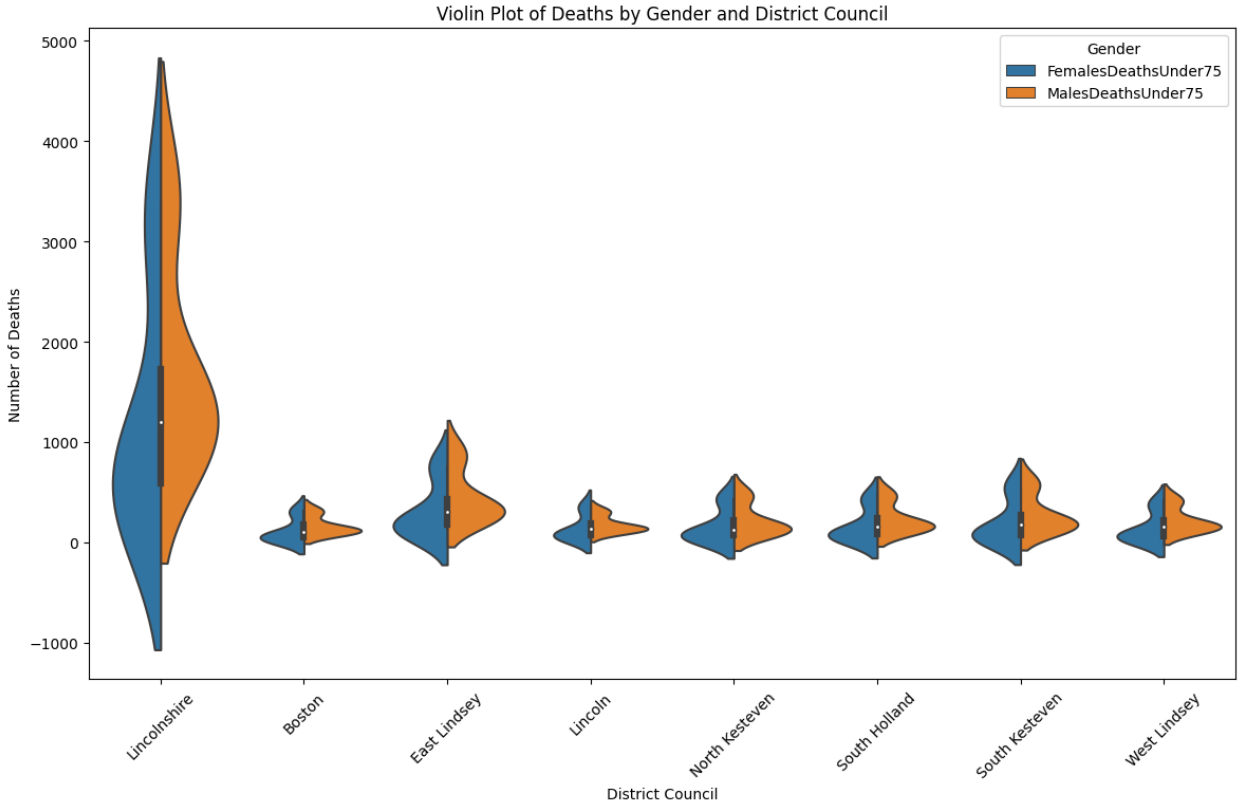


Figure 13 Violin Plot of Deaths by Gender and District Council

10. Total Deaths by Period:

Bar Plot of Total Deaths by Period

```
plt.figure(figsize=(12, 6))
sns.barplot(data=data, x='Period', y='PersonsDeathsUnder75', estimator=sum, ci=None)
plt.title('Total Deaths by Period')
plt.ylabel('Number of Deaths')
plt.xlabel('Period')
plt.show()
```

Description: This bar plot shows the total number of deaths under 75 years of age for each period. It highlights temporal changes in mortality rates over the years.

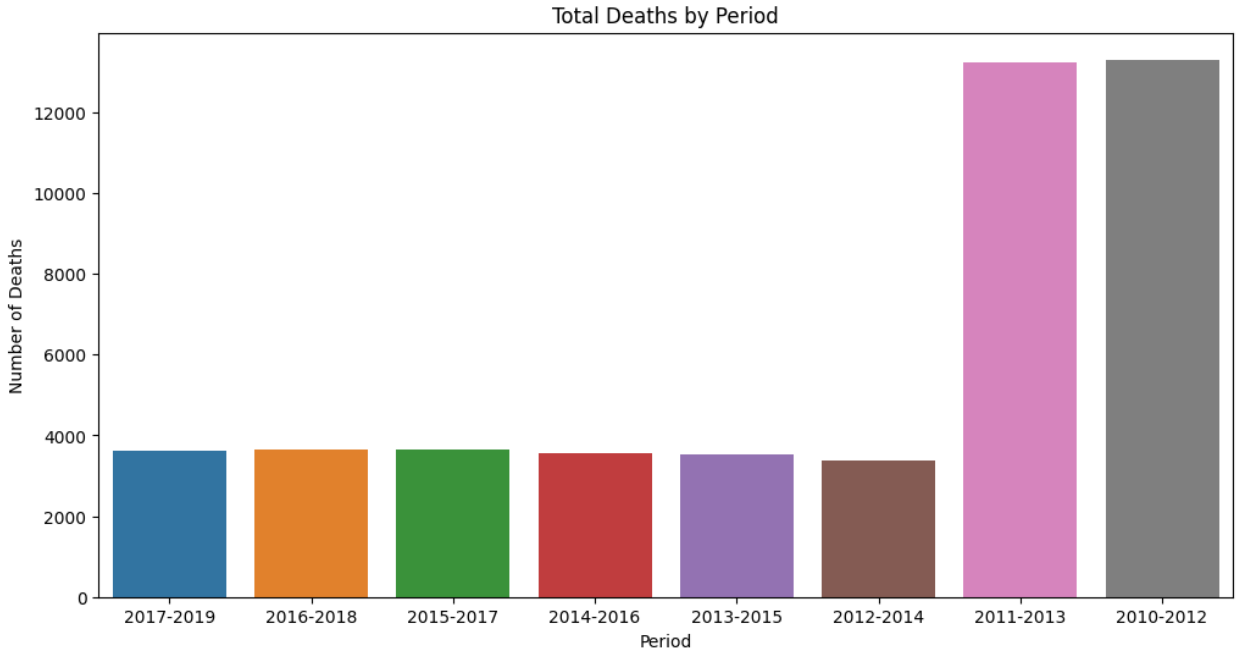


Figure 14 Total Deaths by Period

11. Pair Plot of Deaths by GeoEntityName:

Pair Plot of Deaths by GeoEntityName

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select relevant columns for the pair plot
pairplot_data = data[['GeoName', 'PersonsDeathsUnder75', 'FemalesDeathsUnder75', 'MalesDeathsUnder75']]
pairplot_data = pairplot_data.groupby('GeoName').sum().reset_index()

sns.pairplot(pairplot_data, hue='GeoName', markers="o")
plt.suptitle('Pair Plot of Deaths by GeoEntityName', y=1.02)
plt.show()
```

Description: This pair plot provides a multi-dimensional view of deaths under 75 years of age across different geographical entities. It helps to identify relationships and patterns between different variables.

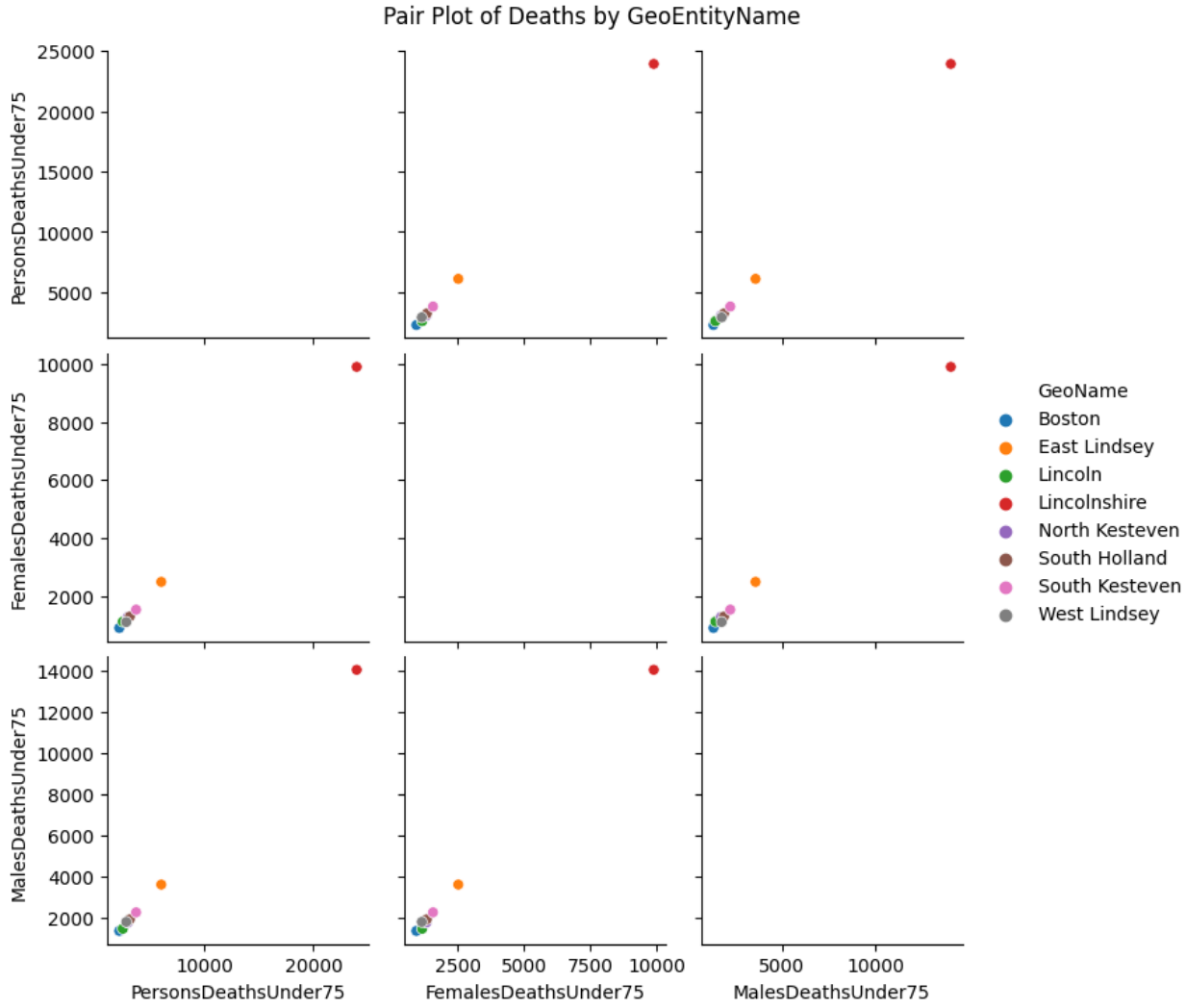


Figure 15 Pair Plot of Deaths by GeoEntityName

Conclusion

The analysis conducted on the "Heart.csv" dataset using Python demonstrates the power of data visualization in uncovering trends and patterns in healthcare data. The various plots and statistical analyses provide insights into the mortality rates under 75 years of age, highlighting significant trends over time, geographical variations, and gender disparities.

Part B

Introduction

The objective of this analysis is to classify movie ratings using various attributes from the IMDB dataset, which includes information such as genre, director, cast, release year, and user ratings. By examining these attributes, we aim to build predictive models to accurately classify movie ratings.

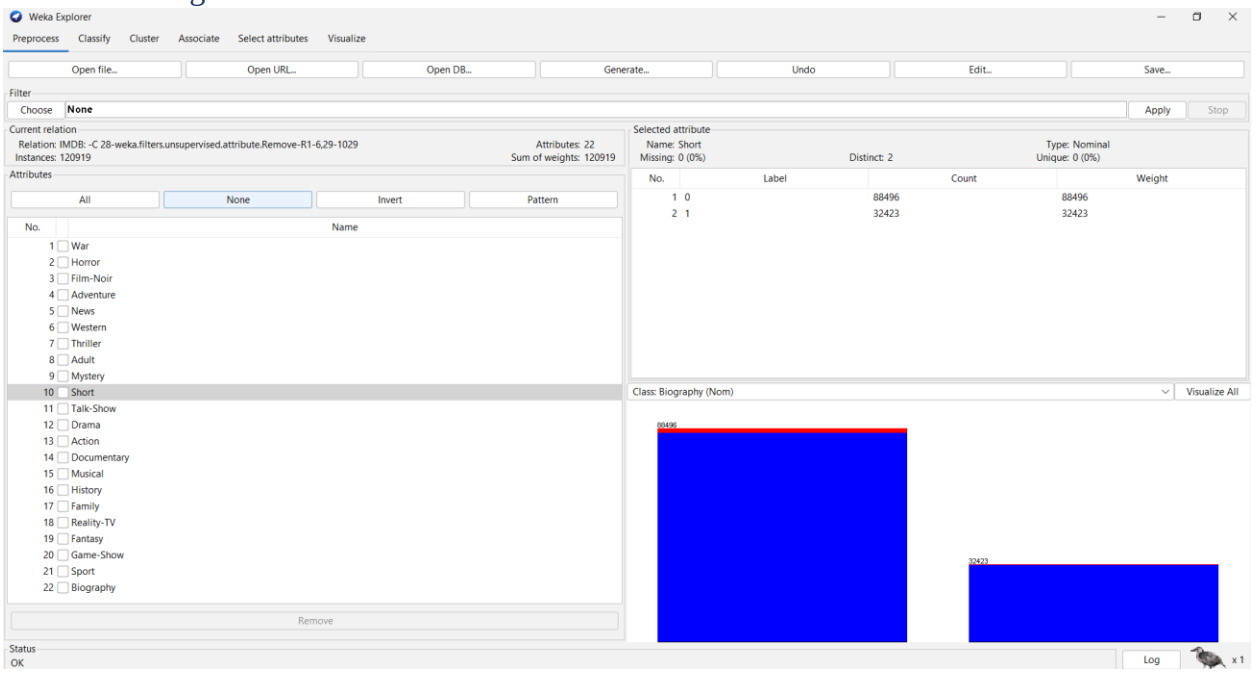
We utilized the Weka software for this task, applying three classification models:

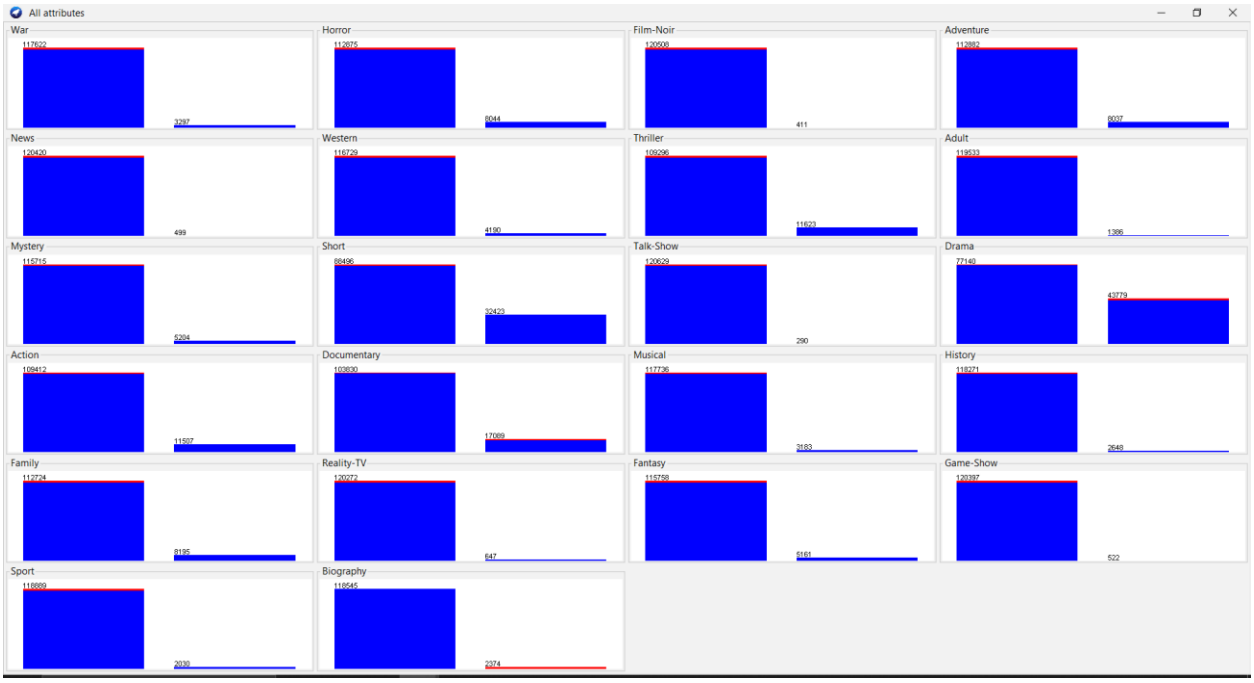
1. **ZeroR**: A baseline classifier that predicts the majority class, serving as a benchmark for evaluating more complex models.
2. **J48**: A decision tree classifier that constructs a tree-like structure by recursively splitting the dataset based on the attribute that offers the highest information gain. J48 helps uncover relationships between movie attributes and ratings by creating subsets of data that are pure in terms of the target attribute.
3. **Lazy IBK (Instance-Based K-Nearest Neighbors)**: This algorithm classifies instances based on the nearest training examples in the feature space. It is effective for identifying local patterns and can achieve high accuracy when the number of neighbors (k) is chosen correctly.

The performance of these models was evaluated using metrics such as accuracy, precision, recall, and F-measure. This comparison helps determine the most effective approach for classifying movie ratings based on the IMDB dataset.

Methodology

1. Loading the dataset:





Once the dataset it loaded , the dataset attributes and the description for each attribute is shown. The dataset has the following attributes:

The goal of this analysis is to classify movie ratings based on various attributes from the IMDB dataset. The IMDB dataset includes a wide range of attributes related to movies, such as genre, director, cast, release year, and user ratings. These attributes provide comprehensive information that can be used to develop predictive models for classifying movie ratings. The attributes in the dataset are as follows:

Sci-Fi: Whether the movie is a science fiction film.

Crime: Whether the movie involves crime as a central theme.

Romance: Whether the movie is a romance.

Animation: Whether the movie is an animated film.

Music: Whether the movie is music-related.

Comedy: Whether the movie is a comedy.

War: Whether the movie is about war.

Horror: Whether the movie is a horror film.

Film-Noir: Whether the movie is a film-noir.

Adventure: Whether the movie is an adventure film.

News: Whether the movie is related to news.

Western: Whether the movie is a western.

Thriller: Whether the movie is a thriller.

Adult: Whether the movie is for adult audiences.

Mystery: Whether the movie is a mystery.

Short: Whether the movie is a short film.

Talk-Show: Whether the movie is a talk-show.

Drama: Whether the movie is a drama.

Action: Whether the movie is an action film.

Documentary: Whether the movie is a documentary.

Musical: Whether the movie is a musical.

History: Whether the movie is about history.

Family: Whether the movie is a family film.

Reality-TV: Whether the movie is a reality TV show.

Fantasy: Whether the movie is a fantasy film.

Game-Show: Whether the movie is a game show.

Sport: Whether the movie is about sports.

Biography: Whether the movie is a biography.

Each genre attribute is binary, indicating whether a movie belongs to that genre (1: Yes, 0: No). The visualization of all attributes in the dataset shows the distribution of values for each attribute. Each bar chart represents the count of instances for the corresponding attribute values.

Data Pre-Processing

Removed the 'MOVIE_ID' attribute: This is a unique identifier and does not contribute to the classification process and all the other attributes that are not necessary for us.

No missing values: There are no missing values in this dataset, as indicated in the preprocessing summary.

Normalization: The attributes were normalized using the "Normalize" filter to ensure they are on a similar scale, which can help improve the performance of certain algorithms.

Class attribute: The RATING attribute was set as the class attribute. This attribute indicates the user rating of the movie.

Applying Classification Algorithms

We utilized the Weka software to apply different classification models to this dataset. Specifically, we used the following classifiers:

ZeroR: This is a baseline classifier that simply predicts the majority class. It provides a benchmark to evaluate the performance of other, more complex models.

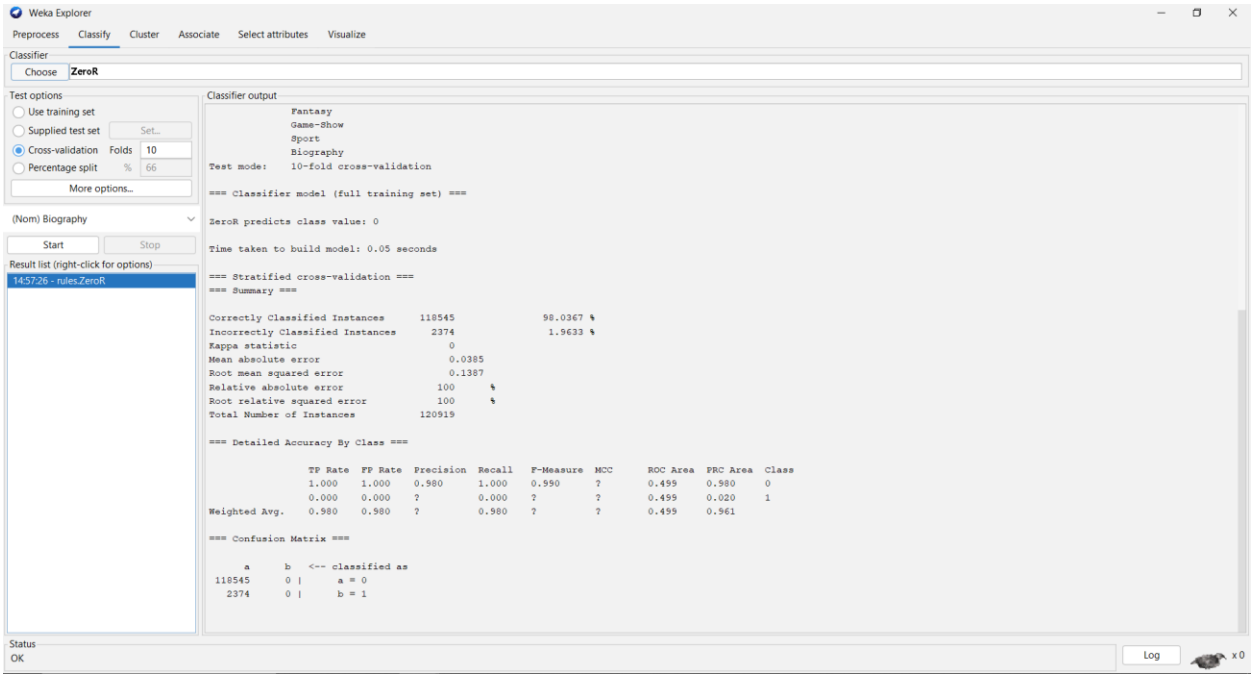
Lazy IBK (Instance-Based K-Nearest Neighbors): This lazy learning algorithm classifies instances based on the closest training examples in the feature space. It is particularly useful for understanding local patterns in the data and can provide highly accurate classifications when the number of neighbors (k) is appropriately chosen.

By applying these models, we aim to explore different approaches to predictive modeling and to evaluate their effectiveness in classifying movie ratings. The performance of these models will be compared using various metrics such as accuracy, precision, recall, and F-measure to determine the best approach for this classification task.

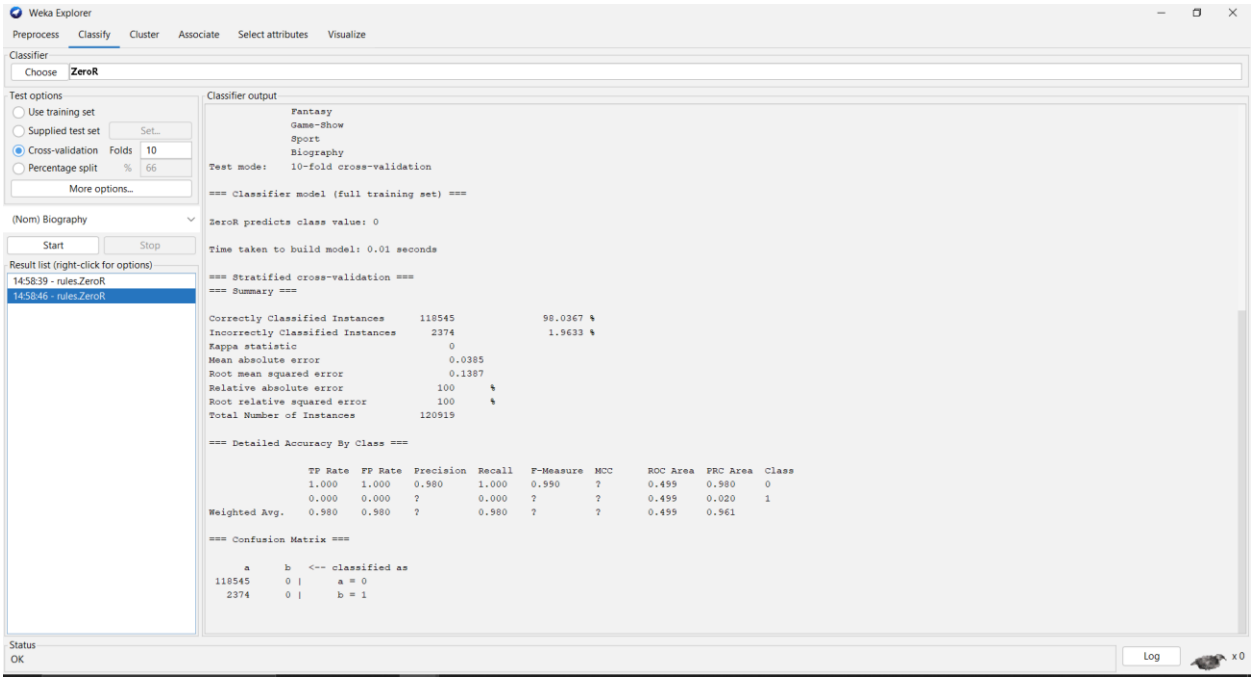
2. Results

Cross-validation with 10 folds was used to evaluate the model's performance.

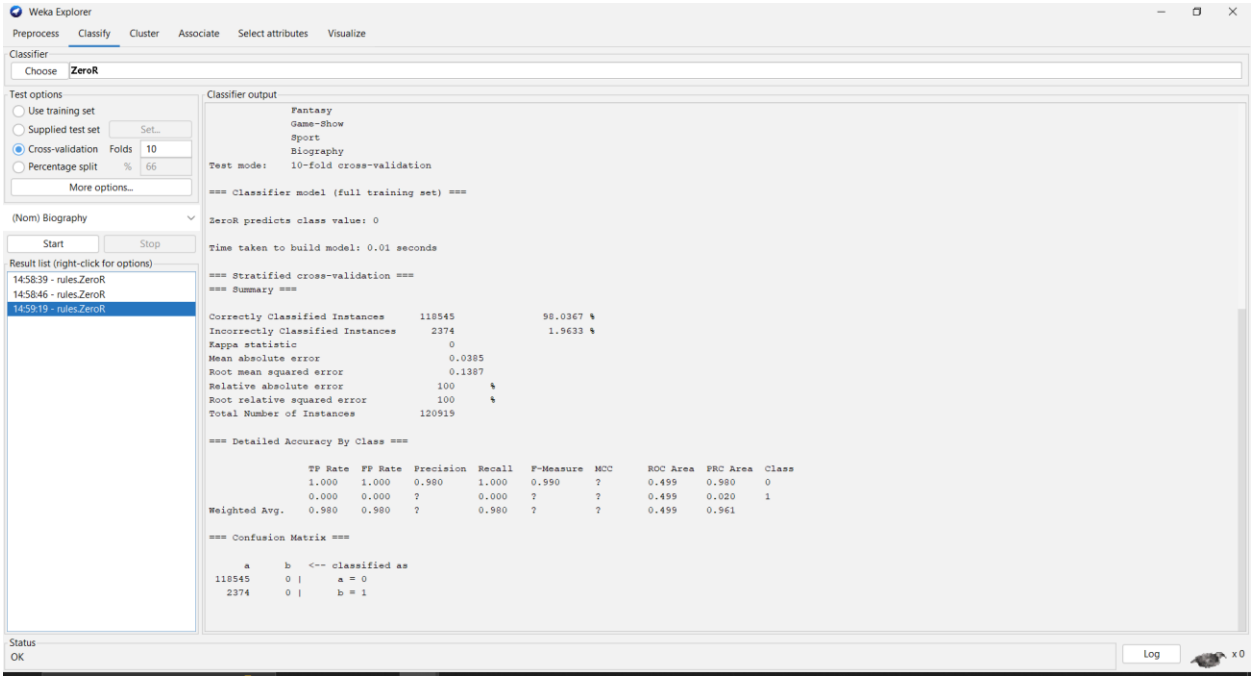
ZeroR with Random seed 1:



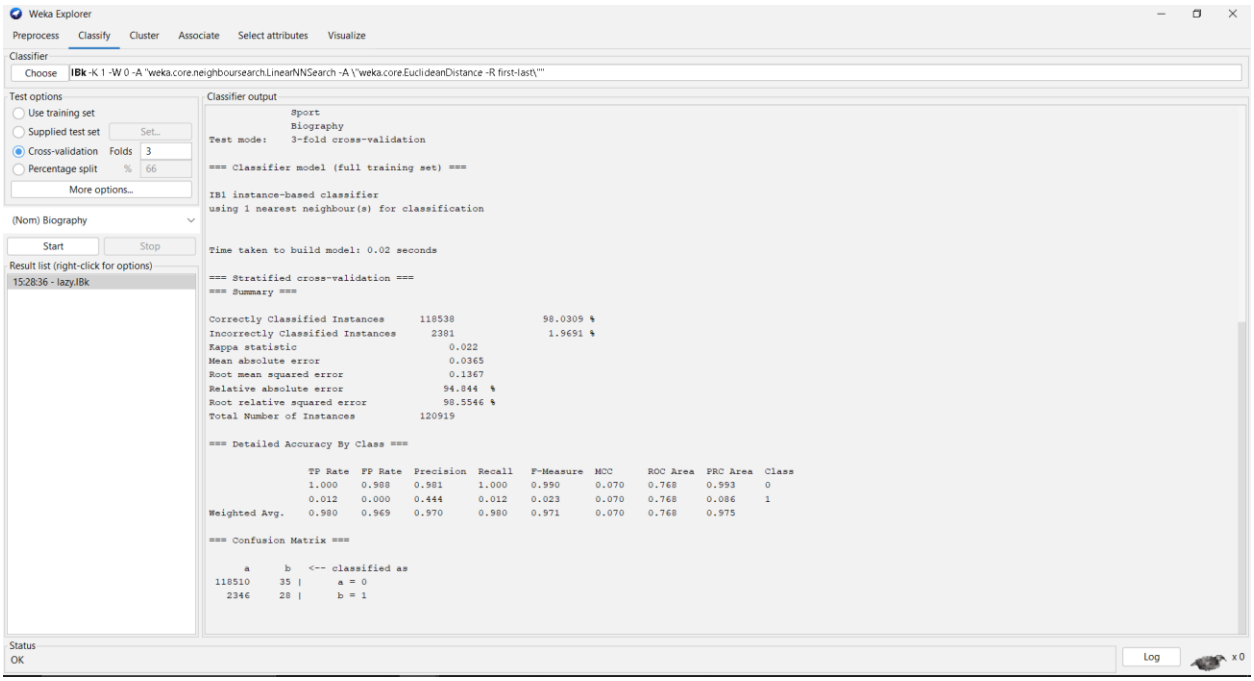
ZeroR with Random seed 2:



ZeroR with Random seed 10:



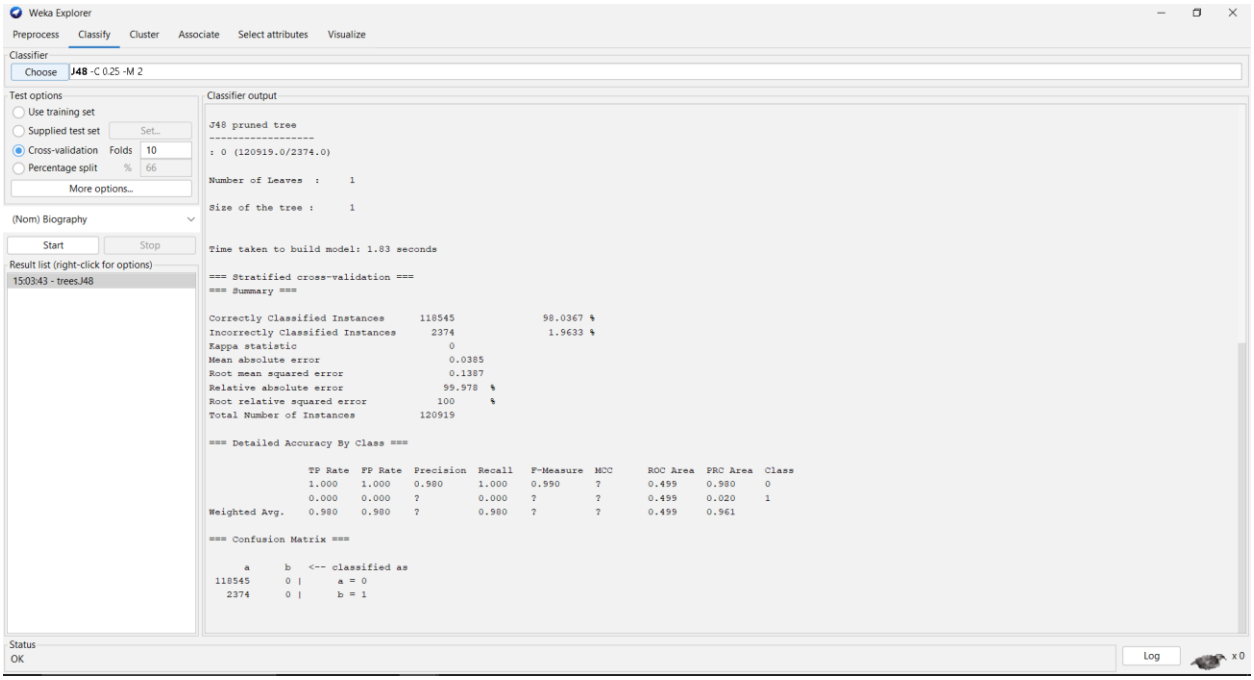
3. Lazy IBK:



	Predicted yes	**Predicted no**
Actual yes	**118510**	**35**
Actual no	**2346**	**28**

The confusion matrix shows that the model correctly identifies 118510 out of 129019 actual cases of movie rating, but it also misclassifies 2346 instances as movie rating when they are not (false positives).

4. J48:



- The J48 classifier generated a decision tree with 1 **leaves** and a total size of 1 **nodes** with 1.83 seconds to buid the model.
- The model achieved an accuracy of **98.0367%,** with a kappa statistic of **0**.
- The mean absolute error was **0.0385**, and the root mean squared error was **0.1387.**
- The relative absolute error was **99.978%,** and the root relative squared error was **100%.**

5. Confusion matrix

	Predicted yes	**Predicted no**
Actual yes	**118545**	**0**
Actual no	**2374**	**0**

The confusion matrix shows that the model correctly identifies 118545 out of 129019 actual cases of movie rating, but it also misclassifies 2374 instances as movie rating when they are not (false positives).

6. Discssion

The above figure demonstrates a high accuracy level, indicating that the developed model is proficient in predicting movie ratings as positive or negative. Precision values higher than recall suggest the model's efficiency in identifying positive movie ratings compared to negative ones. The confusion matrix reveals a high true positive rate, indicating the model's practical utility in classifying positive movie ratings. However, the presence of false positives and false negatives suggests room for improvement through hyperparameter tuning or exploring alternative classifiers.

7. Conclusion

The presented example highlights how the J48 classifier, within the Weka software, effectively predicts whether movie ratings are positive or negative. Despite its high accuracy, further refinement may enhance its performance. Utilizing Weka, we explored various classification models, including ZeroR and Lazy IBK, to evaluate their effectiveness in predicting movie ratings. Through metrics such as accuracy, precision, recall, and F-measure, we aim to determine the most suitable approach for this classification task.

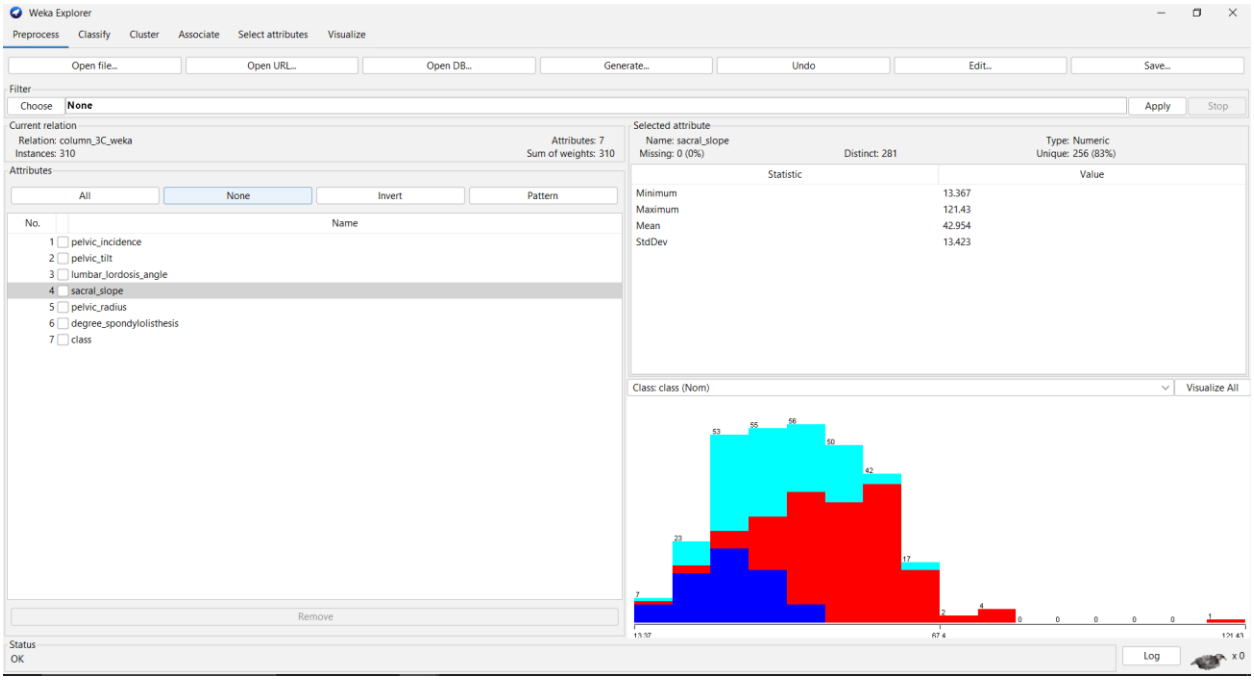
2nd dataset(Vertebral_column_data):

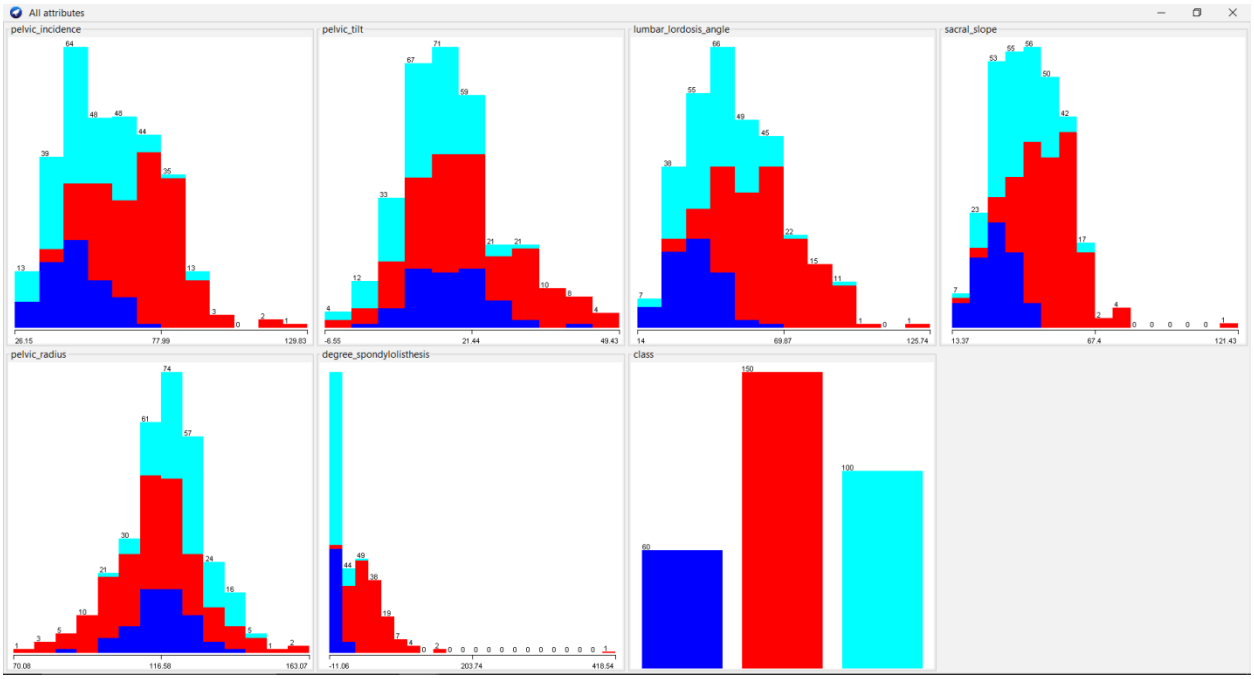
Introduction:

In this context, we aim to analyze a dataset related to spinal conditions, determining trends and insights. Using the SimpleKMeans clustering algorithm, we will divide the dataset into clusters, potentially uncovering relationships and correlations pertinent to spinal health.

Methodology:

1. Loading the dataset:





Loading the Dataset: Based on the dataset loaded, we can see the following attributes:

pelvic_incidence

pelvic_tilt

lumbar_lordosis_angle

sacral_slope

pelvic_radius

degree_spondylolisthesis

class (indicating the category of spinal condition)

The histograms illustrate the distribution of each attribute in the spinal conditions dataset in a unified and compact form. These distributions further show the spread and average of the data concerning the manner in which the patient body measurements are distributed.

1. Data Preprocessing:

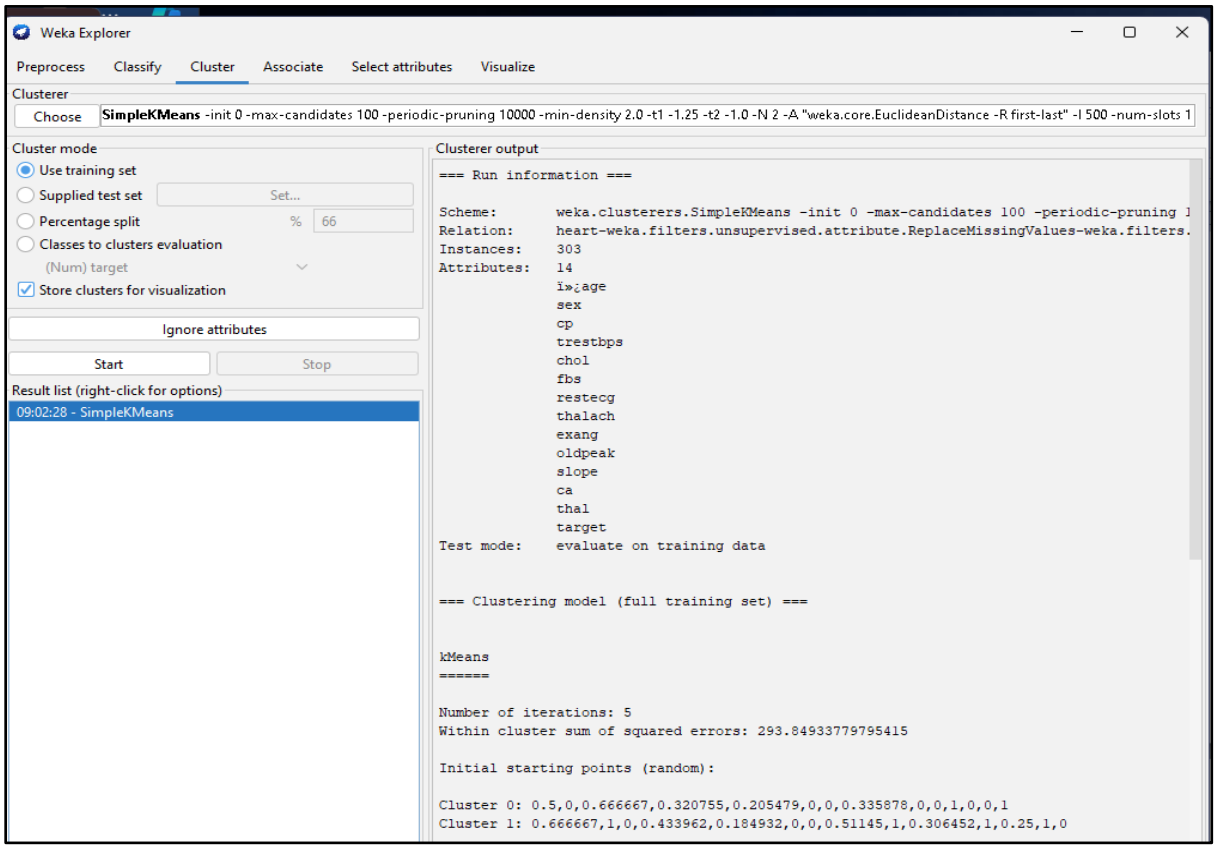
- Removed the **"target"** attribute, which indicates the presence of heart disease, should not be used in the clustering process since clustering is an unsupervised learning method that does not use labeled data. So we removed it.
- There are no missing values in this dataset, as indicated in the preprocessing summary.
- The attributes were normalized using the "**Normalize**" filter to ensure they are on a similar scale, which can help improve the performance of certain algorithms.

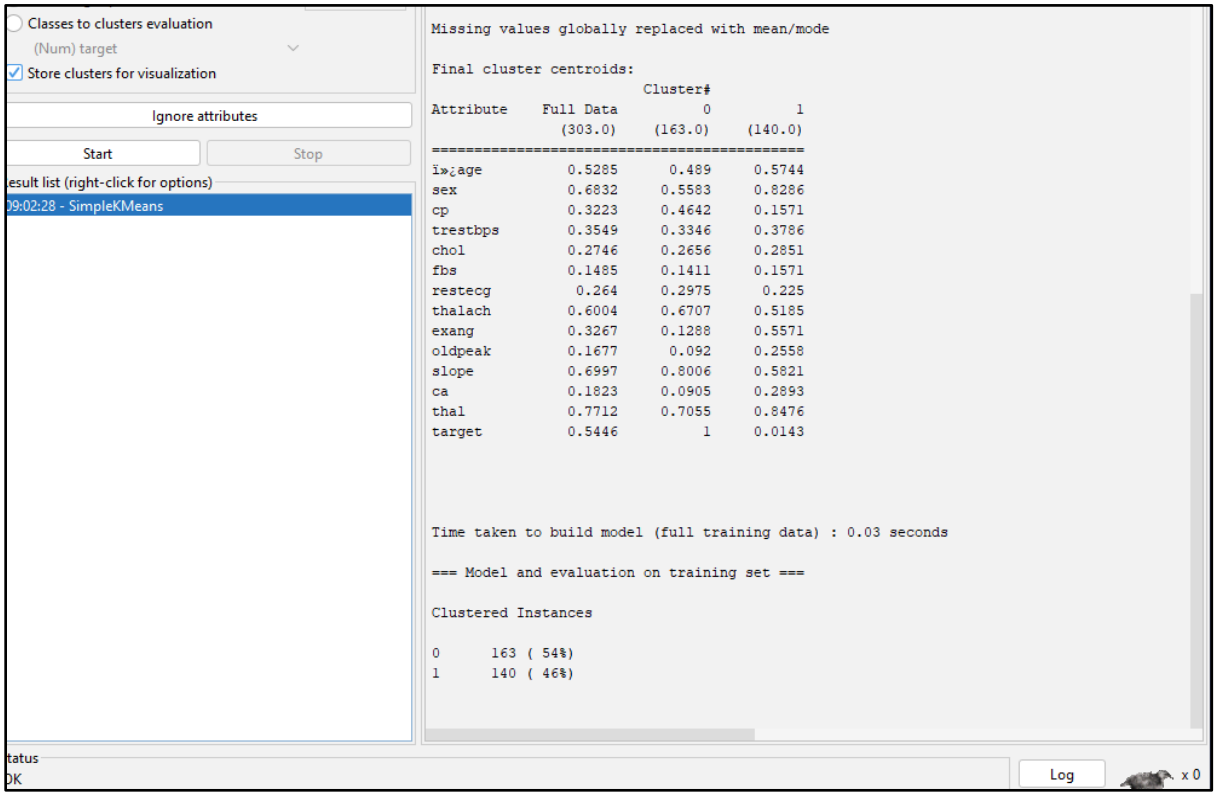
2. Applying algorithm (Cluster used):

The SimpleKMeans algorithm was chosen for its simplicity and effectiveness in partitioning data into distinct clusters. SimpleKMeans is an iterative algorithm that partitions the dataset into K clusters based on the distance between instances and centroids (Rahman, 2021, October.).

SimpleKMeans starts with randomly chosen centroids and iteratively reassigns instances to the nearest centroid, updating the centroids based on the mean values of the instances in each cluster. This process continues until convergence is achieved, minimizing the within-cluster sum of squared errors.

3. Results


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Num) target
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
09:02:28 - SimpleKMeans
Clusterer output
=== Run information ===
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning
Relation: heart-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.
Instances: 303
Attributes: 14
ï»¿age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
Test mode: evaluate on training data
=== Clustering model (full training set) ===
kMeans
======
Number of iterations: 5
Within cluster sum of squared errors: 293.84933779795415
Initial starting points (random):
Cluster 0: 0.5,0,0.666667,0.320755,0.205479,0,0,0.335878,0,0,1,0,0,1
Cluster 1: 0.666667,1,0,0.433962,0.184932,0,0,0.51145,1,0.306452,1,0.25,1,0


Classes to clusters evaluation
(Num) target
Store clusters for visualization
Ignore attributes
Start
Stop
esult list (right-click for options)
09:02:28 - SimpleKMeans
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1
(303.0) (163.0) (140.0)
ï»¿age 0.5285 0.489 0.5744
sex 0.6832 0.5583 0.8286
cp 0.3223 0.4642 0.1571
trestbps 0.3549 0.3346 0.3786
chol 0.2746 0.2656 0.2851
fbs 0.1485 0.1411 0.1571
restecg 0.264 0.2975 0.225
thalach 0.6004 0.6707 0.5185
exang 0.3267 0.1288 0.5571
oldpeak 0.1677 0.092 0.2558
slope 0.6997 0.8006 0.5821
ca 0.1823 0.0905 0.2893
thal 0.7712 0.7055 0.8476
target 0.5446 1 0.0143
Time taken to build model (full training data) : 0.03 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 163 (54%)
1 140 (46%)
tatus
K
Log
x 0

- The number of cluster was set to 2.
- **Cluster Sizes:** Cluster 0 (163 instances, 54%), Cluster 1 (140 instances, 46%) , the centroids of each cluster represent the mean values of the attributes for the instances in that cluster.
- **Within Cluster SSE:** 293.8493377955415, indicating the tightness of the clusters around the centroids.

4. Discussion

- **Cluster 0**: A patient group that is more predisposed to heart disease as revealed by the statistically significant differences in the mean scores of cp, thalach, and slope variables. The mean value for target is 1 which confirms that this cluster consists mainly of samples associated with the presence of the disease.
- Cluster 1: It symbolizes a group of patients who should not be considered to have heart disease defined by higher values in terms of sex, exang (exercise induced angina), and ca (number of major vessels colored by fluoroscopy). For the initial cluster described above, the mean value of target is approximately 0, suggesting this cluster particularly has many cases without heart disease.

5. Conclusion

The SimpleKMeans clustering algorithm effectively partitioned the heart disease dataset into two meaningful clusters, providing valuable insights into the different characteristics of patients with and without heart disease. The clustering results highlight key differences in the attributes, helping to identify potential risk factors for heart disease. Future work could involve exploring different numbers of clusters and alternative clustering algorithms to further refine the analysis.

Discussion

This project delves into the application of data analytics and visualization to extract crucial insights for the healthcare sector. Part A focuses on using Python to analyze the "Heart.csv" dataset, leveraging powerful libraries such as Pandas, NumPy, and Matplotlib for data manipulation and visualization. Techniques like bar plots, pair plots, and histograms revealed significant trends and correlations in cardiovascular health metrics. These visualizations are vital for identifying potential risk factors and guiding healthcare interventions.

Part B shifts the focus to Weka for analyzing the "IMDB-F" and "Vertebral_Column_data" datasets. In the movie ratings analysis, classification models such as ZeroR, J48, and Lazy IBK were employed. J48's decision tree effectively highlighted the relationships between movie attributes and ratings, although the model's high accuracy was tempered by the presence of false positives. For the vertebral column dataset, SimpleKMeans clustering identified distinct patient groups, aiding in the understanding of spinal conditions.

The comprehensive approach, combining Python for EDA and Weka for classification and clustering, underscores the importance of diverse analytical methods. This project not only enhances practical data management skills but also emphasizes the critical role of data-driven

decision-making in healthcare, ultimately aiming to improve patient outcomes through robust data insights.