

T.C.
BAHCESEHIR UNIVERSITY
GRADUATE SCHOOL OF EDUCATION
THE DEPARTMENT OF BIG DATA ANALYTICS AND MANAGEMENT

**FINE-TUNING AND HYPERPARAMETER OPTIMIZATION;
A PATH TO SUPERIOR MODEL PERFORMANCE AND ACCURATE
PREDICTIONS**

MASTER'S THESIS
LAMISS GARGOURI

ISTANBUL 2023

T.C.
BAHCESEHIR UNIVERSITY
GRADUATE SCHOOL OF EDUCATION
THE DEPARTMENT OF BIG DATA ANALYTICS AND MANAGEMENT

**FINE-TUNING AND HYPERPARAMETER OPTIMIZATION;
A PATH TO SUPERIOR MODEL PERFORMANCE AND ACCURATE
PREDICTIONS**

MASTER’S THESIS

LAMISS GARGOURI

THESIS ADVISOR
Assoc.Prof.Dr.ATINÇ YILMAZ

ISTANBUL 2023
T.C



**T.C.
BAHCESEHIR UNIVERSITY
GRADUATE SCHOOL**

...../...../.....

MASTER THESIS APPROVAL FORM


Program Name:	Big Data Analytics and Management
Student's Name and Surname:	Lamiss Gargouri
Name Of The Thesis:	FINE-TUNING AND HYPERPARAMETER OPTIMIZATION; A PATH TO SUPERIOR MODEL PERFORMANCE AND ACCURATE PREDICTIONS
Thesis Defense Date:	14/06/2023

This thesis has been approved by the Graduate School which has fulfilled the necessary conditions as Master thesis.

.....

Director of Institute

This thesis was read by us, quality and content as a Master's thesis has been seen and accepted as sufficient.

	Title, Name	Institution	Signature
Thesis Advisor:	Assoc.Prof.Dr. Atinç Yılmaz	Bahcesehir university	 ATINÇ YILMAZ <small>Dijital olarak imzalayan ATINÇ YILMAZ Tarih: 2023.06.27 00:36:02 +03'00'</small>
2nd Member	Asst.Prof.Dr. Tamer Uçar	Bahcesehir university	TAMER UÇAR <small>Digitally signed by TAMER UÇAR Date: 2023.06.28 00:48:35 +03'00'</small>
3rd Member (Outside Institution)	Asst.Prof.Dr. Ediz Şaykol	Beykent university	EDİZ ŞAYKOL <small>Dijital olarak imzalayan EDİZ ŞAYKOL Tarih: 2023.06.28 21:51:52 +03'00'</small>

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname :

Signature

ABSTRACT
FINE-TUNING AND HYPERPARAMETER OPTIMIZATION;
A PATH TO SUPERIOR MODEL PERFORMANCE AND
ACCURATE PREDICTIONS

Gargouri, Lamiss

Big Data Analytics & Management Masters Program

Thesis Advisor: Assoc.Prof.Dr.ATINÇ YILMAZ

January 2023, 161 pages

This thesis explores the use of data mining tools and techniques in business intelligence, focusing on how they can enhance the process of obtaining insights and knowledge from massive datasets, enabling more effective decision-making in businesses.

The research is divided into sections that explain the core principles of business intelligence and data mining, as well as how they can be combined to improve business analytics. A practical application and case study focusing on forecasting song popularity on the Spotify platform showcases the usefulness and potential of data mining techniques in a real-world context.

The case study describes data collecting, preprocessing, and feature engineering methodologies, as well as data mining techniques used to find patterns and trends in music data. The study adds to the existing literature by demonstrating the actual implementation of data mining tools and methodologies for business intelligence.

This study adds to the current literature by demonstrating the actual implementation of data mining tools and methodologies for business intelligence. This thesis highlights the potential benefits and problems of integrating data mining techniques in real-world contexts by concentrating on the specific instance of forecasting song popularity on Spotify.

The insights provided will help music business experts, marketing teams, and artists make informed decisions about song releases, promotional methods, and target audience selection. Overall, this thesis emphasizes the importance of data mining tools and methodologies in improving business intelligence operations and bridges the theoretical and practical divides by providing a real application and case study that demonstrates the power of data mining approaches in deriving actionable insights from massive datasets.

Keywords: Data Mining, Business Intelligence, Decision-making, Case Study, Implementation.

ÖZ

FINE-TUNING AND HYPERPARAMETER OPTIMIZATION; A PATH TO SUPERIOR MODEL PERFORMANCE AND ACCURATE PREDICTIONS

Gargouri, Lamis

Big Data Analytics & Management Masters Program

Tez Danışmanı: Assoc.Prof.Dr.ATINÇ YILMAZ

Mayıs 2023, 161 sayfa

1

Bu tez, veri madenciliği araç ve tekniklerinin iş zekası bağlamında kullanımını araştırmaktadır. Çalışma, veri madenciliğinin, işletmelerde daha etkili karar almaya olanak tanıyarak, büyük veri kümelerinden yararlı içgörüler ve bilgiler elde etme sürecini nasıl geliştirebileceğini inceliyor. Araştırma, iş zekası ve veri madenciliğinin temel ilkelerinin yanı sıra iş analitiğini geliştirmek için bunların nasıl birleştirilebileceğini açıklayan bölümlere ayrılmıştır. Gerçek dünya bağlamında veri madenciliği tekniklerinin yararlılığını ve potansiyelini sergileyen, Spotify platformunda şarkı popüleritesini tahmin etmeye odaklanan pratik bir uygulama ve vaka çalışması da sağlanmaktadır.

Anahtar Kelimeler: Veri Madenciliği, İş Zekası, Veri Kümeleri, Karar Verme, Uygulama, Örnek Olay İncelemesi, Öngörü, Teknikler.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to the following individuals who have played significant roles in the completion of this thesis:

First and foremost, I am immensely thankful to my thesis supervisor, Professor Atınc Yilmaz, for their guidance, support, and expertise throughout this research journey. Their invaluable insights, constructive feedback, and unwavering encouragement have been instrumental in shaping this work.

I would like to extend my sincere appreciation to the staff and faculty members of Bahçeşehir University/ Big Data Analytics & Management Department. Their dedication to academic excellence, resources, and support services have created an environment conducive to research and learning. Their commitment to nurturing the intellectual growth of students is commendable.

I would like to express my gratitude to the jury members for their time, effort, and expertise in evaluating this thesis. Their constructive feedback and insightful suggestions have undoubtedly contributed to the quality and refinement of this work.

To my parents, I am eternally grateful for their unwavering love, encouragement, and unwavering belief in my abilities. Their endless support, sacrifices, and motivation have been the cornerstone of my academic journey. Their belief in my potential has been a constant source of inspiration.

Finally, I extend my heartfelt thanks to my close friends, whose unwavering support, understanding, and encouragement have been vital throughout this thesis. Their presence, friendship, and belief in me have provided me with strength and motivation during challenging times.

Once again, I express my deepest appreciation and gratitude to all those who have played a role, both big and small, in shaping this research endeavor. Your support and belief in me have been truly humbling and inspiring.

TABLE OF CONTENTS

ETHICAL CONDUCT.....	iii
ABSTRACT.....	iv
ÖZ.....	v
ACKNOWLEDGMENTS.....	vii
TABLE OF CONTENTS.....	viii
LIST OF TABLES.....	viii
LIST OF FIGURES/ILLUSTRATIONS/SCHEMES.....	x
list of symbols/abbreviations.....	Xi
Chapter 1: Introduction	1
Chapter 2: Literature Review	5
Chapter 3: Business Intelligence	9
3.1 The Importance of Business Intelligence.....	9
3.2 Concepts & Applications of Business Intelligence	11
3.3 Business Intelligence For Better Decision.....	13
3.4 Business Intelligence Tools.....	14
3.5 Business Intelligence Skills.....	16
Chapter 4: Data Mining	18
4.1 About Data Mining	18
4.1.1. The Origins Of Data Mining	18
4.1.2 Benefits Of Data Mining	19
4.1.3 Disadvantages Of Data Mining	20
4.1.4 The Usage Of Data Mining	22
4.2 Top Data Mining Techniques	23
4.2.1 Decision Trees	23
4.2.2 AdaBoost (Adaptive Boosting)	24
4.2.3 Support Vector Machine (SVM)	25
4.2.4 Logistic Regression	26
4.2.5 K-nearest Neighbors (KNN)	27
4.2.6 Random Forest Classifier	28
4.3 Tools Used In Data Mining	29
4.4 The Stages Of Data Mining	30

4.5 The Main Data Mining Users.....	32
4.6 Real-life Data Mining Projects Examples.....	33
4.7 The Value Of Data Mining Projects	35
Chapter 5: Data Warehouse	37
5.1 Design Considerations For Data Warehouse	38
5.2 Data Warehouse Development Approaches.....	39
5.3 Data Warehouse Architecture.....	41
5.4 Data Resources.....	43
5.5 Data Loading Processes.....	45
5.6 Data Warehouse Design.....	46
5.7 Data Warehouse Access.....	48
5.8 Data Warehouse Best Practices.....	49
Chapter 6: Data Mining & Business Intelligence	51
6.1 Data Mining in Business Analytics.....	53
6.2 The Relationship Between DM & BI	54
6.3 The Cloud & The Future Of DM For BI.....	55
6.4 The Role Data Mining Plays For BI.....	57
6.5 The Benefits Of Data Mining In BI.....	58
6.6 The Challenges Of Data Mining In BI.....	60
6.7 The Industries That Benefit From DM In BI.....	63
6.7.1 Retails	63
6.7.2 Health Care	64
6.7.3 Financial Services	65
6.7.4 Manufacturing	66
6.7.5 Telecommunications	66
6.7.6 Government	67
6.8 Key Differences Between DM & BI.....	69
6.9 Summary.....	70
Chapter 7: Experimental Studies	72
7.1 Dataset	74
7.2 Code	76
7.2.1 Import Libraries	76
7.2.2 Data Visualization	77

7.2.3 Exploratory Data Analysis (EDA)	81
7.2.3.1 The Relationship Between All The Features.....	81
7.2.3.2 Popularity's Most Linearly Connected Features...	83
7.2.3.3 Mean Popularity Across The Year.....	84
7.2.3.4 Mean Popularity Across Energy.....	86
7.2.3.5 Mean Popularity Across Loudness.....	87
7.2.3.6 Mean Popularity Across Danceability	88
7.2.3.7 Mean Popularity Across Tempo.....	89
7.2.3.8 Mean Popularity Across Acousticness.....	90
7.2.3.9 Conclusion.....	91
7.2.4 Initialisation.....	92
7.2.4.1 Preparation.....	92
7.2.4.2 Transformations Of Characteristics.....	97
7.2.5 Prediction Models	99
7.2.5.1 Decision Tree	100
7.2.5.2 K_nearest Neighbor Classifier	108
7.2.5.3 Logistic Regression	110
7.2.5.4 AdaBoost Classifier	112
7.2.5.5 Random Forest Classifier	114
7.2.5.6 Results	116
7.2.5.7 Tests	118
7.2.5.8 Conclusion	123
7.2.6 Fine Tuning	124
7.2.7 Summary.....	126
Chapter 8: Conclusion	128
Chapter 9: Comparison Studies	131
Chapter 10: Suggestions	138
References	140

LIST OF TABLES

TABLES

Table 1 Pros & Cons of Data Mining	21
Table 2 Data Mining Tools	29
Table 3 Data Mining VS Business Intelligence	52
Table 4 Benefits of Data Mining in BI	59
Table 5 The Challenges of DM in BI	61
Table 6 The Industries That Benefit from DM & BI	68
Table 7 Results Comparison Table	123

LIST OF FIGURES

FIGURES

Figure 1	Decision Tree Code Example	24
Figure 2	Adaboost Classifier Code Example	24
Figure 3	Support Vector Machine Code Example	25
Figure 4	Logistic Regression Code Example	26
Figure 5	K-Nearest Neighbors Code Example	28
Figure 6	Random Forest Classifier Code Example	28
Figure 7	Dataset	74
Figure 8	Screenshot of Importing Libraries	76
Figure 9	Screenshot of Reading, Dropping & Removing	77
Figure 10	Screenshot of Inspecting the First few Rows of the Dataset....	77
Figure 11	Screenshot of Inspecting the Last few Rows of the Dataset	78
Figure 12	Screenshot of the Dataset Shape	78
Figure 13	Screenshot of Adding the pop_rating Column	79
Figure 14	Screenshot of Creating pop_rating Countplot	80
Figure 15	Graph of Popularity Ratings	80
Figure 16	Screenshot of Creating the Correlation Matrix Using Seaborn ..	81
Figure 17	Correlation of Numerical Characteristics	82
Figure 18	Screenshot of Creating the Heatmap of the Correlation Matrix .	83
Figure 19	Heatmap of the Popularity Most Linearly Connected Characteristics.....	83
Figure 20	Screenshot of Creating the Graph of Mean Popularity Accross the Year.....	84
Figure 21	Graph of Mean Popularity Accross the Year	85
Figure 22	Code of the Function That Create the Scatterplot	85
Figure 23	Code of Creating the Regress_plot Function (Mean Popularity Across the Energy)	86
Figure 24	Graph of Mean Popularity Across the Energy	87
Figure 25	Code of Creating the Regress_plot Function (Mean Popularity Across Loudness)	87

Figure 26 Graph of Mean Popularity Across Loudness	88
Figure 27 Code of Creating the Regress_plot Function (Mean Popularity Across Danceability)	88
Figure 28 Graph of Mean Popularity Across Danceability	89
Figure 29 Code of Creating the Regress_plot Function (Mean Popularity Across Tempo)	89
Figure 30 Graph of Mean Popularity Across Tempo	90
Figure 31 Code of Creating the Regress_plot Function (Mean Popularity Across Acousticness)	90
Figure 32 Graph of Mean Popularity Across Acousticness	91
Figure 33 Screenshot of the Preparation Code 1	92
Figure 34 Screenshot of the Preparation Code 2	92
Figure 35 Screenshot of the Preparation Code 3	93
Figure 36 Screenshot of Arranging the Popularity Values	94
Figure 37 Result of the Occurrences	94
Figure 38 Screenshot of Defining the List of Artists and Dropping Unnecessary Columns	95
Figure 39 Screenshot of Removing Rows	96
Figure 40 Screenshot of Splitting Dataframe Into Features	96
Figure 41 Screenshot of Using the ColumnTransformer Object	97
Figure 42 Screenshot of Transforming the X & y Train	98
Figure 43 Screenshot of Defining the run_model Function	99
Figure 44 Decision Tree Code	100
Figure 45 Decision Tree Accuracy Result	101
Figure 46 Screenshot of the Confusion Matrix Code (Decision Tree)	103
Figure 47 Confusion Matrix Results	104
Figure 48 Screenshot of the SVM Classifier Code	106
Figure 49 Confusion Matrix of the SVM Classifier	106
Figure 50 KNN Classifier Code	108
Figure 51 Confusion Matrix of the KNN Classifier	108
Figure 52 Screenshot of Logistic Regression Code	110
Figure 53 Confusion Matrix of the Logistic Regression	110
Figure 54 Screenshot of Adaboost Classifier Code	112
Figure 55 Confusion Matrix of Adaboost Classifier	113
Figure 56 Screenshot of Random Forest Classifier Code	114

Figure 57 Confusion Matrix of the Random Forest Clasifier	115
Figure 58 Screenshot of the Results Code	116
Figure 59 The Accuracy Results of all the Models	117
Figure 60 Screenshot of Creating Groups	118
Figure 61 Dropping G1	118
Figure 62 The Accuracy Results After Dropping G1	119
Figure 63 Dropping G2	119
Figure 64 The Accuracy Results After Dropping G2	120
Figure 65 Dropping G3	120
Figure 66 The Accuracy Results After Dropping G3	121
Figure 67 Dropping G4	121
Figure 68 The Accuracy Results After Dropping G4	122
Figure 69 Screenshot of Fine-tuning Code	124
Figure 70 The Best Parameters	125
Figure 71 Calling the run_model Using Random Forest Classifier	125

LIST OF ABBREVIATIONS

ETL	Extract, Transform, Load
BI	Business Intelligence
DM	Data Mining
SAP	Systems Applications and Products in Data Processing
UI	User Interface
IBM	International Business Machines
AI	Artificial Intelligence
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
WEKA	Waikato Environment for Knowledge Analysis
KNIME	Konstanz Information Miner
SASS	Syntactically Awesome Style Sheets
SPSS	Statistical Package for the Social Sciences
MATLAB	Matrix Laboratory
DBMS	Database Management System
CRM	Customer Relationship Management
API	Application Programming Interface
OLAP	Online Analytical Processing
RDBMS	Relational Database Management System
ERP	Enterprise Resource Planning
PoS	Proof-of-Stake
CSV	Comma Separated Value

SQL	Structured Query Language
JSON	JavaScript Object Notation
HR	Human Resources
EDA	Exploratory Data Analysis

Chapter 1

Introduction

In today's data-driven world, firms aim to acquire a competitive advantage and make educated decisions by leveraging the massive volumes of data at their disposal. With its emphasis on deriving actionable insights from data, business intelligence has become a crucial component of successful enterprises. Data mining, an important approach in business intelligence, enables the finding of useful patterns and trends buried within big databases, enabling businesses to make data-driven choices.

This thesis seeks to investigate data mining tools and techniques in the context of business intelligence, specifically how they might be used to successfully turn raw data into useful information. Organizations may unleash the potential of their data assets and create more informed decision-making processes by employing data mining methods and procedures.

The study is structured into sections to offer a thorough examination of the subject. Part one provides an overview of business intelligence, its function in allowing data-driven decision-making, and its importance in modern enterprises. This section underlines the significance of gaining meaningful insights from data while also highlighting the limits of standard data analysis methodologies.

Part two delves into data mining, examining numerous approaches and algorithms that allow businesses to uncover hidden patterns and trends. It discusses data mining approaches that are routinely used in business intelligence. Furthermore, this part highlights data mining's obstacles and issues, such as data quality, privacy, and scalability.

Part three focuses on data warehousing, which is critical in enabling effective data storage and retrieval for business intelligence applications. It delves into data warehouse design and architecture, as well as the ETL (Extract, Transform, Load) process for integrating and preparing data for analysis.

Part four investigates the integration of data mining techniques into business intelligence operations, building on the foundations built in the previous sections. It emphasizes the advantages of using data mining for business intelligence, such as

better market awareness, consumer segmentation, fraud detection, and predictive analytics. This section also addresses the problems and ethical concerns associated with data mining in the context of business intelligence.

Part five concludes with an application and case study demonstrating the actual application of data mining techniques for business information. The purpose of this thesis is to investigate data mining methods and methodologies for business intelligence, with a particular emphasis on their application to anticipate song popularity on the Spotify platform. The relevance of music streaming services and the necessity for music business experts to understand the elements influencing song popularity led to the selection of this study subject. By analyzing song data with data mining techniques, important insights may be gleaned, easing decision-making processes in areas such as song releases, marketing tactics, and target audience selection.

Several common data mining algorithms were used in this study, including decision trees, support vector machine (SVM) classifiers, K-nearest neighbor, logistic regression, AdaBoost classifiers, and random forest classifiers. These algorithms were chosen for their ability to handle categorization problems as well as their application to predicting song popularity. We want to produce accurate forecasts and acquire deeper insights into the elements influencing song popularity by combining these algorithms.

The major goal of this research is to add to the current literature by demonstrating the actual implementation of data mining tools and techniques in the context of business intelligence. We hope to illustrate the potential benefits and problems of integrating data mining techniques in real-world contexts, notably in the music business, through the case study on song popularity prediction.

We hope to give insights into the elements driving song popularity on Spotify by sharing the findings of various data mining techniques. Furthermore, we use fine-tuning approaches and parameter optimization to improve the accuracy and performance of the data mining technologies we use. Pursuing the best accuracy with the best criteria enhances the accuracy and efficacy of our forecasts.

In the pursuit of accurate and effective predictions, we recognized the importance of fine-tuning our data mining technologies. Fine-tuning involves optimizing the

various hyperparameters associated with the predictive models, enabling us to achieve the best possible performance. By systematically adjusting parameters such as learning rate, regularization, batch size, and activation functions, we aim to strike the optimal balance between underfitting and overfitting.

Through an iterative process, we experimented with different combinations of hyperparameters and evaluated the model's performance based on established criteria such as accuracy, precision, recall, and F1 score. Our objective was to identify the best hyperparameter settings that would maximize the model's predictive power and generalizability. This process involved careful consideration and extensive testing to ensure that our models were robust and capable of making accurate predictions in real-world business scenarios.

Furthermore, the identification of the best parameters was crucial in fine-tuning our models effectively. We explored various feature selection techniques and assessed their impact on the model's performance. By identifying the most relevant and informative features, we aimed to reduce noise and improve the model's overall accuracy and interpretability. Techniques such as forward selection, backward elimination, and regularization were employed to identify the optimal set of features that contributed the most to the predictive power of the model.

The fine-tuning of our data mining techniques and the optimization of hyperparameters and parameters played a vital role in enhancing the accuracy and efficacy of our predictions. This rigorous approach allowed us to uncover meaningful patterns, trends, and insights from the business data, enabling informed decision-making and driving business intelligence.

This study's contributions go beyond the unique case study of music popularity on Spotify. We hope to motivate and educate other scholars and practitioners in using data mining tools and methodologies to extract useful insights from their own datasets by demonstrating their practical applicability. Furthermore, the purpose of this research is to bridge the theoretical and practical divide by proving the potential of data mining approaches to improve decision-making processes in the business intelligence area.

Finally, this thesis discusses the relevance of data mining tools and methodologies in the context of business intelligence. We hope to contribute to the literature by

demonstrating the practical use and benefits of data mining in a real-world situation by concentrating on the specific issue of forecasting song popularity on Spotify. The addition of numerous data mining techniques and the adjustment of their parameters improves the accuracy and dependability of the forecasts even more. Finally, the goal of this study is to improve knowledge and use of data mining tools and methodologies for better decision-making in business intelligence situations.

Chapter 2

Literature Review

Businesses that employ data mining may gain a competitive advantage, obtain a better understanding of their customers, gain greater control over their everyday operations, enhance client acquisition rates, and identify new business chances. Many industries will benefit from data analytics in various ways. Certain firms are looking for the best ways to acquire new customers, while others are looking for new marketing methods and system updates. Because of the data mining process, businesses have the knowledge and understanding to make judgments, review their data, and advance.

A thorough search was carried out using keywords like "data mining tools," "data mining techniques," "business intelligence," and other relevant topics. Relevant peer-reviewed publications, conference papers, and books for their contributions to the field of data mining in the context of business intelligence were examined. The chosen literature provides an overview of major data mining tools and methodologies, as well as an examination of their use in real-world business contexts.

These are some of the books and articles that I have read:

- Data Virtualization for Business Intelligence Systems: Revolutionizing Data Integration for Data Warehouses (*Morgan Kaufmann Series on Business Intelligence*) Kağıt Kapak – Resimlendirilmiş, 25 July 2012:
 - ⇒ Rick van der Lans's most recent book, Data Virtualization for Business Intelligence Systems: Revolutionizing Data Integration for Data Warehouses, has done the IT industry a great service. Rick approaches data virtualization from a variety of perspectives. He provides a high-level explanation for data virtualization, notably in historical terms in connection to technology in general and data implementations in particular. Rick also provides a full examination of data virtualization functioning, as well as screenshots from well-known solutions. He also forecasts the future of data virtualization, taking into consideration his own projections as well as those of the CTOs of three distinct data virtualization software businesses. A companion publication to this one is Data Virtualization: Moving Beyond

Traditional System Integration to Achieve Flexibility, which contains 10 in-depth data virtualization case studies from a range of companies.

- Integration of Data Mining in Business Intelligence Systems (*Ana Azevedo (Algoritmi R&D Center/University of Minho, Portugal & Polytechnic Institute of Porto/ISCAP, Portugal) and Manuel Filipe Santos (Algoritmi R&D Center/University of Minho, Portugal) September, 2014:*
 - ⇒ This book gives a thorough examination of how data mining techniques are incorporated into business decision-making tools. It is a useful resource for decision-makers, researchers, students, developers, and professionals who want to learn how data mining is used in business systems. Business analytics, business information systems, competitive intelligence, data analysis, data discovery, decision support systems, and programming languages are among the subjects covered in the book. It provides cutting-edge research and pertinent concepts in data exploration and analysis, making it a great resource for a wide range of academic subjects.
- Web Data Mining and Applications in Business Intelligence and Counter-Terrorism (*Bhavani Thuraisingham) June 26, 2003 by CRC Press:*
 - ⇒ "Web Data Mining and Applications in Business Intelligence and Counter-Terrorism" provides a thorough and concise explanation of web mining, with an emphasis on customer relationship management (CRM) as well as security and counter-terrorism applications. As a program manager for data and application security, the author stresses the practical implementation of web mining tools and techniques in real-world circumstances. The book discusses how firms may identify and handle web-based opportunities and dangers. Businesses may improve consumer interactions, increase revenue, and uncover present and prospective hazards by harnessing web-based data. Furthermore, the book emphasizes the importance of web mining in intelligence operations, underlining how these same tools may be used to battle the ever-present threat of terrorism.

- Applied Data Mining for Business Intelligence (*Niels Arnth-Jensen*) 2006:
 - ⇒ The primary goal of this project is to collaborate with gatetrade.net to create innovative Data Mining solutions for certain data processing requirements. Several techniques were built and assessed, including k-nearest neighbors, Fuzzy C-Means, and Unsupervised Fuzzy Partitioning - Optimal Number of Clusters. Furthermore, 10 distinct cluster validity criteria were developed and evaluated in order to identify the appropriate number of segments, with six segments chosen as the best fit for inclusion in the data sets. A Data Processing Framework was developed to accommodate the data formats used by gatetrade.net.

- Business Intelligence and Data Mining (*Anil K. Maheshwari, PhD, 2015*):
 - ⇒ The author's goal was to write an exciting and instructive book that will appeal to a wide readership. This book gives a thorough review of vital material, backed by real-world examples, to entice readers to investigate the topic of data mining. The author arranged the chapters to correspond with a standard one-semester graduate course, drawing on years of experience in both academia and the IT sector. Each chapter opens with a case study based on a real-world situation, and a case study runs throughout the book. Various data mining techniques are comprehensively discussed, including decision trees, regression, artificial neural networks, and cluster analysis. Important subjects covered in the book include big data, text mining, and web mining. An introduction is also offered to help those who are new to data modeling.

Conclusion:

This overview of the literature emphasizes the importance of data mining tools and techniques in the context of business intelligence. The examples chosen highlight the breadth of available tools, such as SAS, RapidMiner, and IBM SPSS Modeler, as well as the efficiency of approaches like as clustering, classification, association rule mining, and text mining. Businesses that incorporate data mining successfully into their business intelligence ecosystems may get important insights and make educated choices. Organizations may use data mining tools and techniques to identify hidden

patterns, trends, and linkages in their data, resulting in increased operational efficiency, improved customer interactions, and better strategic planning.

The studied literature demonstrates the enormous range of data mining technologies accessible, each with its own set of features and capabilities. These technologies allow firms to gather meaningful insights from a variety of data sources and conduct advanced analytics. Furthermore, the efficacy of several data mining approaches such as clustering, classification, association rule mining, and text mining has been emphasized. Organizations may use these strategies to segment their client base, anticipate future events, improve business processes, and extract valuable information from unstructured textual data.

Successful data mining integration with business intelligence may have a revolutionary effect on enterprises. It helps them to acquire a competitive edge through the discovery of hidden possibilities, risk mitigation, and data-driven decision making. Furthermore, data mining contributes in the identification of market trends, the comprehension of consumer behavior, the optimization of marketing tactics, and the general improvement of corporate performance.

Finally, the studied literature underlines the critical importance of data mining tools and methodologies in business intelligence. Organizations that use the potential of these tools and methodologies may acquire significant insights from their data, resulting in improved decision-making, higher operational efficiency, and increased market competitiveness. As the area of data mining evolves, organizations must remain up to date on the newest tools and techniques in order to leverage the benefits of data mining for their business intelligence operations.

Chapter 3

Business Intelligence

Business intelligence (BI) is a set of methodologies and technologies used by businesses to convert data into actionable insights. BI includes a wide range of operations, including as data gathering, storage, and analysis from a variety of sources, such as transactional databases, customer interactions, market research, and social media. The goal of business intelligence is to assist firms in making better decisions by providing a comprehensive view of their operations and the variables that drive them.

Businesses may acquire and analyze data from a number of sources, including structured data from databases, unstructured data from social media and other online sources, and semi-structured data from sensors and other connected devices, using BI systems. The data is then translated into a format that business users can readily understand and utilize, such as dashboards, reports, and visualization tools.

BI may be used to find new company possibilities, improve operational efficiency, optimize marketing efforts, and anticipate future trends, among other things. BI is frequently connected with data analytics, but it is a larger word that encompasses a wide variety of data management and decision-making operations.

3.1.The Importance of Business Intelligence

Business intelligence (BI) is becoming an increasingly vital tool for businesses of all sizes and sectors. By giving a complete perspective of their operations and the elements that drive them, BI enables organizations to make data-driven choices.

There are various reasons why business intelligence is so important:

- ***Improved decision-making:*** BI provides a plethora of data that may be utilized to inform decision-making at all organizational levels. Businesses may obtain a better knowledge of their consumers, operations, and markets by evaluating data from many sources such as transactional databases, customer interactions, market research, and

social media. This can help them make better resource allocation decisions, find new possibilities, and enhance efficiency.

- ***Increased efficiency:*** Business intelligence may assist companies in identifying inefficiencies and bottlenecks in their processes, allowing them to take remedial action and enhance overall efficiency. A BI system, for example, may disclose that a specific process is taking longer than it should, or that certain resources are being overused. Businesses may enhance their efficiency and decrease expenses by addressing these challenges.
- ***Competitive advantage:*** BI may help firms gain a competitive advantage by giving a deeper insight of their markets and consumers. Businesses may find trends and patterns in data from a number of sources to inform their marketing and sales strategy. They may also utilize BI to uncover new possibilities and create new goods or services to fulfill their consumers' evolving wants.
- ***Improved customer satisfaction:*** BI may assist companies understand what their consumers want and how to better satisfy their demands. Businesses may uncover chances to enhance their goods, services, and overall customer experience by studying consumer data. This can result in enhanced client satisfaction and loyalty, which can fuel long-term development and success.
- ***Enhanced risk management:*** By offering a holistic perspective of their operations, BI can assist firms in identifying and mitigating possible hazards. A BI system, for example, may disclose that a specific supplier is routinely delivering late, which might impair production and result in lost sales. Businesses that detect this risk early on can take actions to reduce it and avert future interruptions.

In conclusion, BI is a critical tool for enterprises of all sizes and sectors. It enables organizations to make better decisions, increase efficiency, gain a competitive edge, improve customer happiness, and manage risk more effectively. Businesses may position themselves for long-term success and development by investing in BI.

3.2. Concepts and Applications of Business Intelligence

Business intelligence (BI) refers to a variety of ideas and tools used by firms to translate data into actionable insights. Some popular BI principles are as follows:

- ***Data warehousing:*** A data warehouse is a centralized collection of data designed for quick querying and analysis. Large volumes of structured data from many sources, such as transactional databases, log files, and survey findings, are generally stored in data warehouses.
- ***Data mining:*** Data mining is the process of discovering patterns and correlations in data using algorithms and statistical models. It is an effective method for detecting trends and forecasting future results.
- ***Data visualization:*** The process of expressing data in a graphical or visual manner, such as charts, graphs, and maps, is known as data visualization. By making data more palatable, data visualization tools may help consumers better comprehend and analyze data.
- ***Dashboards:*** A dashboard is a graphical display of essential data and performance indicators. Dashboards may be used to monitor and track a company's progress, spot patterns, and make sound choices.
- ***Reporting:*** Reporting is the process of creating and disseminating reports that give information about a company's operations and performance. Reports can be created on a daily, weekly, or monthly basis, or on an ad hoc basis as needed.

- ***Predictive analytics:*** Machine learning algorithms are used in tools to anticipate future events based on previous data. Predictive analytics may assist companies in making educated decisions regarding future activities.
- ***Scorecards:*** Visualizations that measure an organization's progress toward certain goals. Scorecards may be used to assess an organization's performance and highlight opportunities for improvement.
- ***Spatial analysis:*** Tools for analyzing and visualizing data in respect to geographical locations. Spatial analysis may help you recognize trends and patterns in data that are specific to an area.
- ***Mobile BI:*** BI applications that are optimized for use on mobile devices. Mobile BI allows users to access data and insights on the go, regardless of their location.
- ***Collaboration and sharing:*** Features that enable users to collaborate on data analysis projects and share their findings with others. Collaboration and sharing can assist teams in remaining aligned and making better decisions as a group.

Organizations may use a variety of BI technologies to analyze data and derive insights. Typical instances include:

- ***Sales analysis:*** BI may be used to evaluate sales data in order to detect patterns, estimate future demand, and enhance pricing and marketing tactics.

- ***Customer analysis:*** BI may be used to evaluate consumer data in order to detect trends and preferences, enhance customer segmentation, and create focused marketing efforts.
- ***Supply chain analysis:*** BI may be used to evaluate supply chain data in order to detect bottlenecks, optimize inventory levels, and strengthen supplier relationships.
- ***Financial analysis:*** BI may be used to analyze financial data in order to detect patterns, estimate future performance, and enhance financial planning and budgeting.
- ***Operations analysis:*** BI may be used to evaluate data from multiple operational processes to find inefficiencies, enhance performance, and save costs.

To summarize, business intelligence (BI) is a powerful tool that businesses can utilize to turn data into actionable insights and inform decision-making. It comprises a diverse set of concepts and applications that may be used to a wide range of corporate demands and objectives.

3.3. Business Intelligence for Better Decision

Business intelligence (BI) is a useful tool for firms that wish to use data to make better choices. Businesses can use BI systems to collect and analyze data from many sources, including as transactional databases, consumer interactions, market research, and social media. Business intelligence insights may be utilized to inform decision-making at all levels of a business, from strategic planning to operational changes.

BI may help firms make better decisions in a variety of ways:

- ***Comprehensive view of operations:*** By acquiring and analyzing data from many sources, BI gives a comprehensive perspective of an organization's activities. This can assist decision-makers in seeing the

large picture and seeing patterns and trends that may be missed from a limited view.

- ***Improved accuracy:*** BI systems analyze data using statistical models and algorithms, which can improve decision-making accuracy. Organizations may make better informed decisions that are less prone to prejudice by relying on facts rather than gut impulses or subjective judgments.
- ***Faster decision-making:*** By delivering real-time data and allowing users to instantly produce reports and dashboards, BI systems may assist speed up the decision-making process. This can assist firms in responding more swiftly to market developments or client wants.
- ***Enhanced risk management:*** By offering a holistic perspective of their operations, BI may assist firms in identifying and mitigating possible hazards. A BI system, for example, may disclose that a specific supplier is routinely delivering late, which might impair production and result in lost sales. Businesses that detect this risk early on can take actions to reduce it and avert future interruptions.

In conclusion, BI is a crucial tool for firms that wish to make better data-driven decisions. It enables firms to collect and analyze data from many sources, resulting in a holistic perspective of their operations and the forces that drive them. Organizations may use BI to increase the accuracy of their decision-making, make choices faster, and manage risk better.

3.4. Business Intelligence Tools

There are several business intelligence technologies available, including data warehouses, data mining software, and dashboarding tools. Some common examples of business intelligence tools are:

- **Tableau:** Tableau is a data visualization and business intelligence platform that allows users to effortlessly build interactive dashboards and reports. It is user-friendly, with a drag-and-drop interface and a large selection of pre-built interfaces to various data sources. Users can quickly and simply analyze, display, and share data insights using Tableau, allowing them to make educated decisions based on data-driven insights. Tableau is very flexible, with several formatting and stylistic choices, as well as the ability to generate custom computations and interactive dashboards and reports.
- **Power BI:** Microsoft Power BI is a cloud-based business intelligence and analytics tool that enables users to connect to, view, and analyze data from many sources. It comes with a variety of tools and capabilities for producing interactive dashboards, reports, and charts, as well as real-time data streaming and natural language queries. Power BI is intended to be simple to use, with a straightforward interface and a variety of pre-built interfaces to common data sources. It also has collaboration and sharing tools, which allow users to effortlessly share their ideas with others and collaborate on data analysis projects.
- **SAP BusinessObjects:** SAP BusinessObjects is a collection of business intelligence (BI) tools for analyzing and reporting on data from various sources. It comprises a variety of applications, including as reporting, data visualization, and predictive analytics tools. SAP BusinessObjects' purpose is to give customers with a comprehensive set of BI tools that will assist them in making educated decisions based on data insights. It is intended to be simple to use, with a number of interactive features and a straightforward, intuitive UI.
- **IBM Cognos:** IBM Cognos is a set of business intelligence (BI) and performance management tools for analyzing, visualizing, and reporting on data from various sources. It comprises a variety of applications, including as reporting, data visualization, and predictive

analytics tools. With a clear, intuitive interface and a number of interactive capabilities, IBM Cognos is meant to be simple to use. It is also very scalable and can be tailored to an organization's unique requirements. IBM Cognos' mission is to give customers with a comprehensive range of BI capabilities that will assist them in making educated decisions based on data insights.

These are just a few examples of the numerous business intelligence technologies accessible. The instruments that an organization use will be determined by its unique demands and aims.

3.5.Business Intelligence Skills

Business intelligence (BI) skills are a collection of technical and analytical abilities used to gather, process, and analyze data in order to inform organizational decision-making. BI specialists are in charge of planning and implementing systems for gathering and storing data from diverse sources, as well as establishing and maintaining the infrastructure required for data analysis and reporting. They employ a variety of tools and techniques to translate data into insights that may be used to enhance business operations, guide strategic planning, and optimize performance, such as data visualization, statistical analysis, and machine learning.

Among the specialized BI talents are:

- Data analysis
- Data visualization
- Data mining
- Dashboarding
- ETL (Extract, Transform, Load)
- Reporting
- Data management
- Data warehousing
- Predictive analytics

- Machine learning
- Statistical analysis
- Data modeling
- Business process management
- Project management
- Communication and presentation skills

Overall, BI capabilities are critical for any firm seeking to make data-driven choices and achieve a competitive edge through data utilization.

Chapter 4

Data Mining

The practice of uncovering patterns and correlations in huge datasets in order to extract relevant and potentially actionable information is known as data mining. It is analyzing and extracting insights from data using algorithms and statistical models with the purpose of identifying hidden trends, patterns, and correlations that may not be immediately obvious.

Data mining is frequently utilized in a wide range of industries, including business, finance, healthcare, and scientific research, to assist businesses in making informed decisions, increasing efficiency and production, and gaining a competitive edge. It is an interdisciplinary area that incorporates approaches from computer science, statistics, and machine learning. It employs tools and techniques such as machine learning algorithms, data visualization, and data manipulation.

4.1.About Data Mining

4.1.1. The Origins of Data Mining: The practice of studying and extracting patterns from massive data sets in order to unearth important insights and information is known as data mining. It is a subject with origins in computer science and statistics that has changed greatly over time as technology has progressed and the amount of data accessible for analysis has risen dramatically.

Data mining has its roots in the 1960s, when academics began utilizing computers to analyze and analyse massive volumes of data. Early data mining approaches included simple statistical analysis and the use of fundamental algorithms such as decision trees and clustering. The discipline of artificial intelligence (AI) began to emerge in the 1980s, and researchers began to use machine learning techniques to evaluate data.

The advent of the internet and the abundance of electronic data in the 1990s resulted in substantial breakthroughs in data mining. New methodologies and tools for analyzing and extracting insights from huge

and complicated data sets were created, and data mining became a key tool for firms and organizations seeking a competitive edge through data-driven decision making. Data mining is now employed in a wide range of industries, including banking, marketing, healthcare, and scientific research.

- 4.1.2. **Benefits of Data Mining:** The act of examining big data sets to identify patterns, trends, and correlations that might yield important insights and information is known as data mining. As the amount of data created has risen tremendously in recent years, it has become an increasingly crucial tool for businesses and organizations.

Improved decision making is one of the primary advantages of data mining. Organizations can find trends and patterns that may not be immediately evident by analyzing massive volumes of data. This can assist individuals in making better-informed judgments based on data-driven insights rather than preconceptions or gut reactions. Data mining, for example, might be used by a shop to study client purchase habits and uncover trends in consumer behavior, such as which goods are most popular among various demographics. This data might then be utilized to help guide marketing and sales tactics, as well as product development.

Organizations can also benefit from data mining by discovering inefficiencies and opportunities for development. Data mining, for example, may be used by an airline to examine flight data and detect trends in delays or cancellations. This data might then be utilized to optimize routes and timetables, leading to fewer delays and cancellations and a more efficient overall operation.

Another advantage of data mining is its capacity to improve client experiences. Organizations may obtain a better knowledge of their consumers' requirements and preferences by studying customer data. This allows them to better personalize their products and services to their consumers' demands, resulting in enhanced customer satisfaction and loyalty.

Data mining may also assist businesses in increasing profits by uncovering new possibilities and making better judgments. A financial organization, for example, may employ data mining to examine market patterns and find possible investment possibilities. Organizations may raise their chances of success and improve their bottom line by making data-driven decisions.

Aside from these advantages, data mining may assist firms in improving risk management by detecting and managing possible hazards. A bank, for example, may employ data mining to discover trends of fraudulent behavior and put preventative measures in place. Organizations may safeguard themselves and their customers by proactively recognizing and resolving possible hazards.

Overall, data mining has the potential to give companies with useful insights that will assist them in making better decisions, improving operations, and increasing profitability. It is a vital instrument utilized in a wide range of industries, including banking, healthcare, marketing, and scientific research.

4.1.3. Disadvantages of Data Mining: While data mining offers various advantages, it also has certain drawbacks. One of the primary downsides of data mining is the possibility of privacy violations. Because data mining entails the collecting and analysis of enormous volumes of personal data, there is a risk that this information may be abused or mismanaged. For example, if a company collects information about a person's shopping patterns and then utilizes that information to target them with targeted adverts, it may be considered an invasion of privacy. To alleviate these worries, firms must be clear about their data gathering processes and have solid data protection rules.

Another consequence of data mining is the possibility of ethical problems. Data mining algorithms can be biased if the data on which they are taught is skewed, resulting in unjust or discriminating results.

For example, if a data mining algorithm is trained on data from a single demography, it may not effectively represent other demographics, resulting in biased results. Organizations must be aware of this possible bias and take measures to mitigate it.

Data mining might also suffer from technical restrictions. Data mining is dependent on the quality and accuracy of the data being examined, and if the data is inadequate or wrong, the data mining process may produce faulty findings. Furthermore, data mining may be complicated and need specific skills and expertise, which can be a barrier for some firms.

Data mining's complexity may potentially be a drawback. Data mining employs complicated algorithms and statistical approaches that might be difficult to comprehend and execute. For companies without the appropriate skills or resources, this might be a hurdle.

Finally, data mining may be an expensive venture. Implementing and maintaining a data mining system takes a large amount of time and money, which can be difficult for enterprises with minimal resources.

In conclusion, while data mining may bring useful insights and knowledge, it also has drawbacks such as privacy problems, ethical challenges, technical constraints, complexity, and expense. When determining whether or not to install a data mining system, firms must carefully evaluate these issues.

Table 1

Pros & Cons of Data Mining

Pros	Cons
Improved decision making	Privacy concerns
Increased efficiency	Ethical issues
Enhanced customer experiences	Technical limitations
Increasing profitability	Complexity
Improved risk management	Cost

4.1.4. The Usage of Data Mining: Data mining is a powerful tool that can be used for a wide range of purposes. Some of the ways in which data mining is commonly used include:

- ***Market analysis:*** Data mining may be used to find possible opportunities and assess industry trends. Data mining, for example, might be used by a shop to study client purchase habits and uncover trends in consumer behavior, such as which goods are most popular among various demographics. This data might then be utilized to help guide marketing and sales tactics, as well as product development.
- ***Fraud detection:*** Data mining may be used to discover fraudulent behavior patterns, such as fraudulent credit card transactions or insurance claims. Organizations can detect unexpected trends or anomalies that may suggest fraudulent behavior and take preventative measures by analyzing vast volumes of data.
- ***Customer relationship management:*** Data mining may be used to examine consumer data and obtain a better knowledge of their wants and needs. This data may be utilized to better adapt products and services to consumers' demands, resulting in enhanced customer satisfaction and loyalty.
- ***Predictive modeling:*** Data mining may be used to create predictive models that anticipate future outcomes or trends. A healthcare company, for example, may employ data mining to examine patient data and create a prediction model to determine which patients are at high risk of getting specific diseases. This data might then be utilized to develop preventative actions to lower the risk of these illnesses.
- ***Text mining:*** Data mining techniques may be used to extract important insights from text data, such as social media postings or consumer reviews. Text mining, for example, might be used by a corporation to evaluate consumer evaluations of their goods and uncover common

themes or trends that could be used to guide product development or marketing tactics.

- **Optimization:** Data mining may be used to uncover patterns and trends in data, which can then be utilized to assist businesses optimize their operations. Data mining, for example, may be used by an airline to examine flight data and detect trends in delays or cancellations. This data might then be utilized to optimize routes and timetables, leading to fewer delays and cancellations and a more efficient overall operation.

Overall, data mining may be utilized for a variety of tasks such as market analysis, fraud detection, customer relationship management, predictive modeling, text mining, and optimization. It is a useful tool that is utilized in many fields, including banking, healthcare, marketing, and scientific research.

4.2.Top Data Mining Techniques

Data mining is the process of discovering patterns and trends in large datasets in order to extract valuable insights and make informed decisions. There are several techniques used in data mining, including:

- 4.2.1. **Decision Trees:** In data mining, decision trees are a common and successful way for developing predictive models. They are a form of tree-based method that divides a dataset into smaller and smaller subgroups depending on different criteria. The algorithm picks the feature with the biggest information gain at each split, which is a measure of how effectively the split will minimize uncertainty about the target variable. The method is repeated until the subsets are pure, which means they include only one class or label.

The following code might be used to train a decision tree model:

```
from sklearn import tree

# Train the model
model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
```

Figure 1. Decision tree code example

X_train and **y_train** are the training data's features and labels, respectively, while **X_test** is the test set's feature data. The **fit** approach trains the decision tree model on the training data, while the 'predict' method provides predictions on the test set using the learned model.

Decision trees offer several advantages, including the ease with which they can comprehend and depict numerical and categorical data. They can, however, be prone to overfitting, especially if the tree is allowed to grow too deep. To minimize overfitting, measures such as pruning or limiting the maximum depth of the tree are commonly used.

4.2.2. Adaboost (Adaptive Boosting): AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that combines several weak classifiers to form a strong classifier.

Here's the AdaBoost classifier code from scikit-learn:

```
from sklearn.ensemble import AdaBoostClassifier

# Create an AdaBoost classifier with Decision Tree as the base estimator
adaboost_clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1), n_estimators=50, random_state=42)

# Fit the model on your training data
adaboost_clf.fit(X_train, y_train)

# Predict on your test data
y_pred = adaboost_clf.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
```

Figure 2. Adaboost classifier code example

In the above code, `AdaBoostClassifier` is imported from the `sklearn.ensemble` module. We create an instance of the classifier with **`DecisionTreeClassifier(max_depth=1)`** as the base estimator, which means each weak classifier will be a decision tree with a maximum depth of 1. We also set the number of estimators to 50, which means we will train 50 weak classifiers. Finally, we fit the model on the training data, make predictions on the test data, and calculate the accuracy of the model using **`accuracy_score`** function from the **`sklearn.metrics`** module.

4.2.3. Support Vector Machine (SVM): Support Vector Machines (SVM) is a strong technique for classification and regression analysis in data mining and machine learning. SVM is a supervised learning technique that finds the hyperplane in a high-dimensional space that best separates data points from distinct classes.

```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an SVM classifier with a linear kernel
clf = svm.SVC(kernel='linear')

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Figure 3. Support vector machine code example

In this example, we first split the data into training and test sets using scikit-learn's **`train_test_split()`** function. Then use the **`svm.SVC()`** function to create her SVM classification with a linear kernel. Train a classifier on the training data using the **`fit()`** method, and then make predictions on the test data using the **`predict()`** method. Finally, use scikit-learn's **`Accuracy_score()`** function to assess the accuracy of the classifier.

4.2.4. Logistic Regression: Logistic regression is a widely used statistical technique in data mining that is used to predict binary outcomes (1/0, yes/no, true/false) based on a set of independent variables. It is a type of supervised learning algorithm that is often used in classification problems. The goal of logistic regression is to find the best set of coefficients associated with input characteristics of binary output variables.

The logistic regression model is a linear model for binary classification, in which the log-odds of the true class is modeled as a linear combination of the features (predictors) and a bias term. The log-odds is also known as the logit, which is a measure of how likely an instance belongs to the positive class. The logistic function, also known as the sigmoid function, is used to transform the logit into a probability between 0 and 1.

Here's an example of logistic regression implemented in Python using the scikit-learn library:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# load your data
X, y = some_data, some_labels

# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# create the logistic regression model
model = LogisticRegression()

# train the model on the training data
model.fit(X_train, y_train)

# make predictions on the test data
y_pred = model.predict(X_test)

# evaluate the model's performance
accuracy = model.score(X_test, y_test)
print("Accuracy: ", accuracy)
```

Figure 4. Logistic regression code example

In this example, we first load the data and divide it into training and test sets, using the test set to evaluate the performance of the model. Then we created a typical logistic regression model and trained it on the training data using the **fit()** method. Then we use the **predict()** method to predict the test

data and evaluate the performance of the model using the `score()` method, which returns the accuracy of the model.

It is worth noting that this is a simple example of logistic regression. Here the model is trained and evaluated with a single-learn test split on the data. In practice, more complex model selection and evaluation techniques are often used, such as k-duplicate cross-validation and grid search for hyperparameter tuning. You can also use regularization techniques such as L1 and L2 to prevent overfitting and increase the generalizability of the model.

4.2.5. K-Nearest Neighbors (KNN): K-nearest neighbors (KNN) is a widely used supervised learning algorithm in data mining that is used for classification and regression problems. It is a type of instance-based learning algorithm, which means that it memorizes the training instances and uses them to make predictions on new instances. The basic idea behind KNN is to find the k-nearest training instances to a new instance and predict the output based on the majority class or mean value of the k-nearest neighbors.

In classification problems, the algorithm finds the k-nearest training instances to a new instance and assigns the majority class among the k-nearest neighbors to the new instance. In regression problems, the algorithm finds the k-nearest training instances to a new instance and assigns the mean value among the k-nearest neighbors to the new instance.

Here's an example of KNN implemented in Python using the scikit-learn library:

```

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# load your data
X, y = some_data, some_labels

# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# create the KNN model with k = 3
model = KNeighborsClassifier(n_neighbors=3)

# train the model on the training data
model.fit(X_train, y_train)

# make predictions on the test data
y_pred = model.predict(X_test)

# evaluate the model's performance
accuracy = model.score(X_test, y_test)
print("Accuracy: ", accuracy)

```

Figure 5. K-nearest neighbors code example

4.2.6. Random Forest Classifier: Random forest classifier is a type of ensemble learning technique that builds a large number of decision trees and aggregates their predictions to produce the final output. Each tree is trained on a different subset of data and a different subset of features. The output of the random forest classifier is determined by taking the output modes of the individual decision trees.

Here is sample code for training a random forest classifier in Python using scikit-learn:

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Random Forest classifier with 100 trees
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
rf.fit(X_train, y_train)

# Evaluate the model on the testing data
accuracy = rf.score(X_test, y_test)

```

Figure 6. Random forest classifier code example

4.3.Tools Used In Data Mining

Data mining tools are software applications that help companies extract valuable insights from large amounts of data. These tools use advanced algorithms and statistical techniques to detect patterns, relationships, and insights that can be used to improve business processes and make more informed decisions. Some popular data mining tools are WEKA, RapidMiner, KNIME, Orange and Rattle. These tools can be used for a variety of tasks such as classification, clustering, association rule extraction, and anomaly detection. It also has various data visualization and preprocessing features.

Some of these tools are open source and free to use, while others are commercial and require a license. Many of these tools also integrate with other technologies, such as big data platforms and machine learning libraries, turning them into versatile solutions for data mining and analytics.

Table 2

Data Mining Tools

Tool Name	Description	Type	Platform
WEKA	A collection of machine learning algorithms for data mining tasks. It includes tools for data preprocessing, visualization, and modeling. It can be used through a GUI or through a command-line interface.	Open-Source	Java
RapidMiner	A data science platform that includes tools for data preparation, modeling, evaluation, and deployment. It also offers integration with big data platforms and other tools such as R and Python.	Commercial	Windows, Linux, Mac
KNIME	A data integration, transformation, and analysis platform that allows users to visually design and execute workflows. It includes a wide range of pre-built	Open-Source	Windows, Linux, Mac

	nodes for data mining and machine learning.		
Orange	A data visualization and analysis tool that includes features for data preprocessing, statistical modeling, and machine learning. It also includes a feature for interactive data visualization.	Open-Source	Windows, Linux, Mac
Rattle	A data mining tool for R that provides a GUI for data preprocessing and visualization, as well as a wide range of modeling and evaluation techniques.	Open-Source	Windows, Linux, Mac

This table is not complete, there are many other data mining tools available. Other notable data mining tools include SASS, SPSS, MATLAB, and the scikit-learn library in Python.

4.4.The Stages of Data Mining

Data mining is a process that involves several stages to extract valuable insights from large datasets. These stages include:

- ***Business understanding:*** This is the first and most important stage, in which the data mining project is defined and goals are set. During this stage, the firm determines the precise business problem that needs to be solved and how data mining may help.
- ***Data understanding:*** The organization obtains and investigates the data that will be used for the data mining project during this stage. This entails gathering, confirming, and cleansing data, as well as comprehending its properties and the correlations between various factors.

- ***Data preparation:*** This stage entails converting and preparing the data in preparation for data mining analysis. This can include tasks like dealing with missing data, detecting outliers, and selecting features.
- ***Modeling:*** Once the data has been prepared, it may be utilized to create a model from which insights can be extracted. This step entails identifying patterns and links in the data using various data mining techniques such as classification, clustering, and association rule mining.
- ***Evaluation:*** The model produced in the previous step is tested in this stage to determine its correctness, robustness, and utility. This might involve tasks like cross-validation and evaluating performance indicators.
- ***Deployment:*** If the model is found to be suitable, it may be deployed in a production setting and used to generate predictions or find trends in fresh data. To ensure the model's continuous functioning, the company should also implement monitoring and maintenance methods.
- ***Review:*** This is a continual process that entails assessing the model's performance and making modifications as needed to increase its accuracy and usefulness.

It is critical to keep constant contact with business stakeholders throughout the data mining process, as their comments and input may assist to ensure that the data mining process is aligned with the business objectives.

4.5.The Main Data Mining Users

Data mining is the process of collecting useful insights and information from massive amounts of data. It is used in a wide range of industries and by a variety of users. The primary customers of data mining may be divided into four categories: corporations, researchers, government organizations, and people.

Data mining allows businesses to obtain a competitive advantage by discovering patterns and trends in their data. This may be utilized to better target customers, optimize marketing efforts, boost efficiency, and make better strategic decisions. Data mining is used by businesses in a variety of industries, including retail, banking, healthcare, and manufacturing, to obtain insights into consumer behavior, discover product trends, detect fraudulent activity, and cut costs. Retail organizations, for example, employ data mining to evaluate client purchase behaviors and find popular items, whereas banks and financial institutions use it to detect fraudulent activity and identify possible dangers.

Data mining is used by researchers to evaluate massive amounts of data for scientific or academic objectives. This might involve evaluating genetic data patterns to find possible drug targets, mining social media data to understand public opinion, or analyzing meteorological data patterns to enhance weather forecasting. Data mining is also used to extract insights from complicated data in domains such as bioinformatics, natural language processing, and image processing.

Data mining is used by government organizations to detect patterns and trends in huge amounts of data in order to influence policy choices. They can, for example, utilize data mining to detect high-crime areas and distribute resources appropriately, or to examine data on healthcare usage and identify possible faults in the healthcare system. Data mining may also be used to discover fraud, waste, and abuse in government programs, as well as to enhance public service delivery.

Data mining is also used by individuals to get insights into their own data. People use data mining, for example, to get insights into their own financial data in order to make better financial decisions, or to track their fitness and health statistics in order to enhance their general well-being.

All of these user groups rely on data mining to extract important insights from massive amounts of data and make better decisions based on that knowledge. Data mining is a continually growing discipline, with new approaches and technologies being created to assist users in extracting even more important insights from their data. Data mining is anticipated to become much more common in the future as the amount of data created continues to expand at an unparalleled rate.

Data mining is primarily used by corporations, researchers, government organizations, and people to extract useful insights and information from vast volumes of data to make better decisions. Data mining is a strong tool that can be utilized in a variety of sectors, and it is expected to become much more relevant in the future as the amount of data created increases.

4.6. Real-Life Data Mining Projects Examples

Data mining is a process used to extract valuable insights and knowledge from large sets of data. It is used in a wide range of industries and for a variety of purposes. Some of the most common real-life data mining projects include:

- ***Customer Segmentation:*** Businesses utilize data mining to categorize their consumers based on their behavior and demographics. This enables them to better target their marketing efforts and increase client retention.
- ***Fraud Detection:*** Data mining is used by banks and financial organizations to detect fraudulent activity and identify possible problems. Analyzing client transactions to uncover strange trends and questionable activity is one example.
- ***Product Recommendations:*** Data mining is used by retailers to study client purchase behavior and find popular items. This data may be utilized to offer product suggestions to customers, thereby increasing sales and customer happiness.

- ***Healthcare:*** In healthcare, data mining is used to uncover patterns and trends in patient data that can assist improve patient outcomes and save expenses. Data mining can, for example, be used to evaluate electronic health information in order to identify high-risk individuals and focus actions to improve their health.
- ***Network Intrusion Detection:*** In network security, data mining is used to identify and prevent cyber-attacks. This might involve examining network data to spot patterns of activity that suggest an impending assault.
- ***Text Mining:*** In natural language processing, data mining is used to extract insights from enormous amounts of text data. This might involve reviewing customer feedback to find typical complaints or praise or evaluating social media data to understand public sentiment.
- ***Image Processing:*** In image processing, data mining is used to derive insights from enormous amounts of picture data. Analyzing satellite pictures to uncover patterns of land use, for example, or analyzing medical photos to identify potential health risks, are examples of this.
- ***Predictive Maintenance:*** In manufacturing, data mining is used to forecast when equipment may break and plan maintenance appropriately. This can assist to avoid equipment downtime and save money.

These are only a handful of the many applications of data mining in real-world initiatives. Data mining is a powerful technique for extracting meaningful insights from big amounts of data and making better decisions. As the amount of data created continues to expand, data mining is anticipated to become even more significant in a variety of businesses.

4.7.The Value of Data Mining Projects

Data mining initiatives are significant because they enable businesses to extract useful insights and information from vast amounts of data. This data may then be utilized to enhance operations, make better decisions, and create a competitive edge.

One of the primary reasons for the importance of data mining initiatives is that they enable firms to obtain a better knowledge of their consumers. Businesses may segment their clients into distinct categories and focus their marketing efforts more effectively by evaluating customer data. This can assist to enhance client retention and sales.

Data mining initiatives are very helpful in detecting and combating fraud. Data mining is used by banks and financial organizations to detect fraudulent activity and identify possible problems. This can aid in the prevention of financial losses as well as the protection of consumers' personal and financial information.

In healthcare, data mining initiatives are used to find patterns and trends in patient data that can assist improve patient outcomes and minimize expenses. Data mining can, for example, be used to evaluate electronic health information in order to identify high-risk individuals and focus actions to improve their health.

Manufacturing companies employ data mining projects to forecast when equipment may break and arrange maintenance appropriately. This can assist to avoid equipment downtime and save money.

Furthermore, data mining initiatives are becoming increasingly crucial in disciplines such as natural language processing, image processing, and network security in order to extract insights from vast amounts of data. Analyzing social media data to comprehend public opinion, evaluating consumer feedback, analyzing medical pictures to identify potential health concerns, and identifying cyber-attacks are all examples of this.

Overall, data mining initiatives are critical because they enable firms to glean useful insights and information from vast amounts of data. This data may then be utilized to enhance operations, make better decisions, and create a competitive edge. Data mining initiatives will become progressively more

significant in a variety of businesses as the amount of data created continues to expand at an unprecedented rate.

Chapter 5

Data Warehouse

A data warehouse is a centralized store of structured data for reporting and analysis. It is intended to facilitate efficient data querying and analysis and is often used to assist business intelligence (BI) tasks.

Data warehouses are based on a sophisticated database management system (DBMS) designed for searching and analyzing enormous amounts of data. These systems are built to manage high levels of concurrency while also allowing for quick data querying and aggregation.

A data warehouse is distinguished by the fact that it stores data in a structured and ordered manner, frequently employing a schema-on-write approach. This implies that data is cleaned and converted before being placed into the warehouse, allowing for more efficient querying and analysis. A transactional database, such as a customer relationship management (CRM) system, on the other hand, employs a schema-on-read strategy, which implies that data is not modified until it is queried.

A data warehouse also typically stores data in a denormalized form, meaning that data is stored in a more redundant manner in order to improve query performance. This is in contrast to a transactional database, which is typically normalized in order to reduce redundancy and improve data integrity.

Data warehouses are often filled with information derived from a number of sources, including transactional databases, log files, and API requests. This data is regularly extracted, transformed, and loaded (ETL) into the data warehouse, frequently using a batch process. The data in the data warehouse is subsequently made available to users via business intelligence (BI) tools such as dashboards, reports, and OLAP cubes.

Data warehouses are extensively utilized in businesses of all kinds to support a wide range of business operations, including decision making, performance tracking, and trend analysis. They are also frequently utilized to aid data mining and machine learning applications.

One significant advantage of utilizing a data warehouse is that it allows users to access and evaluate data from many sources in a single spot. This is particularly valuable in businesses with vast and complicated data sets because it helps users to obtain insights and make data-driven choices without having to browse different systems and databases.

Another advantage of data warehouses is their ability to facilitate efficient querying and processing of massive amounts of data. Because data warehouses are particularly created for this purpose, they can frequently handle queries and analyses that would be sluggish or impossible to do on a transactional database.

To summarize, a data warehouse is a centralized collection of structured data designed to facilitate efficient searching and analysis. It is commonly used to assist business intelligence tasks and may be fed data from a number of sources. Data warehouses are frequently utilized in businesses of all sizes to support a wide range of business operations, and they may offer considerable advantages in terms of data accessibility and analysis.

5.1.Design Considerations for Data Warehouse

There are several design considerations to consider when building a data warehouse.

Some of the most important ones include:

- ***Data sources:*** The types and sources of data that will be included in the data warehouse must be considered since they will influence the architecture of the ETL process as well as the layout of the data warehouse itself.
- ***Data volume:*** The volume of data kept in the data warehouse should be evaluated since it will affect the size and performance of the data warehouse.
- ***Data granularity:*** The amount of information necessary in the data warehouse should be evaluated since it will have an influence on the design of the data model and the layout of the data warehouse.

- ***Data history:*** The quantity of historical data that must be preserved in the data warehouse should be taken into account, since this will affect the size and complexity of the data warehouse.
- ***Data integration:*** The data warehouse's data integration needs should be examined, including the need to combine data from numerous sources and the extent of data cleansing and transformation necessary.
- ***Data security:*** The data warehouse's security needs, including the need to secure sensitive data and maintain compliance with relevant rules, should be examined.
- ***Performance:*** The data warehouse's performance needs, including the need to allow quick querying and processing of massive amounts of data, should be evaluated.
- ***Scalability:*** The data warehouse's scalability needs should be evaluated, including the need to accommodate the addition of new data sources and the expansion of data volume over time.
- ***User requirements:*** The demands and requirements of the data warehouse's users, including the sorts of analysis and reporting that will be conducted, should be addressed.
- ***Maintenance:*** The data warehouse's continuing maintenance requirements, including the need for frequent updates and backups, should be considered.

Overall, while creating a data warehouse, it is critical to carefully evaluate an organization's business goals as well as its technological capabilities. This will assist guarantee that the data warehouse is suited to the needs of the business and can successfully support decision-making processes.

5.2.Data Warehouse Development Approaches

A data warehouse is a central repository where data from many sources is stored and managed. It is intended to allow efficient data querying and analysis and is often used to support organizational decision-making processes. There are

various techniques to establishing a data warehouse, each with its own set of benefits and drawbacks.

Here are some common approaches:

- ***Top-down approach:*** This method entails developing and constructing the data warehouse in a hierarchical way, beginning with the greatest degree of granularity and moving down to the lowest level. This strategy is often employed when the data warehouse's business needs are well-defined and the data sources are well-understood.
- ***Bottom-up approach:*** This method begins with the smallest degree of granularity and progresses to the greatest level. When the data sources are not well-defined or the business needs for the data warehouse are not well understood, this strategy is often employed.
- ***Hybrid approach:*** This method combines features of both the top-down and bottom-up techniques. When the business needs for the data warehouse are not well known and the data sources are not well-defined, this strategy is frequently utilized.
- ***Incremental approach:*** The data warehouse is built in incremental stages, with additional data sources and capabilities added as needed. When the business needs for the data warehouse are not well known and the data sources are not well-defined, this strategy is frequently utilized.
- ***Data Mart approach:*** This method entails creating tiny, customized data warehouses to serve certain business processes or tasks. Data marts are commonly used as a starting point for a bigger data warehouse endeavor since they are often quicker and faster to develop than a full-scale data warehouse.

There are also several technical approaches to developing a data warehouse, including:

- ***Relational database approach:*** This method includes storing and managing data in the data warehouse using a typical relational database management system (RDBMS). Because RDBMS technology is established and tools and experience are generally available, this strategy is extensively adopted.
- ***Multidimensional database approach:*** This method entails the use of a specific database management system designed for storing and accessing data in a multidimensional format. When the data in the data warehouse has a complicated structure or when the data has to be searched in a way that is not well-suited to a typical RDBMS, this strategy is frequently utilized.
- ***Big data approach:*** This method involves storing and processing data in the data warehouse using specific big data technologies such as Hadoop or Spark. This method is frequently employed when the data in the data warehouse is too massive or complicated for a typical RDBMS to manage.

Developing a data warehouse, regardless of technique, necessitates a large investment of time and resources. Before deciding on a data warehouse development strategy, an organization's business needs and technological capabilities must be thoroughly considered.

5.3.Data Warehouse Architecture

The design and structure of a data warehouse system, including the hardware and software components, data sources, and the ETL process, is referred to as data warehouse architecture. A data warehouse architecture's purpose is to enable efficient querying and processing of huge amounts of data in support of business intelligence (BI) activities.

A data warehouse's architecture refers to the general design and organization of the system. A typical data warehouse architecture includes the following elements:

- ***Data sources:*** These are the systems and databases that will host the data for the data warehouse. Transactional databases, flat files, web logs, and other data sources may be included.
- ***ETL (Extract, Transform, Load) tools:*** ETL tools are used to extract data from diverse data sources, convert it into a format suitable for loading into the data warehouse, then put it into the data warehouse itself.
- ***Data warehouse database:*** This is the database where data from multiple sources is saved and structured so that it can be easily retrieved and analyzed. To arrange the data, a star schema or a snowflake structure may be used.
- ***Data marts:*** These are smaller, subject-specific data warehouses designed to store and analyze data for certain business operations or departments.
- ***BI (business intelligence) tools:*** These are the tools that users utilize to access and evaluate data from the data warehouse. Dashboards, reports, and OLAP cubes are examples of BI tools.
- ***Data lakes:*** These are massive, centralized raw data repositories used for large-scale data storage and processing. Data lakes are frequently used in tandem with data warehouses because they can store and handle massive amounts of data before it is converted and put into the data warehouse.

- ***Cloud-based data warehouses:*** These are cloud-based data warehouses rather than on-premises data warehouses. This enables enterprises to benefit from the cloud's scalability and flexibility without having to invest in infrastructure and upkeep.

In summary, data warehouse architecture refers to the design and structure of a data warehouse system, which includes the hardware and software components, data sources, and the ETL process. It is intended to facilitate the efficient querying and analysis of vast amounts of data in support of BI operations, and it may comprise components such as data sources, an ETL system, a data warehouse database, data marts, BI tools, data lakes, and cloud-based data warehouses.

5.4.Data Sources

The numerous systems and databases that provide the data that is fed into a data warehouse are referred to as data sources. Transactional databases, log files, and API requests are examples of these. Data sources might be on-premises or in the cloud, organized or unstructured.

The types and sources of data that will be included in the data warehouse must be considered since they will influence the architecture of the ETL process as well as the layout of the data warehouse itself. Data from various sources, for example, may need to be processed and cleansed differently in order to be compatible with the data warehouse. Furthermore, when constructing the data warehouse, the data volume and frequency of updates from various sources should be considered.

There are several data sources that may be utilized to create a data warehouse.

Some common types of data sources include:

- ***Transactional databases:*** These are databases that contain transactional data such as client orders, invoicing, and inventory movements. Customer relationship management (CRM) systems, enterprise resource planning (ERP) systems, and point-of-sale (POS) systems are all examples of transactional databases.

- **Log files:** These are files that store information on events and activities that occur within a system or application. Web server logs, application logs, and security logs are all examples of log files.
- **API calls:** These are calls made to application programming interfaces (APIs) to retrieve data or perform a certain action. APIs can be used to extract data from web-based applications or to access data from external systems.
- **Flat files:** These are files that contain data in a simple, tabular format, such as CSV or Excel files. Flat files can be generated by a variety of sources, including manual data entry, data exports, and scraping tools.
- **Social media data:** This is information derived from social media networks such as Facebook, Twitter, and LinkedIn. Customer sentiment and behavior may be gleaned from social media data.
- **Sensor data:** This is data generated by sensors such as Internet of Things devices, industrial equipment, and wearable gadgets. A wide range of physical processes and environments may be tracked and monitored using sensor data.
- **Unstructured data:** Text, photos, and audio files are examples of data that does not have a set format or schema. Unstructured data can be difficult to manage, but it can also contain useful insights and information.

To summarize, data sources are the systems and databases that supply data to a data warehouse. The types and sources of data that will be included in the data warehouse must be considered since they will influence the architecture of the ETL process as well as the layout of the data warehouse itself. Transactional databases, log files, API calls, flat files, social media data, sensor data, and unstructured data are all examples of data sources that may be utilized to create a data warehouse.

5.5.Data Loading Processes

Data loading is the process of transferring data from various sources into a data repository, such as a data warehouse or data lake. There are several key processes involved in data loading, including:

- **Extraction:** This is the process of extracting data from diverse data sources in order to populate the data repository. To get data from the source systems, SQL queries, APIs, or other techniques may be used.
- **Transformation:** This is the process of cleaning and formatting data so that it may be loaded into a data repository. This might include deleting duplicates, rectifying data problems, and applying data types and formatting requirements, among other things.
- **Loading:** This is the process of moving data from data sources to a data repository. This may include employing specialized tools and methods, such as extract, transform, and load (ETL) software, to load data into the data repository in an efficient and dependable manner.

There are several approaches to data loading, including:

- **Full load:** This entails simultaneously loading all of the data from the data sources into the data repository. This is usually done when the data repository is first set up or when the data in the data repository needs to be updated with a complete set of data from the source systems.
- **Incremental load:** This merely entails inserting new or changed data into the data repository. This is usually done when the data in the data repository is constantly being updated with fresh information from the source systems.

- **Batch load:** This entails importing data into the data repository via batch processes on a regular basis (e.g., nightly, weekly). This method is frequently utilized when a large number of data sources are being used to feed the data repository and the data loading process has to be divided into smaller, more manageable portions.

Overall, data loading is an essential component of every data management system. It entails obtaining data from diverse sources, purifying and formatting the data, and putting the data into the data repository in an efficient, dependable, and consistent manner with the overall data architecture.

5.6.Data Warehouse Design

The process of generating a systematic strategy for the creation and implementation of a data warehouse is referred to as data warehouse design. A well-designed data warehouse should be scalable, efficient, and adaptable in order to serve the business's querying and analytical needs.

There are several key considerations in data warehouse design, including:

- **Data sources:** The data sources that will be utilized to populate the data warehouse should be carefully identified and reviewed. This involves identifying the type of data (structured, unstructured, or semi-structured), the format of the data (e.g., CSV, JSON, SQL), and the frequency with which the data will be updated.
- **Data modeling:** The data model for the data warehouse should be carefully built to accommodate the business's querying and analytical needs. Choosing an acceptable data structure (e.g., star schema, snowflake schema), establishing the necessary dimensions and metrics, and specifying any necessary hierarchies or connections are all part of this process.

- ***Data transformation:*** Before data from diverse sources can be fed into the data warehouse, it may need to be converted, cleansed, and integrated. Typically, extract, transform, and load (ETL) technologies are used to extract data from source systems, transform the data into an appropriate format, and load it into the data warehouse.
- ***Data governance:*** To assure the integrity, security, and accessibility of the data in the data warehouse, a data governance architecture should be built. This involves defining data management roles and duties, creating data quality standards and processes, and implementing data security measures.
- ***Performance:*** The data warehouse's performance is important to the system's success. This covers the rate at which data can be searched and examined, as well as the system's general scalability and dependability.
- ***Data security:*** The security of the data in the data warehouse is important to the system's success. This involves putting in place safeguards against illegal access, data leakage, and data loss.
- ***Data visualization:*** The data warehouse should be built to facilitate the generation of clear, intuitive visuals that allow users to readily comprehend and evaluate the data. This includes selecting relevant visualization tools and design strategies to effectively explain the data insights.

Overall, data warehouse design entails a mix of technical and business factors to guarantee that the data warehouse is suited to the unique needs of the organization and can successfully support decision-making processes.

5.7.Data Warehouse Access

The ability of users to retrieve and query data from a data warehouse is referred to as data warehouse access. User authentication, authorisation, and access controls are frequently used to control data warehouse access.

There are several key aspects of data warehouse access, including:

- ***User authentication:*** This is the process of validating a user's identification before granting them access to the data warehouse. This can include utilizing credentials like a login and password, as well as more secure techniques like two-factor authentication.
- ***Authorization:*** This is the process of allowing users to access certain data and functions within the data warehouse. This may include rights to access, query, change, or remove data according on the user's job or responsibilities within the company.
- ***Access controls:*** These are the safeguards put in place to keep the data in the data warehouse safe from illegal access or misuse. Encryption, firewalls, and role-based access restrictions are examples of such measures.
- ***Data security:*** This is the process of putting safeguards in place to secure data in a data warehouse against unwanted access, data leakage, and data loss. Access restrictions, encryption, and other security measures may be used.
- ***Data governance:*** This is the process of developing a structure to secure the data warehouse's integrity, security, and accessibility. This involves defining data management roles and duties, creating data quality standards and processes, and implementing data security measures.

Overall, data warehouse access entails a mix of user authentication, authorization, access controls, data security, and data governance methods to guarantee that only authorized users have secure and regulated access to and query the data in the data warehouse.

5.8.Data Warehouse Best Practices

A data warehouse is a centralized collection of structured data used for quick data searching and analysis.

Here are some best practices for designing and maintaining a data warehouse:

- Define the data warehouse's purpose and aims clearly. This will aid in the design of the data warehouse schema as well as the selection of relevant tools and technology.
- For the data warehouse schema, utilize a dimensional model. A dimensional model is a data-organization scheme that divides data into facts (measurable occurrences) and dimensions (contextual data about the facts). This approach is simple to grasp and delivers quick query performance.
- In the data warehouse, employ a consistent naming system for tables, columns, and other objects. This will make it easier for others to interpret and use the data.
- Regularly load data into the data warehouse. ETL (Extract, Transform, Load) tools may be used to extract data from diverse sources, transform it into a consistent format, and load it into the data warehouse.
- To update the data warehouse, utilize incremental loading. Instead of refreshing the complete dataset, this method just loads new or updated data. This can save both time and money.
- Ensure that the data in the data warehouse is current and correct. This may be accomplished by data cleaning and quality assurance.
- Protect the data warehouse against unwanted access. This may be accomplished through the use of authentication, authorisation, and encryption.

- Monitor and optimize the data warehouse's performance. Indexing, segmentation, and columnar storage can all be used to increase query performance.
- Back up the data warehouse on a regular basis to avoid data loss. This can be accomplished via database snapshots or other backup methods.
- Document and keep the data warehouse up to date. Documenting the schema, data sources, ETL methods, and any other relevant information falls under this category.

By following these best practices, you can guarantee that your data warehouse is efficient, accurate, and safe, and that it acts as a useful resource for data analysis and decision-making.

Chapter 6

Data Mining & Business Intelligence

Data mining and business intelligence (BI) are two closely connected topics that are frequently used to evaluate and acquire insights from data. They do, however, have several fundamental peculiarities that must be understood.

Business intelligence is a collection of tools and strategies for analyzing and reporting on company data in order to help decision-making. Dashboards, reports, online analytical processing (OLAP), and data visualization are common BI tools used to acquire insights into an organization's performance. The data utilized in BI is often organized and comes from corporate systems such as ERP, CRM, and financial systems. Business analysts, managers, and executives often utilize BI technologies to aid in decision-making based on historical performance and present data.

In contrast, data mining is a set of techniques and statistical models used to uncover patterns and correlations in massive datasets. Data mining is the process of identifying trends and patterns that are not immediately obvious by employing machine learning, statistical analysis, pattern recognition, and clustering. Data mining may be used on both organized and unstructured data from a variety of sources, including databases, social media, and site logs. Data scientists, statisticians, and researchers generally utilize data mining methods to unearth insights that may be used to enhance decision-making, anticipate outcomes, or improve business operations.

The focus of BI and data mining differs significantly. BI is concerned with analyzing and reporting current data, whereas data mining is concerned with uncovering hidden patterns and trends in data. Data mining may be applied to both structured and unstructured data from a range of sources, whereas BI generally employs structured data from business systems. This implies that data mining is frequently used to detect patterns in data that are not immediately obvious, and it may assist firms in discovering new possibilities or areas for development that they were previously unaware of.

Another distinction is the instruments and procedures employed. BI tools are usually focused on reporting and visualization, and they are used to show data in a clear and intelligible manner. Data mining tools, on the other hand, are more focused on statistical analysis and machine learning, and they are used to identify hidden patterns and correlations in data.

Finally, there is a distinction in the types of people who use these two fields. Business analysts and managers often utilize BI tools, whereas data scientists and statisticians employ data mining tools. This implies that BI is frequently more focused on giving insights and assistance for corporate decision-making, whereas data mining is more focused on identifying deeper patterns and trends in data.

Overall, business intelligence and data mining are two essential and closely linked topics used to evaluate and acquire insights from data. While they differ in certain ways, they may also be utilized in tandem to give a more comprehensive understanding of the data and promote improved decision-making.

Table 3

Data Mining VS Business Intelligence

	Business Intelligence	Data Mining
Definition	A set of tools and techniques that are used to transform raw data into meaningful and useful information for business analysis and decision making.	A process of discovering patterns and relationships in large datasets to identify trends, classify data, and make predictions.
Goals	To provide a single version of the truth and a comprehensive view of the business. To support decision making by providing relevant and timely information to stakeholders.	To find hidden patterns, trends, and relationships in data that may not be immediately obvious. To make predictions and recommendations based on these insights.

Data Sources	Structured data stored in databases and data warehouses.	Structured and unstructured data from various sources, including databases, social media, and sensors.
Role in Data Analysis	Presentation of results	Identification of patterns and trends
Techniques Used	OLAP, reporting, dashboards, scorecards, visualization, etc.	Machine learning, statistical analysis, pattern recognition, predictive modeling, etc.
Examples of BI Tools	Tableau, Power BI, QlikView, MicroStrategy, etc.	RapidMiner, Weka, Orange, KNIME, etc.
Output	Meaningful and useful information for business analysis and decision making.	Insights and predictions based on patterns and trends in the data.
Business Function Supported	All functions, including sales, marketing, finance, operations, HR, etc.	Any function where data-driven insights can provide value, such as customer segmentation, fraud detection, risk assessment, etc.

6.1.Data Mining In Business Analytics

Data mining techniques are a collection of algorithms and statistical approaches for discovering patterns and correlations in massive datasets. These strategies are often used in business analytics to unearth insights that may help firms improve their performance and guide decision making.

The following are some examples of data mining techniques that are regularly used in business analytics:

- Clustering
- Association Rule Learning
- Decision Trees
- Neural Networks

- Support Vector Machines
- Regression Analysis

In addition to these techniques, numerous additional data mining methods, such as time series analysis, survival analysis, and text mining, can be employed in business analytics. By applying these approaches to massive datasets, companies may acquire useful insights that can help them make better decisions and enhance their operations.

6.2.The Relationship Between DM & BI

Business intelligence (BI) and data mining (DM) are two different but related concepts that are frequently used interchangeably in the context of data analysis and decision making. Here's a more in-depth description of how the two interact:

Business intelligence is a collection of tools and processes for transforming raw data into relevant and usable information for company analysis and decision making. Reporting, dashboards, scorecards, visualization, and online analytical processing (OLAP) are examples of BI technologies. The purpose of BI is to give a single version of the truth and a comprehensive perspective of the company, as well as to help decision making by presenting stakeholders with relevant and timely information.

In contrast, data mining is the act of uncovering patterns and correlations in massive datasets in order to identify trends, categorize data, and make predictions. Machine learning, statistical analysis, pattern recognition, and predictive modeling are examples of data mining approaches. Data mining may be used to analyze organized and unstructured data from a variety of sources, such as databases, social media, and sensors. The purpose of data mining is to discover hidden patterns, trends, and correlations in data that are not immediately visible, and to generate predictions and recommendations based on these discoveries.

In practice, BI and DM frequently collaborate to evaluate data and inform decision-making. Data mining techniques are used to uncover patterns and trends in data, and business intelligence tools are used to convey the results of these studies in a clear and understandable manner. A store, for example, may employ data mining to uncover patterns in client behavior, such as which goods are usually

purchased together. They may then utilize BI tools to develop a dashboard that shows this information in a clear and simple manner, such as a bar chart displaying the most popular product combinations. This data may then be utilized to inform marketing campaigns or to enhance product positioning in stores. There are many different ways in which BI and DM can work together, depending on the specific needs of the business and the data that is available.

Some common examples include:

- ***Customer segmentation:*** Customers can be divided into segments using DM approaches based on their traits and behavior, such as purchase history, demographics, and preferences. Then, using BI tools, dashboards and reports can be created that highlight the characteristics of each segment, such as average order value, lifetime value, and churn rate. This data may be utilized to better target marketing efforts and increase client retention.
- ***Sales forecasting:*** Based on historical data such as past sales, marketing efforts, and economic factors, DM approaches may be used to forecast future sales. BI technologies may then be used to portray these forecasts in an easy-to-understand format, such as a line chart indicating expected sales for the following quarter. This data may be utilized to prepare for future growth and make smart resource allocation decisions.

6.3. The Cloud & The Future of DM for BI

The cloud has had a major impact on the field of data mining for business intelligence, and it is likely to continue shaping the future of this area in significant ways.

The capacity to access enormous volumes of data from a variety of sources is one of the most essential benefits of using the cloud for data mining. Businesses used to have to rely on data that was housed on their own servers, which confined them to data that they had acquired themselves. Businesses, on the other hand, may use the cloud to access a large network of data sources, including social media,

sensors, and other connected devices, as well as traditional data sources like transactional and operational data. This enables companies to obtain a far more thorough understanding of their consumers, markets, and operations.

Another advantage of the cloud is that it makes it easier for organizations to expand their data mining activities. In the past, organizations were frequently required to invest in costly technology and software in order to analyze enormous volumes of data. Businesses, on the other hand, may utilize the cloud to acquire the processing power and storage they require on an as-needed basis, which means they only pay for what they use. This allows organizations to scale up their data mining activities as needed without needing to make large upfront costs.

The cloud also facilitates company collaboration on data mining initiatives. Due to technology and software restrictions, it was sometimes difficult for teams to share data and cooperate on projects in the past. However, thanks to the cloud, teams can effortlessly exchange data and communicate in real time, no matter where they are. This has made it more easier for firms to form cross-functional teams and collaborate on data mining initiatives.

The use of machine learning for data mining is one area where the cloud is anticipated to have a particularly major influence in the future. Large volumes of data are required for machine learning algorithms to be effective, and the cloud makes it much easier for organizations to access and analyse this data. As a result, we may expect to see an increase in the number of organizations employing machine learning for data mining in the future years.

Another area where the cloud is anticipated to have a future influence is the development of new data mining techniques and technologies. As organizations continue to gather and retain more data, new tools and technology to assist them make sense of this data will be required. Because the cloud makes it much simpler for businesses to access and experiment with these new tools, we are likely to witness the development of a lot of innovative new data mining technologies in the coming years.

Overall, the cloud is expected to continue to play a significant role in data mining for business intelligence in the future. It has already had a huge impact, and as organizations gather and retain more data, the demand for effective data mining

tools and technologies will only grow. As a result, we may anticipate a number of fascinating new discoveries in this field during the next few years.

6.4.The Role Data Mining Plays for BI

Data mining is an important part of the business intelligence (BI) process since it requires evaluating massive volumes of data to identify patterns, trends, and correlations that are not always obvious. Businesses may then utilize these insights to make more informed decisions, enhance operations, and gain a competitive edge.

Data mining employs a wide range of techniques, including machine learning, statistical analysis, and visualization, to extract useful insights from data. Data mining, for example, may be used to uncover trends in consumer behavior, such as which items are the most popular or which marketing efforts are the most effective. This data may then be utilized to adapt marketing campaigns, improve inventory management, and enhance the entire consumer experience.

Data mining also plays a significant part in BI by predicting future trends and patterns. Businesses may generate accurate forecasts about future events or trends by examining past data, which can help them remain ahead of the competition and make strategic decisions. For example, a store may utilize data mining to forecast which products will be popular in the next months, allowing them to stock up and perhaps improve sales.

Data mining may also be used to increase corporate process efficiency by detecting bottlenecks or inefficiencies. A manufacturing business, for example, may use data mining to examine production data and identify bottlenecks in the manufacturing process, which may subsequently be addressed to increase efficiency. Data mining may also be used to optimize supply chain management by detecting demand patterns and possibilities to decrease waste and enhance efficiency.

In addition to these applications, data mining is important in risk management and fraud detection. Businesses may detect possible hazards and take actions to reduce them before they become a problem by examining data trends. A financial

institution, for example, may employ data mining to uncover patterns of fraudulent behaviour, allowing them to avoid fraud and safeguard their consumers.

Overall, data mining is an important BI technique since it enables firms to make better decisions, streamline operations, and gain a competitive edge. Businesses may unearth hidden insights and utilize this knowledge to make strategic decisions that promote development and success by analyzing enormous volumes of data.

6.5.The Benefits of Data Mining in BI

Data mining is an important aspect of the business intelligence (BI) process because it requires sifting through vast volumes of data to identify patterns, trends, and correlations that may not be obvious at first. Businesses may then utilize this data to make educated decisions, enhance operations, and gain a competitive edge.

One of the primary advantages of data mining in business intelligence is the better decision-making it provides. Businesses may make better educated decisions based on real facts rather than gut inclination by finding patterns and trends in data. This can result in more accurate forecasting, better marketing effort targeting, and more efficient resource allocation.

Another advantage of data mining in business intelligence is the greater efficiency it may provide. Businesses may optimize their operations and boost efficiency by evaluating data and detecting bottlenecks or inefficiencies in procedures. This can result in cost savings and higher productivity, which can fuel growth and success.

In addition to these advantages, data mining may assist organizations in improving the customer experience. Businesses may adjust their products and services to better fit the requirements and preferences of their consumers by evaluating trends in customer behavior. This can result in higher client happiness and loyalty, both of which are essential for long-term success.

Data mining may also be useful in risk management. Businesses may detect possible hazards and take actions to reduce them before they become a problem by examining data trends. This can assist organizations in avoiding costly mistakes and safeguarding their assets.

Finally, data mining may boost competitiveness by giving firms a competitive advantage. Businesses may acquire a competitive advantage over their competitors by evaluating data and making educated decisions. This can assist firms in standing out in a crowded market and achieving success.

Overall, the advantages of data mining in business intelligence are vast and far-reaching. Businesses may unearth hidden insights and utilize this knowledge to make strategic decisions that promote development and success by analyzing enormous volumes of data.

Table 4

The Benefits of DM in BI

Benefit	Description
Improved Decision Making	Data mining helps businesses to make more informed decisions by uncovering patterns, trends, and relationships in data that may not be immediately visible. This can lead to more accurate forecasting, better targeting of marketing efforts, and more effective resource allocation.
Increased Efficiency	Data mining can help businesses to identify bottlenecks and inefficiencies in processes, allowing them to optimize and improve efficiency. This can lead to cost savings and increased productivity, which can ultimately drive growth and success.
Enhanced Customer Experience	By analyzing patterns in customer behavior, businesses can tailor their products and services to better meet the needs and preferences of their customers. This can lead to increased customer satisfaction and loyalty, which are critical to long-term success.
Improved Risk Management	By analyzing patterns in data, businesses can identify potential risks and take steps to mitigate them before they become a problem. This can help businesses to avoid costly mistakes and protect their assets.

Enhanced Competitiveness	By analyzing data and making informed decisions, businesses can gain a competitive advantage over their rivals. This can help businesses to stand out in a crowded market and drive success.
Improved Fraud Detection	Data mining can be used to identify patterns of fraudulent activity, allowing businesses to take steps to prevent fraud and protect their customers. This can help businesses to safeguard their assets and reputation.
Optimized Resource Allocation	By analyzing data, businesses can identify which resources are being used most effectively and where improvements can be made. This can help businesses to allocate resources more efficiently and improve overall performance.
Enhanced Predictive Analytics	By analyzing historical data, businesses can make informed predictions about future events or trends. This can help businesses to stay ahead of the competition and make strategic decisions.
Improved Process Optimization	Data mining can be used to identify inefficiencies in business processes and optimize them for maximum efficiency. This can lead to cost savings and increased productivity.
Enhanced Market Research	By analyzing patterns in data, businesses can gain a deeper understanding of their target market and identify opportunities for growth. This can help businesses to make more informed marketing decisions and drive success.

6.6.The Challenges of Data Mining in BI

Despite the obvious advantages of data mining in business intelligence (BI), there are a number of hurdles to overcome. These obstacles might vary from technical to organizational and cultural.

The sheer volume of data that must be evaluated is one of the key technological hurdles of data mining in BI. With the explosion of digital data, organizations are frequently confronted with big, complicated datasets that can be difficult to manage and evaluate. This may necessitate the use of specialized software and technology, as well as a team of qualified analysts to interpret the data.

Another problem of data mining in business intelligence is data quality. Data mining can only be effective if the data is accurate, full, and relevant. However, data quality is frequently a concern, since data may be recorded incorrectly or lacking essential elements. This can make gaining useful insights from data challenging.

There are organizational and cultural impediments to data mining in BI, in addition to technological problems. For example, firms may lack the essential infrastructure to support data mining, or top management may lack awareness or buy-in. Concerns about data privacy and security may also exist, which may limit the sorts of data that may be evaluated.

Finally, keeping up with the quick speed of development in the field of data mining is a difficulty. With new technology and approaches continually emerging, organizations must keep current in order to remain competitive. This may necessitate continuing training and financial investment in the relevant resources.

Overall, while data mining may provide several benefits to business intelligence, there are a number of issues that must be addressed. These difficulties can vary from technical concerns to organizational and cultural impediments, and overcoming them necessitates a mix of technical competence, effective leadership, and a supportive culture.

Table 5

The Challenges of DM in BI

Challenge	Description
Large, Complex Datasets	With the proliferation of digital data, businesses are often faced with large, complex datasets that can be difficult to process and analyze. This can require specialized software and hardware, as well as a team of skilled analysts to make sense of the data.
Data Quality Issues	In order for data mining to be effective, the data must be accurate, complete, and relevant. However, data quality can often be an issue, as data may be entered inaccurately or may

	be missing key fields. This can make it difficult to draw meaningful insights from the data.
Organizational and Cultural Barriers	Businesses may lack the necessary infrastructure to support data mining, or there may be a lack of understanding or buy-in from senior management. There may also be concerns around data privacy and security, which can limit the types of data that can be analyzed.
Keeping Up With Change	With new technologies and techniques emerging constantly, businesses need to stay up-to-date in order to remain competitive. This can require ongoing training and investment in the necessary resources.
Ethical and Legal Considerations	Data mining can raise ethical and legal issues, such as concerns around data privacy and the use of sensitive personal information. Businesses need to be aware of these issues and take steps to address them in order to avoid potential problems.
Integration With Existing Systems	Data mining often involves integrating data from multiple sources, which can be challenging if these systems are not compatible. This can require significant effort to ensure that data is properly formatted and can be easily analyzed.
Limited Data Availability	In some cases, businesses may not have access to all of the data they need in order to make informed decisions. This can be due to a variety of factors, such as data being siloed or unavailable.
Insufficient Resources	Data mining can be resource-intensive, requiring specialized software and hardware as well as a team of skilled analysts. Businesses may not have the necessary resources to support data mining, which can limit their ability to extract insights from their data.

6.7.The Industries That Benefit From DM in BI

There are many industries that can benefit greatly from the use of data mining in business intelligence (BI). Some of the industries that are most likely to see significant benefits from data mining include:

- Retails
- Health Care
- Financial Services
- Manufacturing
- Telecommunications
- Government

6.7.1. Retails: One industry that can tremendously benefit from data mining in business intelligence (BI) is retail. Retailers may obtain a better knowledge of their consumers and modify their products and services to better fit their requirements by evaluating data on their purchases, preferences, and behavior.

The capacity to optimize inventory management is one of the primary advantages of data mining in retail. Retailers may detect which goods are the most popular and guarantee that they have enough stock to fulfill demand by monitoring sales and consumer behavior data. This can assist merchants in avoiding out-of-stock situations and reducing waste, resulting in cost savings.

Data mining may also assist merchants in identifying customer behavior trends and tailoring their marketing efforts accordingly. Retailers may design more successful targeted marketing efforts by evaluating data on client purchases and preferences. This can assist shops in increasing sales and driving growth.

Aside from these advantages, data mining may assist merchants in improving the entire consumer experience. Retailers may discover areas for improvement and take efforts to solve them by evaluating data from client interactions. Retailers, for example, may use data mining to detect

bottlenecks in the checkout process and take steps to decrease consumer wait times.

Overall, data mining may be a valuable tool for retailers, assisting them in optimizing inventory management, improving marketing efforts, and improving the consumer experience. Retailers may unearth hidden insights and use this knowledge to promote development and success by analyzing enormous volumes of data.

6.7.2. Health Care: The application of data mining in business intelligence (BI) may considerably assist the healthcare industry. Healthcare companies can find patterns and trends in enormous volumes of data, which may help them enhance patient care and optimize resource allocation.

The capacity to find chances for prevention and early intervention is one of the primary benefits of data mining in healthcare. Healthcare practitioners can find trends in patient medical history data that may suggest a risk of certain disorders or diseases. This enables them to take proactive measures to minimize or lessen these risks, resulting in improved patient outcomes.

In the healthcare industry, data mining may also be utilized to optimize resource allocation. Healthcare organizations can discover areas where resources are being over- or underutilized by examining patient demand data and taking efforts to optimize their utilization. This can assist healthcare companies in lowering expenses and increasing efficiency.

Aside from these advantages, data mining may assist healthcare companies in improving the quality of treatment they give. Healthcare professionals can discover areas for improvement and take efforts to address them by examining data on patient outcomes. They may, for example, employ data mining to discover trends in patient satisfaction and take efforts to improve the patient experience.

Overall, data mining has the potential to be a significant tool in the healthcare sector, assisting organizations in identifying possibilities for prevention and early intervention, optimizing resource allocation, and improving the quality of treatment they give. Healthcare firms may unearth hidden insights and utilize this knowledge to promote change and success by analyzing enormous volumes of data.

6.7.3. Financial Services: Data mining may be a highly beneficial tool for financial services organizations wanting to gather insights and enhance their business. Financial services organizations can identify patterns and links that are not immediately obvious by analyzing enormous volumes of data. This can help businesses make more educated business decisions, such as discovering new markets to enter, optimizing their goods and services, and enhancing their risk management methods.

Predictive analytics is one specific method in which financial services organizations might profit from data mining. Companies may construct models that anticipate future events with high accuracy by examining historical data. This is especially beneficial for risk management since it helps businesses to detect possible concerns and take actions to minimize them before they arise.

Data mining, in addition to predictive analytics, may assist financial services organizations with consumer segmentation. Companies can discover particular groups of consumers with similar characteristics and modify their goods and services to better satisfy the demands of these groups by analyzing customer data. This may assist businesses in increasing customer satisfaction and loyalty while also driving sales and income.

Overall, data mining may be a significant tool for financial services firms seeking a competitive edge and improving their operations. Companies can make smarter decisions, enhance their goods and services,

and manage risk by analyzing data and revealing hidden patterns and linkages.

- 6.7.4. Manufacturing: Data mining may be a highly beneficial tool for manufacturing organizations wanting to acquire insights and enhance their business. Manufacturing companies can identify patterns and correlations that were previously unknown by analyzing massive amounts of data. This may help businesses make more educated business decisions, such as finding areas for improvement in their production processes, optimizing their supply chain, and lowering expenses.

Predictive maintenance is one specific method in which manufacturing organizations might profit from data mining. Companies can detect when maintenance is likely to be required and arrange it in advance by analyzing data from machinery and equipment, rather than waiting for equipment to break. This can assist decrease downtime and boost factory efficiency.

Data mining may assist manufacturing organizations with quality control in addition to predictive maintenance. Companies can identify places where faults are more likely to occur and take efforts to prevent them by evaluating data from manufacturing processes. This can assist minimize the quantity of damaged items while also increasing consumer happiness.

Overall, data mining may be a valuable tool for manufacturers wanting to optimize their processes and increase efficiency. Companies may find areas for improvement, optimize operations, and save costs by analyzing data and uncovering hidden patterns and linkages.

- 6.7.5. Telecommunications: Data mining may be a highly beneficial tool for telecoms firms wanting to acquire insights and enhance their company. Telecommunications companies can identify patterns and links that are not immediately obvious by analyzing vast amounts of data. This can

help them make more educated business decisions, such as recognizing new market possibilities, optimizing their goods and services, and increasing customer service.

Customer segmentation is one specific method that telecoms firms might profit from data mining. Companies can discover particular groups of consumers with similar characteristics and modify their goods and services to better satisfy the demands of these groups by analyzing customer data. This may assist businesses in increasing customer satisfaction and loyalty while also driving sales and income.

Data mining may assist telecoms businesses with network optimization in addition to consumer segmentation. Companies can find bottlenecks and other areas for improvement in their networks by evaluating data from their networks and taking efforts to fix them. This can assist to improve the network's overall performance and dependability.

Overall, data mining may be an effective tool for telecoms firms seeking a competitive edge and improving their operations. Companies can make more informed decisions, enhance their goods and services, and better serve their consumers by analyzing data and identifying hidden patterns and linkages.

6.7.6. Government: Data mining may be a highly beneficial tool for governments wanting to acquire insights and enhance their operations. Governments can identify patterns and linkages that were previously unknown by examining enormous amounts of data. This can assist them in making more informed decisions regarding their policies and activities, such as finding areas for improvement, maximizing their resources, and better fulfilling the needs of their residents.

Predictive analytics is one specific manner in which governments might profit from data mining. Governments may construct models that anticipate future events with great accuracy by evaluating historical data.

This is especially beneficial for policymaking since it helps governments to detect prospective difficulties and take efforts to remedy them before they materialize.

Data mining, in addition to predictive analytics, may assist governments with resource allocation. Governments can better identify where their resources are most required and allocate them accordingly by examining data on the requirements of different areas or populations. This can assist guarantee that government programs and services reach the people who need them the most.

Overall, data mining may be an effective tool for governments seeking to enhance their operations and better serve their residents' demands. Governments can make better decisions, optimize resources, and discover areas for development by analyzing data and uncovering hidden patterns and linkages.

Table 6

The Industries That Benefit from DM & BI

Industry	How DM is Used	How BI is Used
HealthCare	Analyzing patient data to identify trends and patterns in disease outbreaks, inform treatment decisions, and improve patient outcomes	Presenting data on patient outcomes, treatment effectiveness, and resource utilization to inform healthcare policies and decision making
Finance	Analyzing market trends, identifying fraudulent activity, and optimizing investment portfolios	Presenting data on financial performance, risk, and compliance to inform business strategies and decision making
Retail	Analyzing customer data and sales data to improve marketing efforts and identify new business opportunities	Presenting data on sales, customer demographics, and market trends to inform business strategies and decision making

Manufacturing	Optimizing production processes and reducing costs	Presenting data on production, quality, and efficiency to inform operational strategies and decision making
Government	Analyzing large datasets to identify trends, patterns, and relationships that can help inform public policy and decision making.	Presenting data on population, demographics, and economic indicators to inform public policy and decision making.
Telecommunications	Analyzing customer data and usage data to improve marketing efforts and identify new business opportunities.	Presenting data on customer demographics, usage patterns, and market trends to inform business strategies and decision making.
Energy	Analyzing data on energy production, consumption, and distribution to optimize operations and reduce costs.	Presenting data on energy production, consumption, and demand to inform business strategies and decision making.
Transportation	Analyzing data on transportation routes, schedules, and demand to optimize operations and improve efficiency.	Presenting data on transportation routes, schedules, and demand to inform business strategies and decision making.
Media	Analyzing data on audience demographics, preferences, and engagement to inform content creation and distribution strategies.	Presenting data on audience demographics, preferences, and engagement to inform content creation and distribution strategies.
Education	Analyzing student data and performance data to inform teaching strategies and improve student outcomes.	Presenting data on student performance, demographics, and trends to inform educational policies and decision making.

6.8.Key Differences Between DM & BI

Data mining (DM) and business intelligence (BI) are frequently used interchangeably, however they are not the same thing. Advanced statistical and

machine learning techniques are used in DM to identify patterns and trends in large datasets, whereas BI uses data visualization and reporting tools to present insights in an easily understandable format.

One significant distinction between DM and BI is the focus of the analysis. DM focuses on uncovering hidden patterns and trends in massive datasets, whereas BI focuses on presenting data in a readily understood style to aid decision making. DM frequently includes the use of complicated algorithms and statistical models to evaluate data, whereas BI involves the use of tools like as dashboards, charts, and reports to graphically show data.

Another distinction between DM and BI is the sort of data that is analyzed. DM is typically used to analyze structured data, such as data from databases or spreadsheets, whereas BI is frequently used to analyze both structured and unstructured data, such as data from social media, web logs, and text documents.

Overall, the primary distinctions between DM and BI are the emphasis of the analysis, the type of data analyzed, and the tools and techniques utilized to analyze and display data. Both DM and BI are crucial tools for businesses wanting to extract insights from data and guide decision making, but they serve distinct objectives and are utilized in various ways.

6.9.Summary

Data mining (DM) and business intelligence (BI) are two closely linked topics that are used to extract insights from data and inform decision making. Advanced statistical and machine learning techniques are used in DM to find patterns and trends in massive datasets, whereas BI uses data visualization and reporting tools to deliver insights in an easily accessible style.

The emphasis of the analysis is one significant distinction between DM and BI. DM is concerned with uncovering hidden patterns and trends in massive datasets, whereas BI is concerned with presenting data in an intelligible style to aid decision making. DM frequently includes the use of complicated algorithms and statistical models to evaluate data, whereas BI involves the use of tools like as dashboards, charts, and reports to graphically show data.

Another distinction between DM and BI is the sort of data analyzed. DM is often used to analyze structured data from databases or spreadsheets, but BI is frequently used to evaluate both structured and unstructured data from social media, site logs, and text documents.

Both data mining and business intelligence (BI) are important tools for organizations looking to extract insights from data and inform decision making, but they serve different purposes and are used in different ways. Data scientists, statisticians, and analysts often use DM, whereas business analysts, managers, and executives typically use BI. Both DM and BI require strong analytical skills and the ability to effectively communicate insights to stakeholders.

Chapter 7

Experimental Studies

The popularity of songs on music streaming services like Spotify is a significant factor that influences user engagement and satisfaction. To leverage the power of data science and machine learning, you aim to develop a predictive model that can accurately determine the popularity of songs based on various current and historical features. By applying machine learning algorithms and evaluating their performance using success metrics or criteria, you seek to identify the best algorithm that can effectively predict the popularity of songs.

The problematic can be further detailed as follows:

- **Understanding Song Popularity:** The popularity of a song can be influenced by various factors such as its genre, artist, release date, tempo, energy, danceability, and more. However, determining the exact relationship between these features and the popularity of a song is a complex task. The problem involves exploring and analyzing these features to understand their significance in predicting popularity accurately.
- **Feature Selection and Engineering:** To develop an effective predictive model, it is crucial to identify the most relevant features that contribute to the popularity of songs. This process may involve feature selection techniques to choose the subset of features that have the highest predictive power. Additionally, feature engineering might be required to create new features or transform existing ones to improve the model's performance. Analyzing the link between variables, assessing the most linearly related attributes to popularity, and showing the mean popularity across distinct qualities are all part of the EDA. (Year/ Energy/ Loudness/ Danceability/ Tempo/ Acousticness)

- **Machine Learning Algorithms:** To predict song popularity, you plan to explore and evaluate multiple machine learning algorithms. This involves applying algorithms such as Logistic Regression, K-Nearest Neighbors, Decision Trees, Support Vector Machines, Random Forests, and AdaBoost. The goal is to compare their performance using appropriate success metrics and determine the algorithm that provides the most accurate predictions.
- **Model Evaluation and Selection:** The success metrics or criteria used for evaluating the models could include accuracy score, confusion matrix, classification report, and cross-validation scores. These metrics help assess the models' performance on different popularity levels, identify any biases or limitations, and evaluate their stability and generalization ability. Based on these evaluations, the best-performing model will be selected as the final predictive model for song popularity.
- **Application and Scalability:** Once the best model is developed, the aim is to deploy it within the product or application that allows users to predict the popularity of songs in real-time. The scalability of the model should be considered to ensure it can handle large amounts of data and provide predictions efficiently and accurately.

By addressing these challenges and developing an accurate predictive model for song popularity, you aim to enhance user experiences on music streaming platforms by providing insights into the potential popularity of songs. This solution can have broader applicability beyond your specific product and can be applied to other music streaming services as well.

7.1.Dataset

	acousticness	artists	danceability	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popularity	release_date	speechiness	tempo	valence	year	duration_ms
0	0.995	Carl Woelchsch	0.708	0.1950	0	6KQ3uYMLK25DnLF7wYDD	0.563	10	0.1510	-12.428	1	Singende Battalione 1. Teil	0	1928	0.0506	118.469	0.7790	1928	2.644133
1	0.994	Robert Schumann, Vladimir Horowitz	0.379	0.0135	0	6KuQTu1KdTTkLXKwLpV	0.901	8	0.0753	-26.454	1	Fantasietücke, Op. 111. Più tosto lento	0	1928	0.0462	83.972	0.0757	1928	4.702217
2	0.604	Seweryn Gieszczyński	0.749	0.2200	0	6L63VW0PbdM1HDS8ogndM	0.000	5	0.1190	-19.924	0	Chapter 1.18 - Zamek kanowski	0	1928	0.9290	107.177	0.8800	1928	1.738333
3	0.995	Francisco Canaro	0.781	0.1300	0	6M94FKK15sOACQYRnWPN8	0.887	1	0.1110	-14.734	0	Belamos Juritos - Instrumental (Remasterizado)	0	1928-09-25	0.0926	108.003	0.7200	1928	3.012667
4	0.990	Frédéric Chopin, Vladimir Horowitz	0.210	0.2040	0	6N68FZwLTSOnxj8qkd	0.908	11	0.0980	-15.829	1	Polonaise-Fantaisie in A-Flat Major, Op. 61	1	1928	0.0424	62.149	0.0693	1928	11.452217

Figure 7. Dataset

The "data.csv" file contains almost 160.000 songs gathered via the Spotify Web API. The information comes from Spotify and comprises 169k songs from 1921 to 2020. Every year, the top 100 songs are released.

- The primary attribute is the "id," which is a unique identifier generated by Spotify for each track.
- The numerical attributes include:
 - "acousticness," which represents the level of acoustic quality ranging from 0 to 1.
 - "danceability," indicating how suitable a track is for dancing, ranging from 0 to 1.
 - "energy," representing the energy level of a track, ranging from 0 to 1.
 - "duration_ms," an integer value typically ranging from 200,000 to 300,000, indicating the duration of the track in milliseconds.
 - "instrumentalness," indicating the degree of instrumental content in a track, ranging from 0 to 1.
 - "valence," representing the musical positivity conveyed by a track, ranging from 0 to 1.
 - "popularity," ranging from 0 to 100, indicating the popularity of a song.

- "tempo," a float value typically ranging from 50 to 150, representing the tempo or speed of a track.
 - "liveness," indicating the presence of a live audience in a track, ranging from 0 to 1.
 - "loudness," a float value typically ranging from -60 to 0, representing the loudness of a track.
 - "speechiness," ranging from 0 to 1, representing the presence of spoken words in a track.
 - "year," ranging from 1921 to 2020, indicating the year of release.
- The dummy attributes are:
 - "mode," encoded as 0 for minor key and 1 for major key.
 - "explicit," encoded as 0 for no explicit content and 1 for explicit content.
 - The categorical attributes include:
 - "key," representing the musical key of the track encoded as values ranging from 0 to 11, starting from C as 0 and progressing chromatically.
 - "artists," which is a list of artists associated with the track.
 - "release_date," representing the date of release, mostly in the yyyy-mm-dd format.
 - "name," which is the name of the song.
- ⇒ This dataset provides a comprehensive collection of features that can be used for analyzing and predicting song popularity using data science and machine learning techniques.
- ⇒ The link where I have found my dataset:
- <https://www.kaggle.com/datasets/ektanegi/spotifydata-19212020/code>

7.2.Code

7.2.1. Import Libraries: This code imports several libraries that are commonly used in machine learning tasks:

- Pandas and numpy are used for data manipulation and analysis.
- seaborn and matplotlib are used for data visualization.
- train_test_split is used for splitting data into training and validation sets.
- curve_fit, FunctionTransformer, OneHotEncoder, MinMaxScaler, and ColumnTransformer are used for data preprocessing and transformation.
- LogisticRegression, KNeighborsClassifier, DecisionTreeClassifier, SVC, LinearRegression, RandomForestClassifier, and AdaBoostClassifier are all different models that can be used for machine learning tasks.
- mean_squared_error, accuracy_score, confusion_matrix, cross_val_score, and classification_report are all used for evaluating model performance and making predictions.

```
[3] # Import the required libraries for machine learning
import pandas as pd # data manipulation and analysis
import numpy as np # numerical operations
import seaborn as sns # data visualization
import matplotlib.pyplot as plt # data visualization

[4] # For splitting data into training and validation sets
from sklearn.model_selection import train_test_split

[5] # For data preprocessing and transformation
from sklearn.preprocessing import FunctionTransformer # transform data using user-defined function
from sklearn.preprocessing import OneHotEncoder # convert categorical features to numeric features
from sklearn.preprocessing import MinMaxScaler # scale features to a specified range
from sklearn.compose import ColumnTransformer # apply different transformations to different columns

[6] # For models
from sklearn.linear_model import LogisticRegression # binary classification model
from sklearn.linear_model import LinearRegression # regression model
from sklearn.neighbors import KNeighborsClassifier # k-nearest neighbors classification model
from sklearn.tree import DecisionTreeClassifier # decision tree classification model
from sklearn.svm import SVC # support vector machine classification model
from sklearn.ensemble import RandomForestClassifier # random forest classification model
from sklearn.ensemble import AdaBoostClassifier # AdaBoost classification model

[7] # For scoring and evaluation
from sklearn.metrics import mean_squared_error # mean squared error metric for regression
from sklearn.metrics import accuracy_score # accuracy metric for classification
from sklearn.metrics import confusion_matrix # confusion matrix for classification
from sklearn.metrics import classification_report # classification report for classification
from sklearn.model_selection import cross_val_score # cross validation for model evaluation
```

Figure 8. Screenshot of importing libraries

7.2.2. Data Visualization

```
# Import pandas library for data manipulation and analysis
import pandas as pd

# Load the dataset from a CSV file and store it in a pandas dataframe object
df = pd.read_csv('data.csv')

# Drop unnecessary columns from the dataframe
# The axis=1 argument specifies that we want to drop columns (as opposed to rows)
# The columns to drop are specified as a list of column names
# The resulting dataframe is stored back in the original variable
df = df.drop(["explicit", "id", "mode", "name", "release_date"], axis=1)

# Remove duplicate rows from the dataframe
# The duplicated() method returns a boolean array indicating whether each row is a duplicate of a previous row
# The ~ operator negates this array, so that True becomes False and vice versa
# The resulting boolean array is used to select all rows that are not duplicates using boolean indexing
# The resulting dataframe with no duplicates is stored back in the original variable
df = df[~df.duplicated()]
```

Figure 9. Screenshot of reading, dropping & removing.

⇒ This code reads in a dataset from a CSV file and stores it as a pandas dataframe. It then drops several columns that are not needed for the analysis. Finally, it removes any duplicate rows from the dataframe.

```
[10] #inspect the first few rows of a dataset
df.head()
```

	acousticness	artists	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	popularity	speechiness	tempo	valence	year
0	0.995	['Carl Woitschach']	0.708	158648	0.1950	0.563	10	0.1510	-12.428	0	0.0506	118.469	0.7790	15
1	0.994	['Robert Schumann', 'Vladimir Horowitz']	0.379	282133	0.0135	0.901	8	0.0763	-28.454	0	0.0462	83.972	0.0767	15
2	0.604	['Seweryn Goszczyński']	0.749	104300	0.2200	0.000	5	0.1190	-19.924	0	0.9290	107.177	0.8800	15
3	0.995	['Francisco Canaro']	0.781	180760	0.1300	0.887	1	0.1110	-14.734	0	0.0926	108.003	0.7200	15
4	0.990	['Frédéric Chopin', 'Vladimir Horowitz']	0.210	687733	0.2040	0.908	11	0.0980	-16.829	1	0.0424	62.149	0.0693	15

Figure 10. Screenshot of inspecting the first few rows of the dataset

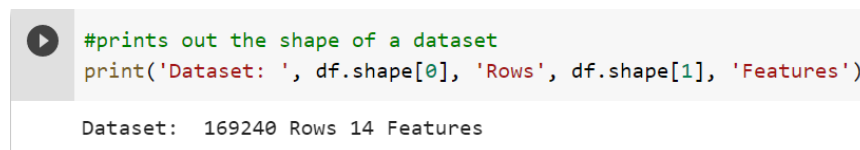
⇒ The **dataset.head()** method is a useful tool in data analysis that allows you to quickly inspect the first few rows of a dataset. By default, it displays the first 5 rows of the dataset in a tabular format, with the column names listed at the top.



	acousticness	artists	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	popularity	speechiness	tempo	valence
169904	0.1730	['DripReport', 'Tyga']	0.875	163800	0.443	0.000032	1	0.0891	-7.461	75	0.1430	100.012	0.30
169905	0.0167	['Leon Bridges', 'Terrace Martin']	0.719	167468	0.385	0.031300	8	0.1110	-10.907	64	0.0403	128.000	0.27
169906	0.5380	['Kyygo', 'Oh Wonder']	0.514	180700	0.539	0.002330	7	0.1080	-9.332	70	0.1050	123.700	0.15
169907	0.0714	['Cash', 'Andy Grammer']	0.646	167308	0.761	0.000000	1	0.2220	-2.557	70	0.0385	129.916	0.47
169908	0.1090	['Ingrid Andress']	0.512	214787	0.428	0.000000	0	0.1050	-7.387	65	0.0271	80.588	0.36

Figure 11. Screenshot of inspecting the last few rows of the dataset

⇒ The **dataset.tail()** method is a useful tool in data analysis that allows you to quickly inspect the last few rows of a dataset. By default, it displays the last 5 rows of the dataset in a tabular format, with the column names listed at the top.



```
#prints out the shape of a dataset
print('Dataset: ', df.shape[0], 'Rows', df.shape[1], 'Features')
```

Dataset: 169240 Rows 14 Features

Figure 12. Screenshot of the dataset shape

⇒ This is a Python code snippet that prints out the shape of a dataset using the pandas library.

The **dataset.shape[0]** returns the number of rows in the dataset, while **dataset.shape[1]** returns the number of columns or features in the dataset.

So the printed statement "**Dataset: <number of rows> Rows, <number of columns> Features**" indicates the size of the dataset in terms of the number of rows and columns or features it contains.

```

# Make a copy of the original DataFrame
classified = df.copy()

# Add a new column called 'pop_rating'
classified['pop_rating'] = ''

# Loop through each row of the DataFrame
for i, row in classified.iterrows():

    # Set the initial rating score to 'unpopular'
    score = 'unpopular'

    # Check if the popularity score is between 50 and 75, inclusive
    if (row.popularity > 50) & (row.popularity < 75):
        score = 'medium'

    # Check if the popularity score is greater than or equal to 75
    elif row.popularity >= 75:
        score = 'popular'

    # Set the 'pop_rating' column for this row to the determined score
    classified.at[i, 'pop_rating'] = score

```

Figure 13. Screenshot of adding the pop_rating column

This code adds a new column called **pop_rating** to a DataFrame **df**. The new column will contain a rating classification based on the value of the 'popularity' column in each row of the original DataFrame.

First, the code makes a copy of the original DataFrame using the **copy()** method to avoid modifying the original data. Then, it initializes the new 'pop_rating' column with empty strings.

Next, it loops through each row in the DataFrame using the **iterrows()** method. For each row, it checks the value of the **popularity** column to determine its popularity rating classification. If the **popularity** value is between 50 and 75, the row is classified as 'medium'. If the **popularity** value is greater than or equal to 75, the row is classified as 'popular'. Otherwise, the row is classified as 'unpopular'.

Finally, the code updates the **pop_rating** column for each row using the **at()** method to set the value of the rating classification determined in the previous step.

⇒ Overall, this code is useful for creating a new column based on the values of an existing column in a DataFrame, which can help with data analysis and visualization.


```

# Create a new figure and axes with size (8, 5)
fig = plt.figure(figsize=(8, 5))
ax = fig.add_subplot(1, 1, 1)

# Create a countplot of the 'pop_rating' column in the 'classified' DataFrame
sns.countplot(x='pop_rating', data=classified, ax=ax)

# Set the x-axis label and title of the plot
ax.set_xlabel('Ratings', fontsize=14)
ax.set_title('Counts', fontsize=14)

# Show the plot
plt.show()

```

Figure 14. Screenshot of creating pop_rating countplot

This code uses the Seaborn library to create a countplot of the **pop_rating** column in the **classified** DataFrame. It creates a new figure with a specified size of 8 inches by 5 inches using Matplotlib's **figure** function and creates a single set of axes using the **add_subplot** method.

The **sns.countplot(x='pop_rating', data=classified, ax=ax)** line creates the countplot using Seaborn's **countplot** function. It specifies the **x** parameter as '**pop_rating**' to indicate that the **pop_rating** column should be used as the x-axis variable. The **data** parameter specifies the DataFrame to use for plotting, and the **ax** parameter specifies the axes object to use.

The next two lines of code use the **set_xlabel** and **set_title** methods of the **Axes** object to set the x-axis label and title of the plot, respectively. Both methods also specify a font size of 14 using the **fontsize** parameter.

Finally, the **plt.show()** function is called to display the plot.

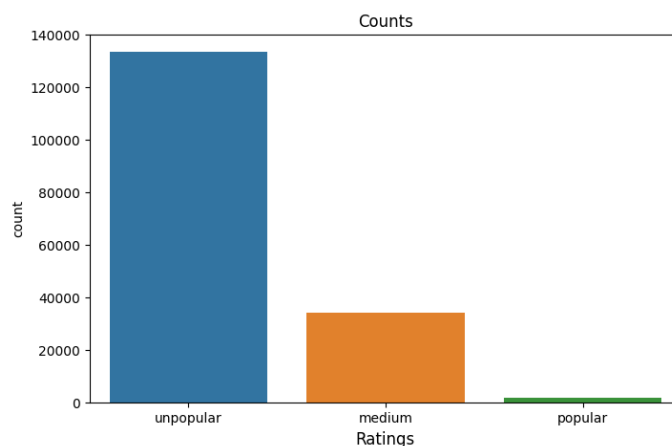


Figure 15. Graph of popularity ratings

The countplot appears to show the distribution of a categorical variable **pop_rating**, which has three possible values: "unpopular", "medium", and "popular".

The y-axis indicates the count of observations in each category, while the x-axis shows the categories themselves. Specifically, the count for the "unpopular" category is around 135,000, the count for the "medium" category is around 35,000, and the count for the "popular" category is around 5,000.

This information suggests that the majority of observations in the dataset are classified as "unpopular" (135,000 out of a total of 173,000, or roughly 78%). The "medium" and "popular" categories, on the other hand, account for only 20% and 2% of the total observations, respectively.

7.2.3 Exploratory Data Analysis (EDA)

7.2.3.1 The Relationship Between all the Features

```
# Select all columns with numeric data types
numeric_columns = df.columns[df.dtypes != 'object']

# Create a new DataFrame containing only the numeric columns
numeric_df = pd.DataFrame(data=df, columns=numeric_columns, index=df.index)

# Calculate the absolute correlation matrix for the numeric DataFrame
corr = np.abs(numeric_df.corr())

# Create a heatmap of the correlation matrix using seaborn
fig, ax = plt.subplots(figsize=(8, 8))
cmap = sns.color_palette("Blues")
sns.heatmap(corr, cmap=cmap, square=True)

# Set the title of the plot and display it
plt.title('Correlation of numerical characteristics')
plt.show()
```

Figure 16. Screenshot of creating the Correlation matrix using seaborn

This code selects all columns in the input DataFrame 'df' that have numeric data types and stores their names in the **numeric_columns** variable. It then creates a new DataFrame **numeric_df** that includes only the selected numeric columns from the original DataFrame.

Next, the code calculates the correlation matrix for the numeric DataFrame **numeric_df** using the **corr()** method, and takes the absolute value of each correlation coefficient using the **np.abs()** function.

Finally, the code creates a heatmap of the absolute correlation matrix using the seaborn library's **heatmap()** function. The **square=True** argument ensures that each cell in the heatmap is square-shaped.

The plot is then given a title using the **title()** method of the matplotlib **pyplot** interface, and the plot is displayed using the **show()** method of the same interface.

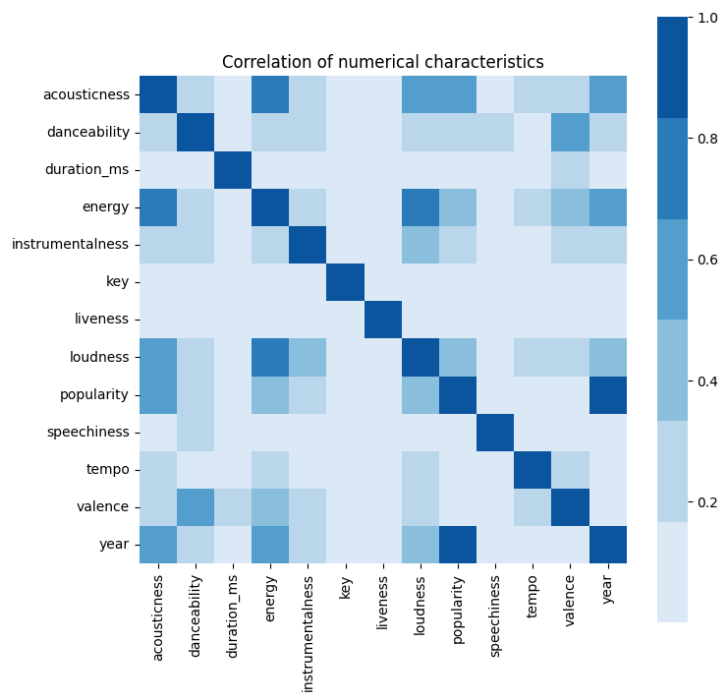


Figure 17. Correlation of numerical characteristics

The resulting plot shows the absolute correlation between each pair of numerical features in the DataFrame. The color of each square in the heatmap represents the strength of the correlation, with darker colors indicating stronger positive correlation and lighter colors indicating weaker or negative correlation.

7.2.3.2 Popularity's Most Linearly Connected Features

```
# Calculate the correlation between the 'popularity' column and all other numeric columns
# in the DataFrame and sort the correlations in descending order
corr = numeric_df.corr()[['popularity']].sort_values(by='popularity', ascending=False)

# Create a new figure with a size of 8x8 inches
plt.figure(figsize=(6, 8))

# Create a heatmap of the correlation matrix with annotations and the 'Blues' colormap
heatmap = sns.heatmap(corr, annot=True, cmap='Blues')

# Set the title of the plot with a font size of 14 points and a padding of 16 points
heatmap.set_title('The popularity most linearly connected characteristics', fontdict={'fontsize':12}, pad=16);
```

Figure 18. Screenshot of creating the heatmap of the correlation matrix

The code uses the **corr()** method to calculate the correlation matrix between all numeric columns in the DataFrame. It then selects only the 'popularity' column from the resulting matrix and sorts the correlations in descending order based on their absolute values.

Next, the code creates a new figure with a size of 6x8 inches using **plt.figure()**. It then creates a heatmap of the correlation matrix using the **sns.heatmap()** function from the seaborn library, with annotations turned on and the **Blues** colormap specified.

Finally, the code sets the title of the plot using the **set_title()** method of the heatmap object. The title is centered with a font size of 12 points and a padding of 16 points.

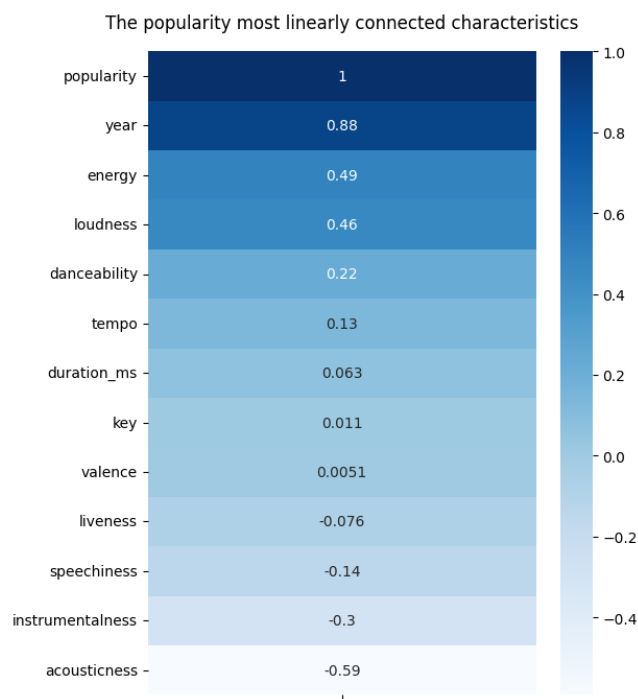


Figure 19. Heatmap of the popularity most linearly connected characteristics

This code snippet calculates the correlation between the **popularity** column and all other numeric columns in the DataFrame, sorts the correlations in descending order based on their absolute values, creates a heatmap of the resulting correlation matrix, and sets the title of the plot.

The resulting plot shows the most linearly connected characteristics to the **popularity** column in the dataset. The color of each square in the heatmap represents the strength and direction of the correlation, with blue shades indicating positive correlation and red shades indicating negative correlation.

The use of the heatmap and annotations in the plot makes it easy to identify the most strongly correlated variables with the **popularity** column. This information can be useful for further analysis and modeling tasks, such as predicting the popularity of new items based on their characteristics. However, it is important to keep in mind that correlation does not necessarily imply causation, and additional analysis may be required to establish the direction of the relationship between variables.

⇒ As we can see in the graph below, ‘popularity’ has a high link with ‘year’, ‘loudness’, and ‘energy’.

7.2.3.3 Mean Popularity Across the Year

⇒ According to the heatmap above, the association between popularity and year is substantial, which makes sense given that most popular music is new. The popularity objective is linearly close to the year characteristic (corr = 0.88). This characteristic will be omitted from our models due to the very high linear association between the year and popularity.

```
# Load the data into a pandas DataFrame
df = pd.read_csv('data.csv')

# Create a new figure with a size of 14x4 inches
fig, ax = plt.subplots(figsize=(14, 4))

# Group the data by year and calculate the mean of the 'popularity' column for each year
mean_popularity = df.groupby('year')['popularity'].mean()

# Create a line plot of the mean popularity over time
ax.plot(mean_popularity.index, mean_popularity.values, color='blue')

# Set the plot title, y-axis label, and x-axis label
ax.set_title('Mean Popularity across The years', c='black', weight='bold')
ax.set_ylabel('Mean Popularity', weight='bold')
ax.set_xlabel('Year', weight='bold')

# Set the x-ticks to show every 5 years from 1920 to 2020
ax.set_xticks(range(1920, 2021, 5))

# Display the plot
plt.show()
```

Figure 20. Screenshot of creating the graph of mean popularity across the year

The code creates a new figure with a size of 14x4 inches and calculates the mean popularity for each year by grouping the data by year and taking the mean of the 'popularity' column.

Then, it creates a line plot of the mean popularity over time using the **plot** function of the axes object. This time, the x-values and y-values are extracted from the Pandas Series returned by the **groupby** operation.

The code sets the plot title, y-axis label, and x-axis label using the **set_title**, **set_ylabel**, and **set_xlabel** functions of the axes object. It also sets the x-ticks to show every 5 years from 1920 to 2020 using the **set_xticks** function.

Finally, the code displays the plot using the 'show' function of the **pyplot** module.

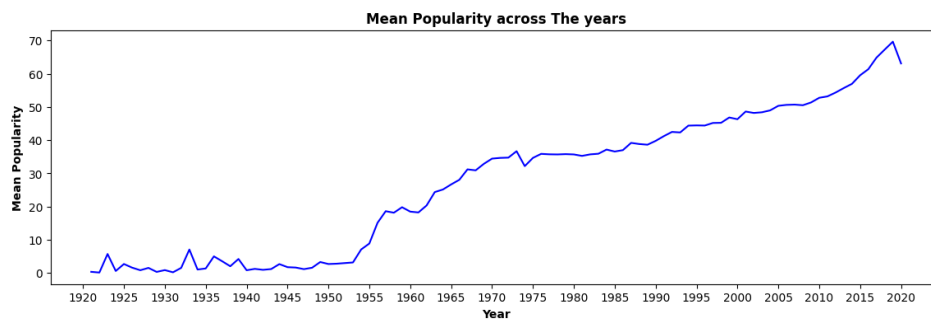


Figure 21. Graph of mean popularity across the year

```
# Define a function to create a scatterplot with a regression line
def regress_plot(x='', y='', data=None, xlab='', ylab='', titl=''):
    """Plots a scatterplot with a regression line using given inputs"""

    # Group the data by the x column, calculate the mean of the y column for each x value, and convert to a DataFrame
    data = data.groupby(x)[y].mean().to_frame().reset_index()

    # Create a figure and axes with the desired size
    fig, ax = plt.subplots(figsize=(10,6))

    # Create a scatterplot with a regression line using the specified x and y columns, and set the color and size of the points and the color of the line
    _ = sns.regplot(x=x, y=y, data=data, scatter_kws={'color': 'b', "s": 10}, line_kws={'color':'black'})

    # Set the label for the x-axis and the font size
    _ = plt.xlabel(xlab, fontsize=12)

    # Set the label for the y-axis and the font size
    _ = plt.ylabel(ylab, fontsize=12)

    # Set the title of the plot, the font size, and the color
    _ = plt.title(titl, fontsize=14, c='black')

    # Set the y-axis limits to the specified range
    _ = plt.ylim(-3, 103)

    # Show the plot
    plt.show()
```

Figure 22. Code of the function that create the scatterplot

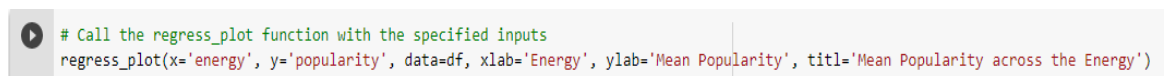
The provided code defines a function called **regress_plot** that creates a scatterplot with a regression line.

This function takes several parameters including the column names for x and y, the data to be plotted, x-axis label, y-axis label, and plot title. It calculates the mean of the y column for each unique value of the x column, creates a scatterplot with a regression line using the specified data, and customizes the plot with labels and titles. Finally, it displays the plot using **plt.show()**.

⇒ As predicted, popularity is strongly connected to the year of release. This makes sense because the Spotify algorithm that makes this choice calculates "popularity" based not just on how many streams a song receives, but also on how recent those streams are.

7.2.3.4 Mean Popularity Across Energy

⇒ According to the correlation heatmap, energy seems to be the greatest predictor. There appears to be a somewhat significant linear correlation of 0.49 to the goal. In terms of intensity and activity, a fast-paced track seems quicker, louder, and noisier.

A code snippet in a light gray box with a dark gray border. It starts with a play button icon in a circle. The code is:

```
# Call the regress_plot function with the specified inputs
regress_plot(x='energy', y='popularity', data=df, xlab='Energy', ylab='Mean Popularity', titl='Mean Popularity across the Energy')
```

*Figure 23. Code of creating the regress_plot function
(mean popularity across the energy)*

This code should create a scatterplot with a regression line showing the relationship between energy and mean popularity.

- x: 'energy'
- y: 'popularity'
- data: df
- xlab: 'Energy'
- ylab: 'Mean Popularity'
- title: 'Mean Popularity across the Energy'

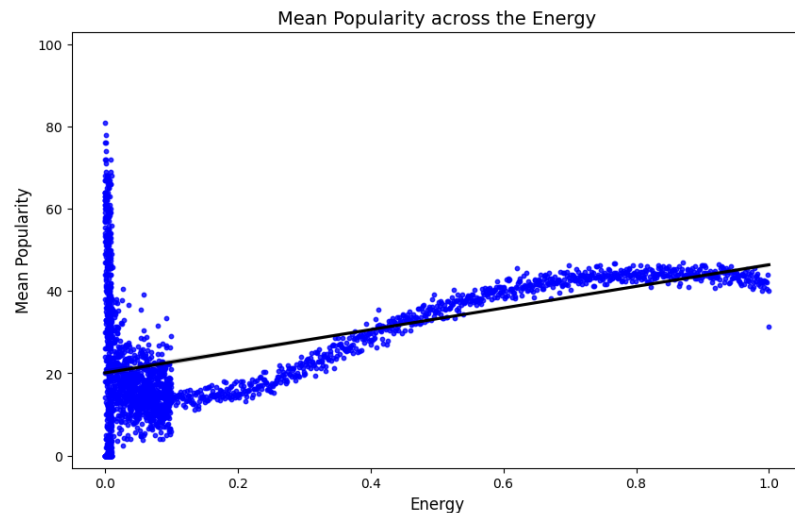


Figure 24. Graph of mean popularity across the energy

⇒ The popularity of a song appears to be influenced by its energy as well. Many popular songs are upbeat, yet they are not always dancing music. Low energy songs have the potential to be more popular because the association is not too strong.

7.2.3.5 Mean Popularity Across Loudness

```
# Call the regress_plot function with the specified inputs
regress_plot(x='loudness', y='popularity', data=df, xlab='Loudness', ylab='Mean Popularity', title='Mean Popularity across Loudness')
```

Figure 25. Code of creating the regress_plot function (mean popularity across loudness)

This code should create a scatterplot with a regression line showing the relationship between loudness and mean popularity.

- x: 'loudness'
- y: 'popularity'
- data: df
- xlab: 'Loudness'
- ylab: 'Mean Popularity'
- title: 'Mean Popularity across the Loudness'

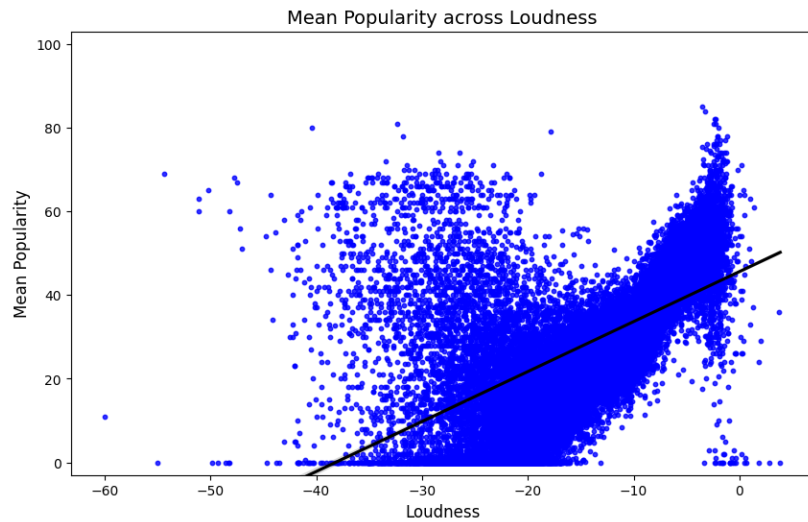


Figure 26. Graph of mean popularity across loudness

⇒ The regressor clearly shows there is a positive correlation, and the data tends to narrow towards the top, but overall the observations are quite scattered. It's not a very strong fit.

7.2.3.6 Mean Popularity Across Danceability

```
# Call the regress_plot function with the specified inputs
regress_plot(x='danceability', y='popularity', data=df, xlab='Danceability', ylab='Mean Popularity', titl='Mean Popularity across Danceability')
```

Figure 27. Code of creating the regress_plot function (mean popularity across danceability)

This code should create a scatterplot with a regression line showing the relationship between danceability and mean popularity.

- x: 'danceability'
- y: 'popularity'
- data: df
- xlab: 'Danceability'
- ylab: 'Mean Popularity'
- title: 'Mean Popularity across the Danceability'

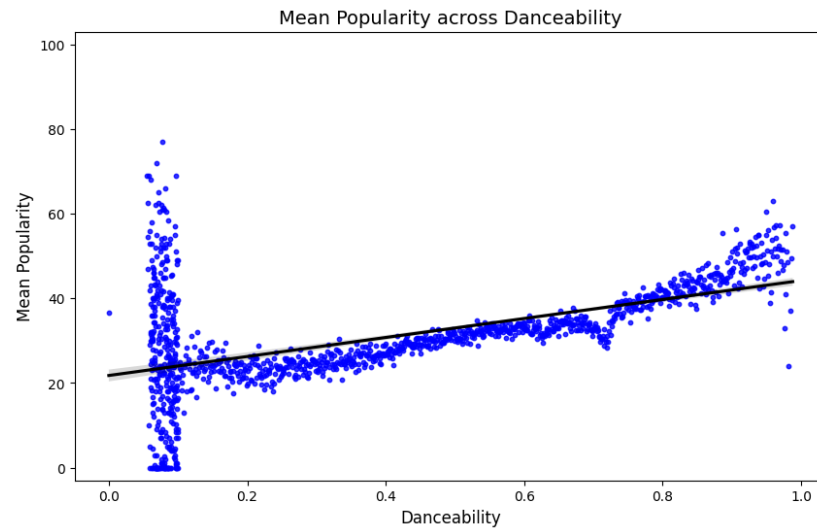


Figure 28. Graph of mean popularity across danceability

⇒ The mix of pace rhythm and beat power determines whether a music is danceable. Danceability appears to have a normal distribution from 0 to 1. The majority of Danceability scores range from 0.2 to 0.8.

7.2.3.7 Mean Popularity Across Tempo

```
# Call the regress_plot function with the specified inputs
regress_plot(x='tempo', y='popularity', data=df, xlab='Tempo', ylab='Mean Popularity', title='Mean Popularity across Tempo')
```

Figure 29. Code of creating the regress_plot function (mean popularity across tempo)

This code should create a scatterplot with a regression line showing the relationship between tempo and mean popularity.

- x: 'tempo'
- y: 'popularity'
- data: df
- xlab: 'Tempo'
- ylab: 'Mean Popularity'
- title: 'Mean Popularity across Tempo'

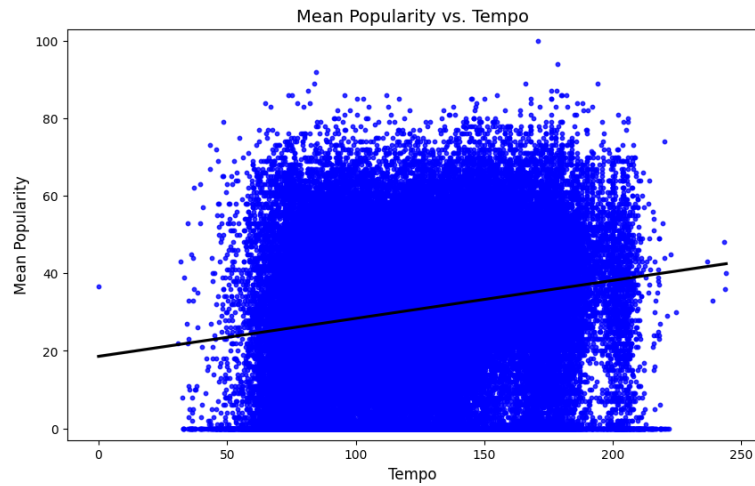


Figure 30. Graph of mean popularity across tempo

⇒ The majority of tempo scores range are from 50 to 200.

7.2.3.8 Mean Popularity Across Acousticness

```
# Call the regress_plot function with the specified inputs
regress_plot(x='acousticness', y='popularity', data=df, xlab='Acousticness', ylab='Mean Popularity', title='Mean Popularity across Acousticness')
```

Figure 31. Code of creating the regress_plot function (mean popularity across acousticness)

This code should create a scatterplot with a regression line showing the relationship between acousticness and mean popularity.

- x: 'acousticness'
- y: 'popularity'
- data: df
- xlab: 'Acousticness'
- ylab: 'Mean Popularity'
- title: 'Mean Popularity across Acousticness'

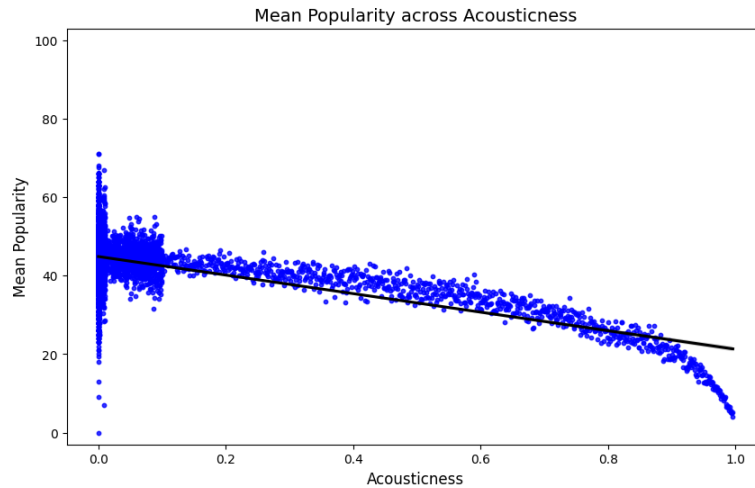


Figure 32. Graph of mean popularity across acousticness

⇒ Acousticness appears to be unrelated to popularity. Most popular songs today use electronic or electric instruments. It is quite uncommon for a piece of music performed by a chamber orchestra or simply acoustic band to become extremely popular.

7.2.3.9 Conclusion

The goal of this study is to see if an algorithm can outperform humans when it comes to spotting similarities and differences between unpopular and popular music.

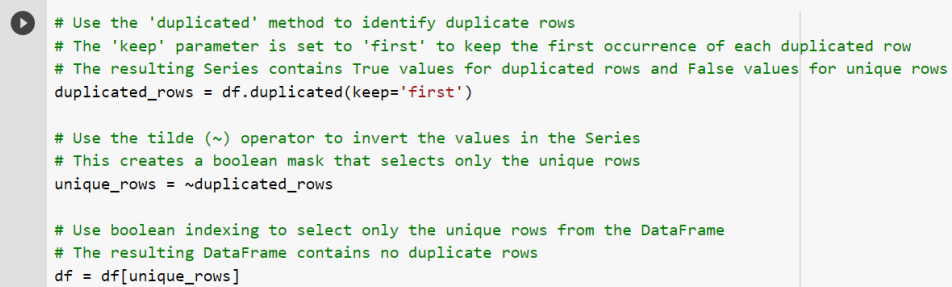
According to the study, writing high-energy songs utilizing electric or electronic instruments increases the probability of success.

The dataset contains 12 attributes, five of which show a negative association with popularity and seven of which show a positive correlation.

We need to create a machine learning model that can use these features to forecast a song's popularity.

7.2.4 Initialisation

7.2.4.1 Preparation

A screenshot of a code editor showing Python code for removing duplicate rows from a pandas DataFrame. The code includes comments explaining each step: using 'df.duplicated()' with 'keep='first'', inverting the resulting Series with '~', and selecting unique rows with 'df[unique_rows]'.

```
# Use the 'duplicated' method to identify duplicate rows
# The 'keep' parameter is set to 'first' to keep the first occurrence of each duplicated row
# The resulting Series contains True values for duplicated rows and False values for unique rows
duplicated_rows = df.duplicated(keep='first')

# Use the tilde (~) operator to invert the values in the Series
# This creates a boolean mask that selects only the unique rows
unique_rows = ~duplicated_rows

# Use boolean indexing to select only the unique rows from the DataFrame
# The resulting DataFrame contains no duplicate rows
df = df[unique_rows]
```

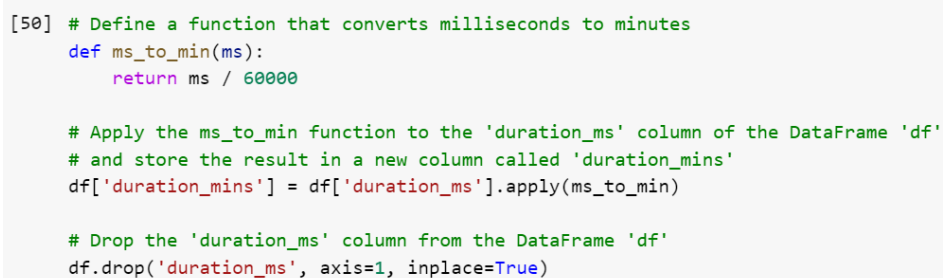
Figure 33. Screenshot of the preparation code 1

This code identifies and removes duplicate rows from a pandas DataFrame **df**.

The **duplicated** method is used to identify duplicate rows in the DataFrame. The **keep** parameter is set to 'first', which means that the first occurrence of each duplicated row will be marked as False, and all subsequent occurrences will be marked as True. The resulting Series **duplicated_rows** contains True values for duplicated rows and False values for unique rows. The tilde (~) operator is then used to invert the values in the **duplicated_rows** Series, creating a Boolean mask that selects only the unique rows.

The resulting Series **unique_rows** contains True values for unique rows and False values for duplicated rows.

Finally, Boolean indexing is used to select only the unique rows from the DataFrame by passing **unique_rows** as a Boolean mask to the DataFrame. The resulting DataFrame **df** contains no duplicate rows and only unique rows.

A screenshot of a code editor showing Python code for converting milliseconds to minutes. It includes a function definition 'ms_to_min' and its application to a DataFrame column 'duration_ms', followed by dropping the original column.

```
[50] # Define a function that converts milliseconds to minutes
def ms_to_min(ms):
    return ms / 60000

# Apply the ms_to_min function to the 'duration_ms' column of the DataFrame 'df'
# and store the result in a new column called 'duration_mins'
df['duration_mins'] = df['duration_ms'].apply(ms_to_min)

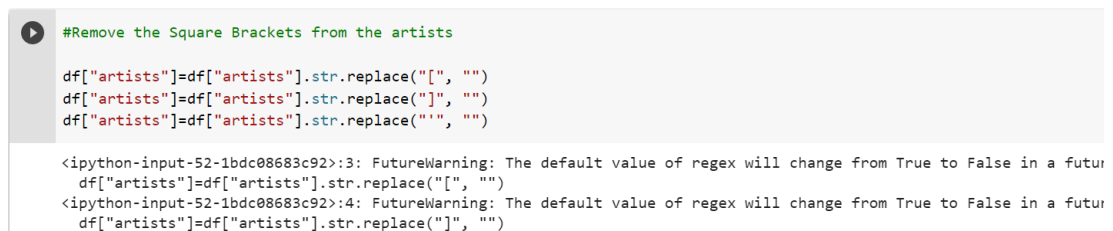
# Drop the 'duration_ms' column from the DataFrame 'df'
df.drop('duration_ms', axis=1, inplace=True)
```

Figure 34. Screenshot of the preparation code 2

This code defines a function **ms_to_min** that takes a value in milliseconds and returns the equivalent value in minutes by dividing the input by 60000.

Then, the **apply** method is used to apply the **ms_to_min** function to each value in the **duration_ms** column of the DataFrame **df**. The resulting values are stored in a new column called **duration_mins**.

Finally, the **drop** method is used to remove the **duration_ms** column from the DataFrame **df**. The **axis** parameter is set to 1 to indicate that we want to drop a column, and the **inplace** parameter is set to **True** to modify the DataFrame **df** in place.



```
#Remove the Square Brackets from the artists
df["artists"]=df["artists"].str.replace("[", "")
df["artists"]=df["artists"].str.replace("]", "")
df["artists"]=df["artists"].str.replace("'", "")

<ipython-input-52-1bdc08683c92>:3: FutureWarning: The default value of regex will change from True to False in a futur
df["artists"]=df["artists"].str.replace("[", "")
<ipython-input-52-1bdc08683c92>:4: FutureWarning: The default value of regex will change from True to False in a futur
df["artists"]=df["artists"].str.replace("'", "")
```

Figure 35. Screenshot of the preparation code 3

This code removes square brackets and single quotes from the **artists** column of a pandas DataFrame **df**.

The **str.replace()** method is used to replace the opening square bracket ('['), closing square bracket (']'), and single quotes ("'") with empty strings("").

The first line removes the opening square bracket from the **artists** column by replacing it with an empty string. The second line removes the closing square bracket from the **artists** column by replacing it with an empty string. The third line removes the single quotes from the **artists** column by replacing them with empty strings.

After these operations, the **artists** column no longer contains square brackets or single quotes, making it easier to work with the data.

```

# Make a copy of the original DataFrame
data = df.copy()

# Create a new column "popularity_level" and assign a value of 1 to rows where "popularity" is between 0 and 30 (inclusive)
data.loc[(df.popularity >= 0) & (df.popularity <= 30), "popularity_level"] = 1

# Assign a value of 2 to rows where "popularity" is between 31 and 60 (inclusive)
data.loc[(df.popularity > 30) & (df.popularity <= 60), "popularity_level"] = 2

# Assign a value of 3 to rows where "popularity" is between 61 and 100 (inclusive)
data.loc[(df.popularity > 60) & (df.popularity <= 100), "popularity_level"] = 3

# Convert the "popularity_level" column to integer type
data["popularity_level"] = data["popularity_level"].astype("int")

# Print the first 10 rows of the DataFrame to check the changes
print(data.head(10))

```

	acousticness	artists	danceability	energy
0	0.995	Carl Woitschach	0.700	0.1950
1	0.994	Robert Schumann, Vladimir Horowitz	0.379	0.0135
2	0.604	Seweryn Gosczyński	0.749	0.2200
3	0.995	Francisco Canaro	0.781	0.1300
4	0.990	Frédéric Chopin, Vladimir Horowitz	0.210	0.2040
5	0.995	Felix Mendelssohn, Vladimir Horowitz	0.424	0.1200
6	0.956	Franz Liszt, Vladimir Horowitz	0.444	0.1970
7	0.988	Carl Woitschach	0.555	0.4210
8	0.995	Francisco Canaro, Charlo	0.683	0.2070
9	0.846	Seweryn Gosczyński	0.674	0.2050

	instrumentalness	key	liveness	loudness	popularity	speechiness
0	0.563	10	0.1510	-12.428	0	0.0506
1	0.901	8	0.0763	-28.454	0	0.0462
2	0.000	5	0.1190	-19.924	0	0.9290
3	0.887	1	0.1110	-14.734	0	0.0926
4	0.908	11	0.0900	-16.829	1	0.0424
5	0.911	6	0.0915	-19.242	0	0.0593
6	0.435	11	0.0744	-17.226	0	0.0400
7	0.836	1	0.1050	-9.878	0	0.0474
8	0.206	9	0.3370	-9.801	0	0.1270
9	0.000	9	0.1700	-20.119	0	0.9540

	tempo	valence	year	duration_mins	popularity_level
0	118.469	0.7790	1928	2.644133	1
1	83.972	0.0767	1928	4.702217	1
2	107.177	0.8800	1928	1.738333	1
3	108.003	0.7200	1928	3.012667	1
4	62.149	0.0693	1928	11.462217	1
5	63.521	0.2660	1928	5.876667	1
6	80.495	0.3050	1928	2.277117	1
7	123.310	0.8570	1928	2.566117	1
8	119.833	0.4930	1928	2.708217	1
9	81.249	0.7590	1928	1.860000	1

Figure 36. Screenshot of arranging the popularity values

This code essentially creates a new column **popularity_level** in the DataFrame **df** based on the value in the existing **popularity** column. The values in the new column are assigned based on a range of values in the "popularity" column. Finally, the code converts the **popularity_level** column to an integer type and prints the first 10 rows of the updated DataFrame to check the changes.

```

[25] # Count the number of occurrences of each value in the "popularity_level" column
popularity_counts = data["popularity_level"].value_counts()

# Print the result
print(popularity_counts)

```

```

2    77538
1    76094
3    15608
Name: popularity_level, dtype: int64

```

Figure 37. Result of the occurrences

This code counts the number of occurrences of each value in the **popularity_level** column of the DataFrame **data**. The result is stored in a variable called **popularity_counts**. Finally, the code prints the result to the console.

```
# Define a list of artists
artists = ['Drake', 'Lady Gaga', 'Taylor Swift', 'The Weeknd']

# Create a list of indices corresponding to the artists above
# Iterate over each artist and find the indices where the 'artists' column matches the artist name
to_drop = [data[data.artists == name].index.tolist() for name in artists]

# Flatten the list of lists into a single list
to_drop = [ind for sub in to_drop for ind in sub]

# Gather the test cases by selecting rows from the original data using the indices to drop
cases = data[data.index.isin(to_drop)]

# Remove the test cases from the original data by dropping the corresponding rows
data.drop(to_drop, inplace=True)

# Drop unnecessary columns from the modified data
data = data.drop(["popularity", "explicit", "id", "mode", "name", "release_date", "artists"], axis=1)

# Print the head of the modified data
data.head()
```

	acousticness	danceability	energy	instrumentalness	key	liveness	loudness	speechiness	tempo	valence	year	duration_mins	popularity_level
0	0.995	0.708	0.1950	0.563	10	0.1510	-12.428	0.0506	118.469	0.7790	1928	2.644133	1
1	0.994	0.379	0.0135	0.901	8	0.0763	-28.454	0.0462	83.972	0.0767	1928	4.702217	1
2	0.604	0.749	0.2200	0.000	5	0.1190	-19.924	0.9290	107.177	0.8800	1928	1.738333	1
3	0.995	0.781	0.1300	0.887	1	0.1110	-14.734	0.0926	108.003	0.7200	1928	3.012667	1
4	0.990	0.210	0.2040	0.908	11	0.0980	-16.829	0.0424	62.149	0.0693	1928	11.462217	1

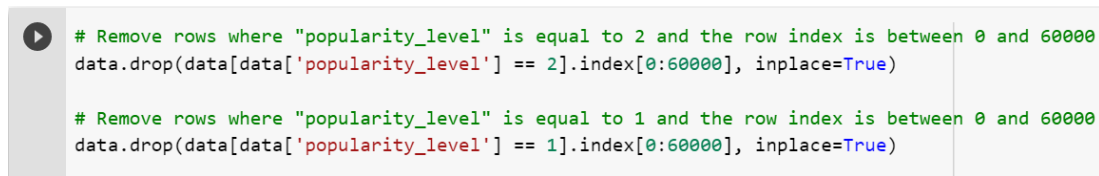
Figure 38. Screenshot of defining the list of artists and dropping unnecessary columns

This code defines a list of artists. Then, iterate over each artist and find the indices where the 'artists' column matches the artist's name in the original **data** DataFrame. These indices are collected in the **to_drop** list. Next, it flattens the list of lists in **to_drop** into a single list using list comprehension.

After that, it gathers the test cases by selecting the rows from the original **data** DataFrame where the indices are present in the **to_drop** list. The resulting DataFrame is stored in the **cases** variable.

Subsequently, it removes the test cases from the original **data** DataFrame by dropping the corresponding rows using the **drop** method with **inplace=True**.

Finally, it drops unnecessary columns from the modified **data** DataFrame by specifying the column names to drop in the **drop** method with the **axis=1** parameter. The resulting DataFrame is assigned back to the variable **data**. The **head()** method is used to display the first few rows of the modified **data** DataFrame.

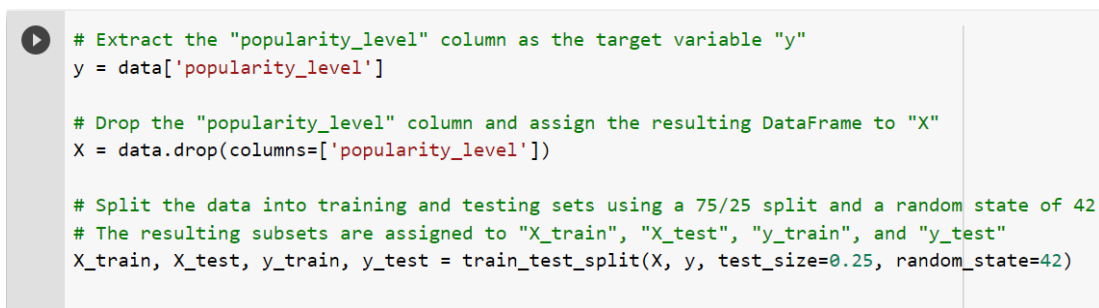


```
# Remove rows where "popularity_level" is equal to 2 and the row index is between 0 and 60000
data.drop(data[data['popularity_level'] == 2].index[0:60000], inplace=True)

# Remove rows where "popularity_level" is equal to 1 and the row index is between 0 and 60000
data.drop(data[data['popularity_level'] == 1].index[0:60000], inplace=True)
```

Figure 39. Screenshot of removing rows

This code drops rows from the DataFrame **data** where **popularity_level** is equal to 2 and the row index is between 0 and 60000. It then drops rows where **popularity_level** is equal to 1 and the row index is between 0 and 60000. The **inplace=True** argument is used to modify the original DataFrame rather than creating a new one.



```
# Extract the "popularity_level" column as the target variable "y"
y = data['popularity_level']

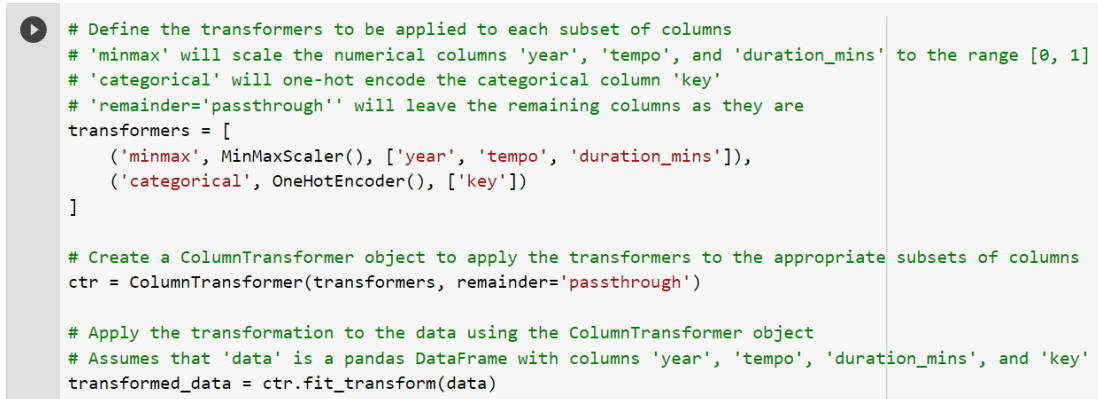
# Drop the "popularity_level" column and assign the resulting DataFrame to "X"
X = data.drop(columns=['popularity_level'])

# Split the data into training and testing sets using a 75/25 split and a random state of 42
# The resulting subsets are assigned to "X_train", "X_test", "y_train", and "y_test"
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Figure 40. Screenshot of splitting dataframe into features

This code splits the DataFrame **data** into features (in the DataFrame "X") and target variable (in the Series "y"). The **popularity_level** column is removed from the features and assigned to the target variable. The data is then split into training and testing sets using a 75/25 split and a random state of 42. The subsets of the data are assigned to **X_train**, **X_test**, **y_train**, and **y_test**. The **train_test_split()** function is part of the scikit-learn library, which is commonly used for machine learning tasks in Python.

7.2.4.2 Transformations of Characteristics

A screenshot of a Jupyter Notebook cell. On the left, there is a play button icon. The cell contains Python code for using a ColumnTransformer. The code defines a list of transformers, creates a ColumnTransformer object, and applies it to a dataset. Comments explain the purpose of each step.

```
# Define the transformers to be applied to each subset of columns
# 'minmax' will scale the numerical columns 'year', 'tempo', and 'duration_mins' to the range [0, 1]
# 'categorical' will one-hot encode the categorical column 'key'
# 'remainder='passthrough'' will leave the remaining columns as they are
transformers = [
    ('minmax', MinMaxScaler(), ['year', 'tempo', 'duration_mins']),
    ('categorical', OneHotEncoder(), ['key'])
]

# Create a ColumnTransformer object to apply the transformers to the appropriate subsets of columns
ctr = ColumnTransformer(transformers, remainder='passthrough')

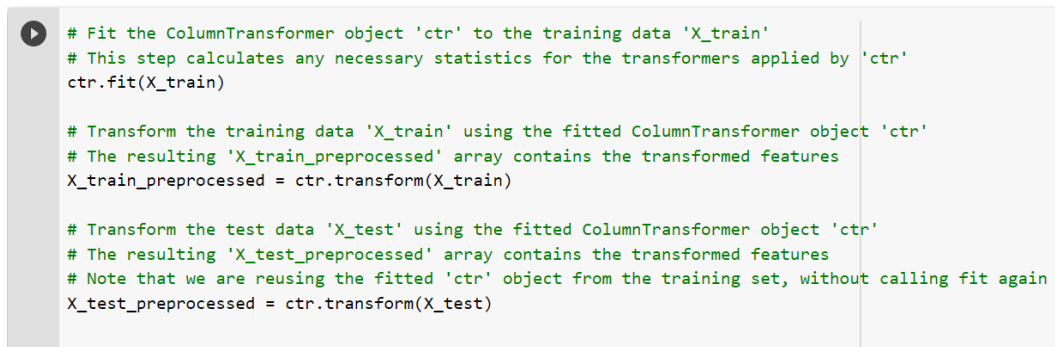
# Apply the transformation to the data using the ColumnTransformer object
# Assumes that 'data' is a pandas DataFrame with columns 'year', 'tempo', 'duration_mins', and 'key'
transformed_data = ctr.fit_transform(data)
```

Figure 41. Screenshot of using the ColumnTransformer object

This code applies feature scaling and one-hot encoding to specific subsets of columns in a pandas DataFrame using a **ColumnTransformer** object from the **sklearn.compose** module. Here's how the code works:

- **transformers** is a list of tuples, where each tuple contains the following information:
 - A string identifier for the transformer, which is used to label the columns in the output.
 - An instance of a transformer class, such as **MinMaxScaler** or **OneHotEncoder**, that will be used to transform the selected columns.
 - A list of column names or indices that should be transformed by the corresponding transformer.
- **ctr** is a **ColumnTransformer** object that is initialized with the list of transformers and the **remainder** parameter set to 'passthrough', which means that any columns not explicitly selected for transformation will be included in the output without modification.
- **transformed_data** is a new pandas DataFrame that contains the transformed columns as well as any remaining columns from the original DataFrame.

Note that this code assumes that the input DataFrame **data** has columns named (**year**, **tempo**, **duration_mins**, and **key**) that correspond to the columns selected for transformation by the **MinMaxScaler** and **OneHotEncoder**. If these column names are different, the code will need to be modified accordingly.

A screenshot of a code editor showing Python code for data transformation. The code is written in green text on a light gray background. It includes comments explaining each step: fitting the ColumnTransformer on training data, transforming the training data, and transforming the test data using the same fitted object. The code uses variables like 'ctr', 'X_train', 'X_test', 'X_train_preprocessed', and 'X_test_preprocessed'.

```
# Fit the ColumnTransformer object 'ctr' to the training data 'X_train'
# This step calculates any necessary statistics for the transformers applied by 'ctr'
ctr.fit(X_train)

# Transform the training data 'X_train' using the fitted ColumnTransformer object 'ctr'
# The resulting 'X_train_preprocessed' array contains the transformed features
X_train_preprocessed = ctr.transform(X_train)

# Transform the test data 'X_test' using the fitted ColumnTransformer object 'ctr'
# The resulting 'X_test_preprocessed' array contains the transformed features
# Note that we are reusing the fitted 'ctr' object from the training set, without calling fit again
X_test_preprocessed = ctr.transform(X_test)
```

Figure 42. Screenshot of transforming the X & y train

The code fits **ColumnTransformer** object **ctr** to the training data **X_train** using the **fit** method. This step calculates any necessary statistics for the transformers applied by **ctr**.

Then, the code applies the fitted **ctr** object to both the training and test data using the **transform** method. The transformed data is stored in **X_train_preprocessed** and **X_test_preprocessed**, respectively.

It is important to note that the **fit** method is only called on the training data, not on the test data. The same **ctr** object is reused for transforming the test data, without calling 'fit' again. This ensures that the test data is transformed using the same statistics as the training data, which is important for ensuring that the evaluation of the model on the test data is meaningful.

7.2.5 Prediction Models

```
results = []

def run_model(model, alg_name):
    # construct the model using training data
    model.fit(X_train, y_train)

    # create forecasts based on test results
    y_pred = model.predict(X_test)

    # determine the accuracy score
    accuracy = accuracy_score(y_test, y_pred)

    # Calculate the confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Perform cross-validation and calculate scores
    scoresDT3 = cross_val_score(model, X_test, y_test, cv=6)

    # Generate the classification report
    Cr = classification_report(y_test, y_pred)

    # Append the algorithm name, accuracy, and model to the results list
    results.append((alg_name, accuracy, model))

    # Print the model name and accuracy score on the test set
    print("Model: ", alg_name)
    print("Accuracy on Test Set for {} = {:.2f}\n".format(alg_name, accuracy))

    # Print the classification report
    print(Cr)

    # Print the cross-validation accuracy mean and standard deviation
    print("{}: CrossVal Accuracy Mean: {:.2f} and Standard Deviation: {:.2f} \n".format(alg_name, scoresDT3.mean(), scoresDT3.std()))
```

Figure 43. Screenshot of defining the `run_model` function

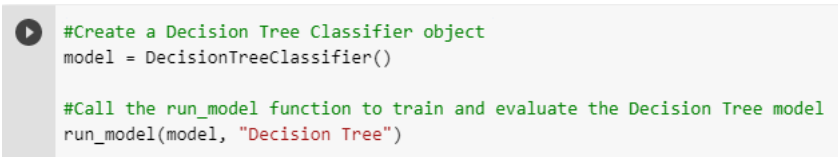
The provided code defines a function **run_model** that takes a machine learning model, along with its algorithm name, as input. The function trains the model on a training dataset (**X_train** and **y_train**), makes predictions on a test dataset (**X_test**), and evaluates the performance of the model.

Here's a breakdown of what the code does:

- The **model.fit(X_train, y_train)** line trains the model on the training data.
- The **model.predict(X_test)** line generates predictions for the test data.
- The **accuracy_score(y_test, y_pred)** line calculates the accuracy of the model by comparing the predicted labels (**y_pred**) with the true labels (**y_test**).
- The **confusion_matrix(y_test, y_pred)** line calculates the confusion matrix, which shows the counts of true positive, true negative, false positive, and false negative predictions.

- The `cross_val_score(model, X_test, y_test, cv=6)` line performs cross-validation on the test data using a 6-fold cross-validation strategy and calculates scores for each fold.
- The `classification_report(y_test, y_pred)` line generates a classification report, which includes precision, recall, F1-score, and support for each class.
- The results, including the algorithm name, accuracy, and model, are appended to the **results** list.
- The algorithm name and accuracy on the test set are printed.
- The classification report is printed.
- The cross-validation accuracy mean, and standard deviation are printed.

7.2.5.1 Decision Tree



```
#Create a Decision Tree Classifier object
model = DecisionTreeClassifier()

#Call the run_model function to train and evaluate the Decision Tree model
run_model(model, "Decision Tree")
```

Figure 44. Decision Tree code

This code first executed a simple decision tree classifier. Because this only generates one tree, it is not computationally intensive, but it is prone to overfitting. It calls a function named `run_model` to train and evaluate a Decision Tree Classifier model.

The first line creates an instance of the **DecisionTreeClassifier** class and assigns it to the variable **model**. The **DecisionTreeClassifier** is a class from the scikit-learn library that implements the decision tree algorithm for classification tasks. By calling **DecisionTreeClassifier()**, the code creates a new decision tree model with default parameter values.

The second line attempts to call a function named **run_model** and pass the **model** object (the Decision Tree Classifier) as the first argument and the string "**Decision Tree**" as the second argument.

The purpose of the **run_model** function is to train and evaluate the model, and it is assumed that it returns the accuracy score of the model's predictions.

```
Model: Decision Tree
Accuracy on Test Set for Decision Tree = 0.74

      precision    recall  f1-score   support

     1       0.90      0.90      0.90      4147
     2       0.67      0.65      0.66      4338
     3       0.66      0.68      0.67      3869

 accuracy            0.74      12354
 macro avg           0.74      0.74      0.74      12354
 weighted avg        0.74      0.74      0.74      12354

Decision Tree: CrossVal Accuracy Mean: 0.74 and Standard Deviation: 0.00
```

Figure 45. Decision Tree accuracy result

⇒ The provided output shows the results of evaluating the Decision Tree model on a test set. Here's an explanation of the different components:

- **Accuracy on Test Set for Decision Tree = 0.74:**

This line indicates that the accuracy of the Decision Tree model on the test set is 0.74. The accuracy represents the proportion of correct predictions made by the model.

- **Precision, Recall, F1-Score, and Support:**

The following lines display precision, recall, F1-score, and support for each class in the classification task. These metrics provide insights into how well the model performs for each individual class in the dataset.

- **Precision:** It measures the proportion of correctly predicted instances of a class among all instances predicted as that class.
- **Recall:** It calculates the proportion of correctly predicted instances of a class among all instances that actually belong to that class.

- F1-Score: It is the harmonic mean of precision and recall, providing a single metric that balances both measures.
- Support: It represents the number of instances in each class present in the test set.

- **Accuracy, Macro Average, and Weighted Average:**

The following lines provide overall metrics for the model's performance:

- Accuracy: It represents the accuracy of the model on the entire test set, considering all classes.
- Macro Average: It calculates the average performance across all classes, giving equal weight to each class. It provides an overall measure of the model's performance without considering class imbalances.
- Weighted Average: It calculates the weighted average of precision, recall, and F1-score, considering the number of instances in each class. It takes into account class imbalances and provides an overall performance measure.

- **Decision Tree: CrossVal Accuracy Mean: 0.74 and Standard Deviation: 0.00:**

This line indicates the mean and standard deviation of the cross-validated accuracy scores for the Decision Tree model. Cross-validation is a technique used to assess a model's performance on multiple subsets of the data. Here, the mean accuracy is 0.74, suggesting that the model performs consistently across different subsets of the data. The standard deviation of 0.00 indicates low variability in the accuracy scores, indicating a stable model performance.

⇒ Overall, the results show that the Decision Tree model achieves an accuracy of 0.74 on the test set and provides additional performance metrics for individual classes and overall averages. The model's stability is indicated by the low standard deviation of the cross-validated accuracy scores.

```
from sklearn.metrics import confusion_matrix

def run_model(model, alg_name):
    # Build the model on training data
    model.fit(X_train, y_train)

    # Make predictions for test data
    y_pred = model.predict(X_test)

    # Calculate the accuracy score
    accuracy = accuracy_score(y_test, y_pred)

    # Create the confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Print the confusion matrix
    print("Confusion Matrix:")
    print(cm)

    # Store the results
    results.append((alg_name, accuracy, model))

# Create a Decision Tree Classifier object
model = DecisionTreeClassifier()

# Call the run_model function to train and evaluate the Decision Tree model
run_model(model, "Decision Tree")
```

Figure 46. Screenshot of the confusion matrix code (Decision Tree)

- The code imports the **confusion_matrix** function from scikit-learn's **metrics** module.
- The **run_model** function is defined, which takes two arguments: **model** (the model object to be trained and evaluated) and **alg_name** (a string representing the name of the algorithm).
- Inside the **run_model** function:
 - The model is trained using the **fit** method on the training data (**X_train** and **y_train**).
 - Predictions are made on the test data (**X_test**) using the trained model's **predict** method, and the predictions are stored in **y_pred**.
 - The accuracy score is computed by comparing the predicted labels **y_pred** with the true labels **y_test** using the **accuracy_score** function.

- The confusion matrix is calculated by passing the true labels **y_test** and the predicted labels **y_pred** to the **confusion_matrix** function.
 - The confusion matrix is printed to the console using the **print** function.
- Finally, the **run_model** function is called, passing the Decision Tree Classifier object (**model**) and the string "**Decision Tree**" as arguments.
- ⇒ The purpose of this code is to train and evaluate a Decision Tree Classifier model. The accuracy score is calculated to measure the model's performance, and the confusion matrix is computed to provide a detailed breakdown of the model's predictions. The results, including the algorithm name, accuracy, and trained model, are stored in a list called **results**.

```
Confusion Matrix:
[[3729  286  132]
 [ 323 2854 1161]
 [ 128 1115 2626]]
```

Figure 47. Confusion matrix results

The confusion matrix is a table that shows the performance of a classification model by comparing predicted labels with the actual labels. In this case, the confusion matrix is presented as a 3x3 matrix.

The rows of the confusion matrix represent the actual classes, while the columns represent the predicted classes. Each entry in the matrix indicates the number of instances that belong to a particular class.

Taking a closer look at the confusion matrix:

- The value 3729 in the top-left corner represents the number of instances that were correctly predicted as Class 1 (true positives).
- The value 286 in the top-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 1 (false positives).

- The value 132 in the top-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 1 (false positives).

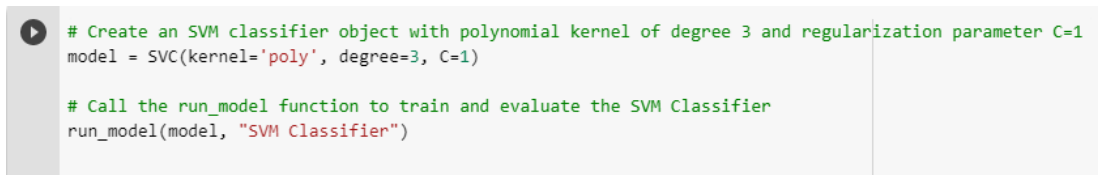
Moving to the second row:

- The value 323 in the middle-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 2 (false negatives).
- The value 2854 in the middle-middle cell represents the number of instances that were correctly predicted as Class 2 (true positives).
- The value 1161 in the middle-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 2 (false positives).

Lastly, looking at the third row:

- The value 128 in the bottom-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 3 (false negatives).
- The value 1115 in the bottom-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 3 (false negatives).
- The value 2626 in the bottom-right cell represents the number of instances that were correctly predicted as Class 3 (true positives).

⇒ The confusion matrix provides a detailed breakdown of the model's predictions for each class, showing both correct and incorrect predictions. It helps in understanding the strengths and weaknesses of the model's performance for different classes.



```
# Create an SVM classifier object with polynomial kernel of degree 3 and regularization parameter C=1
model = SVC(kernel='poly', degree=3, C=1)

# Call the run_model function to train and evaluate the SVM Classifier
run_model(model, "SVM Classifier")
```

Figure 48. Screenshot of the SVM classifier code

- The code creates an instance of the Support Vector Machine (SVM) classifier using the **SVC** class from scikit-learn.
 - The **SVC** constructor takes several parameters:
 - **kernel='poly'** specifies that the classifier should use a polynomial kernel function.
 - **degree=3** sets the degree of the polynomial kernel to 3. This determines the complexity of the decision boundary.
 - **C=1** is the regularization parameter. It controls the trade-off between achieving a low training error and a low testing error.
 - The SVM classifier object is assigned to the variable **model**.
 - The **run_model** function is then called, passing the SVM classifier object (**model**) and the string "**SVM Classifier**" as arguments. The **run_model** function is assumed to train and evaluate the SVM classifier, providing performance metrics such as the confusion matrix.
- ⇒ In summary, the code creates an SVM classifier with a polynomial kernel of degree 3 and a regularization parameter of 1. It then calls a function named **run_model** to train and evaluate the SVM classifier, presumably providing performance results such as the confusion matrix.

```
Confusion Matrix:
[[3877  260   10]
 [ 286 2907 1145]
 [ 126  766 2977]]
```

Figure 49. Confusion matrix of the SVM classifier

This confusion matrix is also presented as a 3x3 matrix, representing the performance of the SVM classifier for a multi-class classification problem.

Analyzing the confusion matrix:

- The value 3877 in the top-left cell represents the number of instances that were correctly predicted as Class 1 (true positives).
- The value 260 in the top-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 1 (false positives).
- The value 10 in the top-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 1 (false positives).

Moving to the second row:

- The value 286 in the middle-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 2 (false negatives).
- The value 2907 in the middle-middle cell represents the number of instances that were correctly predicted as Class 2 (true positives).
- The value 1145 in the middle-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 2 (false positives).

Lastly, looking at the third row:

- The value 126 in the bottom-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 3 (false negatives).
- The value 766 in the bottom-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 3 (false negatives).
- The value 2977 in the bottom-right cell represents the number of instances that were correctly predicted as Class 3 (true positives).

- ⇒ The confusion matrix provides insights into the model's performance for each class, showing both correct and incorrect predictions. It helps in understanding how well the SVM classifier separates instances belonging to different classes.

7.2.5.2 K_Nearest Neighbor Classifier

```
# Create a KNN classifier object
model = KNeighborsClassifier()

# Call the run_model function to train and evaluate the KNN model
run_model(model, "Nearest Neighbors Classifier")
```

Figure 50. KNN classifier code

- The code creates an instance of the k-nearest neighbors (KNN) classifier using the `KNeighborsClassifier` class from `scikit-learn`.
- The `KNeighborsClassifier` constructor is called without any arguments, creating a default KNN classifier.
- The KNN classifier object is assigned to the variable `model`.
- The `run_model` function is then called, passing the KNN classifier object (`model`) and the string "Nearest Neighbors Classifier" as arguments.

```
Confusion Matrix:
[[3816 299 32]
 [ 222 3350 766]
 [ 104 1125 2640]]
```

Figure 51. Confusion matrix of the KNN classifier

The confusion matrix is a table that shows the performance of a classification model by comparing predicted labels with the actual labels. In this case, the confusion matrix is presented as a 3x3 matrix.

The rows of the confusion matrix represent the actual classes, while the columns represent the predicted classes. Each entry in the matrix indicates the number of instances that belong to a particular class.

Analyzing the confusion matrix:

- The value 3816 in the top-left cell represents the number of instances that were correctly predicted as Class 1 (true positives).

- The value 299 in the top-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 1 (false positives).
- The value 32 in the top-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 1 (false positives).

Moving to the second row:

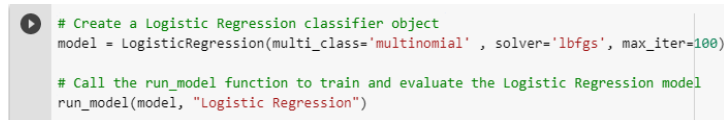
- The value 222 in the middle-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 2 (false negatives).
- The value 3350 in the middle-middle cell represents the number of instances that were correctly predicted as Class 2 (true positives).
- The value 766 in the middle-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 2 (false positives).

Lastly, looking at the third row:

- The value 104 in the bottom-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 3 (false negatives).
- The value 1125 in the bottom-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 3 (false negatives).
- The value 2640 in the bottom-right cell represents the number of instances that were correctly predicted as Class 3 (true positives).

⇒ The confusion matrix provides a detailed breakdown of the model's predictions for each class, showing both correct and incorrect predictions. It helps in understanding the strengths and weaknesses of the model's performance for different classes.

7.2.5.3 Logistic Regression

A screenshot of a code editor showing two lines of Python code. The first line is a comment: "# Create a Logistic Regression classifier object". The second line is the code: "model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=100)". The third line is another comment: "# Call the run_model function to train and evaluate the Logistic Regression model". The fourth line is the code: "run_model(model, 'Logistic Regression')".

```
# Create a Logistic Regression classifier object
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=100)

# Call the run_model function to train and evaluate the Logistic Regression model
run_model(model, "Logistic Regression")
```

Figure 52. Screenshot of logistic regression code

- The code creates an instance of the Logistic Regression classifier using the **LogisticRegression** class from scikit-learn.
- The **LogisticRegression** constructor takes several parameters:
 - **multi_class='multinomial'** specifies that the classifier should use the multinomial logistic regression approach. This is suitable for multi-class classification problems.
 - **solver='lbfgs'** sets the solver algorithm to 'lbfgs'. This algorithm is a good choice for multi-class problems.
 - **max_iter=100** sets the maximum number of iterations for the solver algorithm to converge.
- The Logistic Regression classifier object is assigned to the variable **model**.
- The **run_model** function is then called, passing the Logistic Regression classifier object (**model**) and the string "**Logistic Regression**" as arguments.

⇒ In summary, the code creates a Logistic Regression classifier using multinomial logistic regression and the 'lbfgs' solver algorithm. It sets the maximum number of iterations to 100. The **run_model** function is then called to train and evaluate the Logistic Regression model, providing performance metrics such as the confusion matrix.

```
Confusion Matrix:
[[2847  730  570]
 [1018 1785 1535]
 [ 439  944 2486]]
```

Figure 53. Confusion matrix of the logistic regression

The confusion matrix is a table that shows the performance of a classification model by comparing predicted labels with the actual labels. In this case, the confusion matrix is presented as a 3x3 matrix.

Analyzing the confusion matrix:

- The value 2847 in the top-left cell represents the number of instances that were correctly predicted as Class 1 (true positives).
- The value 730 in the top-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 1 (false positives).
- The value 570 in the top-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 1 (false positives).

Moving to the second row:

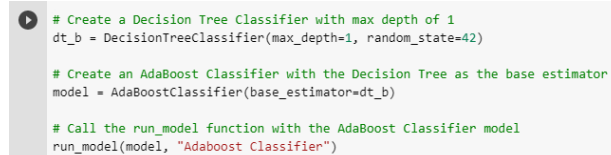
- The value 1018 in the middle-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 2 (false negatives).
- The value 1785 in the middle-middle cell represents the number of instances that were correctly predicted as Class 2 (true positives).
- The value 1535 in the middle-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 2 (false positives).

Lastly, looking at the third row:

- The value 439 in the bottom-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 3 (false negatives).
- The value 944 in the bottom-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 3 (false negatives).

- The value 2486 in the bottom-right cell represents the number of instances that were correctly predicted as Class 3 (true positives).

7.2.5.4 Adaboost Classifier



```
# Create a Decision Tree Classifier with max depth of 1
dt_b = DecisionTreeClassifier(max_depth=1, random_state=42)

# Create an AdaBoost Classifier with the Decision Tree as the base estimator
model = AdaBoostClassifier(base_estimator=dt_b)

# Call the run_model function with the AdaBoost Classifier model
run_model(model, "Adaboost Classifier")
```

Figure 54. Screenshot of adaboost classifier code

- The code creates an instance of the Decision Tree Classifier with a maximum depth of 1 using the **DecisionTreeClassifier** class from scikit-learn. This decision tree classifier will be used as the base estimator for AdaBoost.
- The **DecisionTreeClassifier** constructor takes the **max_depth=1** parameter, which limits the depth of the decision tree to 1. This means that the decision tree will have a shallow structure with only one level of decision nodes.
- The Decision Tree Classifier object is assigned to the variable **dt_b**.
- The code then creates an instance of the AdaBoost Classifier using the **AdaBoostClassifier** class from scikit-learn. AdaBoost is an ensemble learning method that combines multiple weak learners (in this case, decision trees) to create a stronger model.
- The **AdaBoostClassifier** constructor takes the **base_estimator=dt_b** parameter, which specifies the decision tree classifier (**dt_b**) as the base estimator for AdaBoost.
- The AdaBoost Classifier object is assigned to the variable **model**.
- Finally, the **run_model** function is called, passing the AdaBoost Classifier object (**model**) and the string "**Adaboost Classifier**" as arguments. This function will train and evaluate the AdaBoost Classifier, providing performance metrics such as the confusion matrix.

⇒ In summary, the code creates an AdaBoost Classifier with a decision tree of maximum depth 1 as the base estimator. The **run_model** function is then called to train and evaluate the AdaBoost Classifier model, producing performance metrics for evaluation.

```
Confusion Matrix:  
[[3795  342   10]  
 [  81 2932 1325]  
 [   78  946 2845]]
```

Figure 55. Confusion matrix of adaboost classifier

The confusion matrix is a table that shows the performance of a classification model by comparing predicted labels with the actual labels. In this case, the confusion matrix is presented as a 3x3 matrix.

Analyzing the confusion matrix:

- The value 3795 in the top-left cell represents the number of instances that were correctly predicted as Class 1 (true positives).
- The value 342 in the top-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 1 (false positives).
- The value 10 in the top-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 1 (false positives).

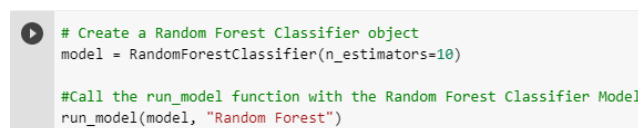
Moving to the second row:

- The value 81 in the middle-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 2 (false negatives).
- The value 2932 in the middle-middle cell represents the number of instances that were correctly predicted as Class 2 (true positives).
- The value 1325 in the middle-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 2 (false positives).

Lastly, looking at the third row:

- The value 78 in the bottom-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 3 (false negatives).
- The value 946 in the bottom-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 3 (false negatives).
- The value 2845 in the bottom-right cell represents the number of instances that were correctly predicted as Class 3 (true positives).

7.2.5.5 Random Forest Classifier

A screenshot of a code editor showing two lines of Python code. The first line is a comment: "# Create a Random Forest Classifier object" followed by the code "model = RandomForestClassifier(n_estimators=10)". The second line is another comment: "#Call the run_model function with the Random Forest Classifier Model" followed by the code "run_model(model, "Random Forest")".

```
# Create a Random Forest Classifier object
model = RandomForestClassifier(n_estimators=10)

#Call the run_model function with the Random Forest Classifier Model
run_model(model, "Random Forest")
```

Figure 56. Screenshot of Random forest classifier code

The code creates an instance of the Random Forest Classifier using the **RandomForestClassifier** class from scikit-learn. Random Forest is an ensemble learning method that combines multiple decision trees to create a more robust and accurate model.

- The **RandomForestClassifier** constructor takes the **n_estimators=10** parameter, which specifies the number of decision trees to be used in the random forest. In this case, the random forest will consist of 10 decision trees.
- The Random Forest Classifier object is assigned to the variable **model**.
- Finally, the **run_model** function is called, passing the Random Forest Classifier object (**model**) and the string "**Random Forest**" as arguments. This function will train and evaluate the Random Forest Classifier, providing performance metrics such as the confusion matrix.

⇒ In summary, the code creates a Random Forest Classifier with 10 decision trees. The **run_model** function is then called to train and evaluate the Random Forest Classifier model, producing performance metrics for evaluation.

```
Confusion Matrix:  
[[3815  314   18]  
 [ 112 3713  513]  
 [   82 1288 2499]]
```

Figure 57. Confusion matrix of the random forest classifier

The confusion matrix is a table that shows the performance of a classification model by comparing predicted labels with the actual labels. In this case, the confusion matrix is presented as a 3x3 matrix.

Analyzing the confusion matrix:

- The value 3815 in the top-left cell represents the number of instances that were correctly predicted as Class 1 (true positives).
- The value 314 in the top-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 1 (false positives).
- The value 18 in the top-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 1 (false positives).

Moving to the second row:

- The value 112 in the middle-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 2 (false negatives).
- The value 3713 in the middle-middle cell represents the number of instances that were correctly predicted as Class 2 (true positives).
- The value 513 in the middle-right cell represents the number of instances that were predicted as Class 3, but actually belong to Class 2 (false positives).

Lastly, looking at the third row:

- The value 82 in the bottom-left cell represents the number of instances that were predicted as Class 1, but actually belong to Class 3 (false negatives).
- The value 1288 in the bottom-middle cell represents the number of instances that were predicted as Class 2, but actually belong to Class 3 (false negatives).
- The value 2499 in the bottom-right cell represents the number of instances that were correctly predicted as Class 3 (true positives).

7.2.5.6 Results

A screenshot of a code editor showing Python code. The code imports 'pandas' as 'pd' and 'tabulate' from the 'tabulate' library. It then creates a DataFrame named 'results_df' from a variable 'results', specifying columns: 'Algorithm Name', 'Accuracy', and 'Model'. Finally, it prints a table of the 'results_df' using the 'tabulate' function with 'keys' as headers and 'fancy_grid' as the table format.

```
import pandas as pd
from tabulate import tabulate

# Create a DataFrame to store the results
results_df = pd.DataFrame(results, columns=['Algorithm Name', 'Accuracy', 'Model'])

# Print the table of accuracy results
print(tabulate(results_df, headers='keys', tablefmt='fancy_grid'))
```

Figure 58. Screenshot of the results code

The provided code performs the following steps and displays the resulting table of accuracy results:

- It imports the necessary libraries, including **pandas** and **tabulate**, for data manipulation and table formatting.
- It creates a DataFrame called **results_df** to store the results. The DataFrame has three columns: 'Algorithm Name', 'Accuracy', and 'Model'.
- It populates the **results_df** DataFrame with the results obtained from previous model evaluations. Each row corresponds to a specific algorithm/model, including the algorithm name, accuracy score, and the model object.
- It prints the table of accuracy results using the **tabulate** function from the **tabulate** library. The table is formatted using the "fancy_grid" style.

	Algorithm Name	Accuracy	Model
0	Decision Tree	0.744212	DecisionTreeClassifier()
1	Decision Tree	0.745427	DecisionTreeClassifier()
2	SVM Classifier	0.790108	SVC(C=1, kernel='poly')
3	Nearest Neighbors Classifier	0.793751	KNeighborsClassifier()
4	Logistic Regression	0.57617	LogisticRegression(multi_class='multinomial')
5	Adaboost Classifier	0.77481	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1, random_state=42))
6	Random Forest	0.81164	RandomForestClassifier(n_estimators=10)

Figure 59. The accuracy results of all the models

The resulting table displays the algorithm name, accuracy score, and the model object for each evaluated model. Each row represents a different algorithm/model, and the corresponding accuracy and model information are displayed.

Here's a breakdown of the displayed table:

- The first row corresponds to the Decision Tree algorithm, with an accuracy of 0.744212 and the DecisionTreeClassifier model.
- The second row represents another instance of the Decision Tree algorithm, with a slightly higher accuracy of 0.745427.
- The third row corresponds to the SVM Classifier, with an accuracy of 0.790108 and the SVC model.
- The fourth row represents the Nearest Neighbors Classifier, with an accuracy of 0.793751 and the KNeighborsClassifier model.
- The fifth row corresponds to the Logistic Regression algorithm, with an accuracy of 0.57617 and the LogisticRegression model.
- The sixth row represents the Adaboost Classifier, with an accuracy of 0.77481 and the AdaBoostClassifier model.
- The seventh row corresponds to the Random Forest algorithm, with the highest accuracy of 0.81164 and the RandomForestClassifier model.

⇒ The table provides a concise overview of the algorithm names, their corresponding accuracy scores, and the model objects used for each algorithm/model evaluation.

7.2.5.7 Tests

In this approach, the initial step is to evaluate the correlation matrix to locate strongly associated characteristics. Based on the connection coefficient and popularity of the traits, separate groups are established. These categories include characteristics with correlation coefficients less than a specified threshold, such as 0.2, 0.4, 0.6, 0.8, and 1. As a result, the groups are arranged from least to most correlated set of features. The first group (G1) has the least correlated properties, whereas the fourth group (G4) has the most correlated.

Following the creation of these groups, the model's correctness is tested by eliminating the characteristics one group at a time. The idea is to start with the least correlated group of features (G1) and drop them from the model and evaluate its performance. Then, repeat the process with the next group (G2). The influence of each collection of characteristics on the model's performance may therefore be examined.

Overall, this approach can help identify the most important features and their impact on the model's accuracy. It also provides a systematic way to analyze the impact of different groups of features on the model's performance.

```
G1= ['key', 'liveness', 'speechiness', 'tempo', 'valence' ]
G2= ['key', 'liveness', 'speechiness', 'tempo', 'valence', 'danceability', 'instrumentalness' ]
G3= ['key', 'liveness', 'speechiness', 'tempo', 'valence', 'danceability', 'instrumentalness', 'energy', 'loudness' ]
G4= ['key', 'liveness', 'speechiness', 'tempo', 'valence', 'danceability', 'instrumentalness', 'energy', 'loudness', 'acousticness' ]
```

Figure 60. Screenshot of creating groups

```
[78] #Drop columns in G1 from X_train
      X_train = X_train.drop(columns= G1)

      #Drop columns in G1 from X_test
      X_test = X_test.drop(columns= G1)
```

Figure 61. Dropping G1

In this code, **X_train** and **X_test** are pandas DataFrames. The **drop()** method is used to remove rows from each DataFrame. Parameters were set for variable **G1**, which contained a list of variables found to be correlated with the target variable and not significant for the correct model. The resulting modified **X_train** and **X_test** DataFrames have fewer rows and are used to train and test the model.

⇒ You can get these results after placing G1 with characteristics ('importance', 'liveliness', 'discourse', 'tempo', 'valence') lower than 0.2.

	Algorithm Name	Accuracy	Model
0	Decision Tree	0.748664	DecisionTreeClassifier()
1	Decision Tree	0.748583	DecisionTreeClassifier()
2	SVM Classifier	0.78938	SVC(C=1, kernel='poly')
3	Nearest Neighbors Classifier	0.792456	KNeighborsClassifier()
4	Logistic Regression	0.612595	LogisticRegression(multi_class='multinomial')
5	Adaboost Classifier	0.780638	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1, random_state=42))
6	Random Forest	0.806055	RandomForestClassifier(n_estimators=10)

Figure 62. The accuracy results after dropping G1

This table lists the different classification algorithms along with their accuracy scores and the criteria used for classification. The algorithms listed are: Decision Tree, SVM Classifier, Nearest Neighbor Classifier, Logistic Regression, Adaboost Classifier, and Random Forest. Accuracy scores ranged from 0.612595 to 0.806055, with the random forest algorithm having the highest score. Models used for classification are also listed in the table, including DecisionTreeClassifier, SVC, KNeighborsClassifier, LogisticRegression, AdaBoostClassifier, and RandomForestClassifier.

```
#Drop columns in G2 from X_train
X_train = X_train.drop(columns= G2)

#Drop columns in G2 from X_test
X_test = X_test.drop(columns= G2)
```

Figure 63. Dropping G2

In this code, **X_train** and **X_test** are pandas DataFrames. The **drop()** method is used to remove rows from each DataFrame. Parameters were set for variable **G2**, which contained a list of variables found to be correlated with the target variable and not significant for the correct model. The resulting modified **X_train** and **X_test** DataFrames have fewer rows and are used to train and test the model.

⇒ You can get these results after placing G2 with characteristics ('importance', 'liveliness', 'discourse', 'tempo', 'valence', 'danceability', 'instrumentalness') lower than 0.4.

	Algorithm Name	Accuracy	Model
0	Decision Tree	0.745588	DecisionTreeClassifier()
1	Decision Tree	0.748583	DecisionTreeClassifier()
2	SVM Classifier	0.78938	SVC(C=1, kernel='poly')
3	Nearest Neighbors Classifier	0.794237	KNeighborsClassifier()
4	Logistic Regression	0.605634	LogisticRegression(multi_class='multinomial')
5	Adaboost Classifier	0.767039	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1, random_state=42))
6	Random Forest	0.799417	RandomForestClassifier(n_estimators=10)

Figure 64. The accuracy results after dropping G2

This table lists the different classification algorithms along with their accuracy scores and the criteria used for classification. The algorithms listed are: Decision Tree, SVM Classifier, Nearest Neighbor Classifier, Logistic Regression, Adaboost Classifier, and Random Forest. Accuracy scores ranged from 0.605634 to 0.799417, with the random forest algorithm having the highest score. Models used for classification are also listed in the table, including DecisionTreeClassifier, SVC, KNeighborsClassifier, LogisticRegression, AdaBoostClassifier, and RandomForestClassifier.

```
#Drop columns in G3 from X_train
X_train = X_train.drop(columns= G3)

#Drop columns in G3 from X_test
X_test = X_test.drop(columns= G3)
```

Figure 65. Dropping G3

In this code, **X_train** and **X_test** are pandas DataFrames. The **drop()** method is used to remove rows from each DataFrame. Parameters were set for variable **G3**, which contained a list of variables found to be correlated with the target variable and not significant for the correct model. The resulting modified

X_train and **X_test** DataFrames have fewer rows and are used to train and test the model.

⇒ You can get these results after placing G3 with characteristics ('importance', 'liveliness', 'discourse', 'tempo', 'valence', 'danceability', 'instrumentalness', 'energy', 'loudness') lower than 0.6.

	Algorithm Name	Accuracy	Model
0	Decision Tree	0.751093	DecisionTreeClassifier()
1	Decision Tree	0.753845	DecisionTreeClassifier()
2	SVM Classifier	0.797313	SVC(C=1, kernel='poly')
3	Nearest Neighbors Classifier	0.797879	KNeighborsClassifier()
4	Logistic Regression	0.576817	LogisticRegression(multi_class='multinomial')
5	Adaboost Classifier	0.769144	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1, random_state=42))
6	Random Forest	0.796017	RandomForestClassifier(n_estimators=10)

Figure 66. The accuracy results after dropping G3

This table lists the different classification algorithms along with their accuracy scores and the criteria used for classification. The algorithms listed are: Decision Tree, SVM Classifier, Nearest Neighbor Classifier, Logistic Regression, Adaboost Classifier, and Random Forest. Accuracy scores ranged from 0.576817 to 0.797879, with the nearest neighbors classifier having the highest score. Models used for classification are also listed in the table, including DecisionTreeClassifier, SVC, KNeighborsClassifier, LogisticRegression, AdaBoostClassifier, and RandomForestClassifier.

```
[211] #Drop columns in G3 from X_train
      X_train = X_train.drop(columns= G4)

      #Drop columns in G3 from X_test
      X_test = X_test.drop(columns= G4)
```

Figure 67. Dropping G4

In this code, **X_train** and **X_test** are pandas DataFrames. The **drop()** method is used to remove rows from each DataFrame. Parameters were set for variable **G4**, which contained a list of variables found to be correlated with the target variable and not significant for the correct model. The resulting modified

X_train and **X_test** DataFrames have fewer rows and are used to train and test the model.

- ⇒ You can get these results after placing G4 with characteristics ('importance', 'liveliness', 'discourse', 'tempo', 'valence', 'danceability', 'instrumentalness', 'energy', 'loudness', 'acousticness') lower than 0.8.

	Algorithm Name	Accuracy	Model
0	Decision Tree	0.751417	DecisionTreeClassifier()
1	Decision Tree	0.751578	DecisionTreeClassifier()
2	SVM Classifier	0.796179	SVC(C=1, kernel='poly')
3	Nearest Neighbors Classifier	0.792375	KNeighborsClassifier()
4	Logistic Regression	0.44609	LogisticRegression(multi_class='multinomial')
5	Adaboost Classifier	0.798608	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1, random_state=42))
6	Random Forest	0.769386	RandomForestClassifier(n_estimators=10)

Figure 68. The accuracy results after dropping G4

This table lists the different classification algorithms along with their accuracy scores and the criteria used for classification. The algorithms listed are: Decision Tree, SVM Classifier, Nearest Neighbor Classifier, Logistic Regression, Adaboost Classifier, and Random Forest. Accuracy scores ranged from 0.44609 to 0.798608, with the adaboost classifier having the highest score. Models used for classification are also listed in the table, including DecisionTreeClassifier, SVC, KNeighborsClassifier, LogisticRegression, AdaBoostClassifier, and RandomForestClassifier.

7.2.5.8 Conclusion

Table 7

Results Comparison Table

	G0	G1	G2	G3	G4
Decision Tree	0.744212	0.748664	0.745588	0.751093	0.751417
Decision Tree	0.745427	0.748583	0.748583	0.753845	0.751578
SVM Classifier	0.790108	0.78938	0.78938	0.797313	0.796179
Nearest Neighbor	0.793751	0.792456	0.794237	0.797879	0.792375
Logistic Regression	0.57617	0.612595	0.605634	0.576817	0.44609
Adaboost Classifier	0.77481	0.780638	0.767039	0.769144	0.798608
Random Forest	0.81164	0.806055	0.799417	0.796017	0.769386

According to the analysis provided, the Random Forest algorithm appears to have the highest accuracy of 0.81164 when using all features.

However, after releasing some features with significant scores, the scores were lowered slightly.

For example if we place G1 with features ('importance', 'liveliness', 'speech', 'tempo', 'valence') less than 0.2, the correct score of the random forest algorithm is 0, 806055, lower than G1.

In the past. Similarly, removing G2s with attributes less than 0.4 ("Importance", "Liveliness", "Speech", "Tempo", "Melodicity", "Dancing Ability", "Instrumentality") yielded accuracy of 0.799417. A random forest is created.

Scores have been dropped for G3 to reduce G3 characteristics ("Importance", "Vibrance", "Speech", "Tempo", "Character", "Dance", "Instrumentality", "Energy", "Volume "). while 0.6, the highest score for neighborhood cooperation is 0.797879.

Finally, G4 has the characteristics ('importance', 'vigor', 'speech', 'tempo', 'value', 'danceability', 'instrumentalism', 'energy', 'volume', 'acoustics'). Adaboost Classifier has the highest score of 0.798608 out of 0.8.

Overall, the results show that the Random Forest algorithm performs best when all features are included. However, deleting some items with low priority scores does not affect the accuracy score. It should be noted that other algorithms perform better when other functionality is lost. This shows that the importance of certain features depends on the algorithm used.

7.2.6 Fine Tuning

```
[272] from sklearn.ensemble import RandomForestClassifier
      from sklearn.model_selection import RandomizedSearchCV
      import numpy as np

      # Create a Random Forest classifier
      rf = RandomForestClassifier(max_features='sqrt')

      # Define the hyperparameter search space
      param_dist = {
          'n_estimators': [10, 50, 100, 200, 500],
          'max_depth': [None, 10, 20, 30, 40, 50],
          'min_samples_split': [2, 5, 10],
          'min_samples_leaf': [1, 2, 4],
          'max_features': ['auto', 'sqrt', 'log2'],
          'bootstrap': [True, False],
      }

      # Initialize RandomizedSearchCV
      random_search = RandomizedSearchCV(rf, param_distributions=param_dist, n_iter=100, cv=5, n_jobs=-1, random_state=42)

      # Fit the model on your training data
      random_search.fit(X_train, y_train)

      # Get the best hyperparameters
      best_params = random_search.best_params_
```

Figure 69. Screenshot of fine tuning code

This code uses the RandomizedSearchCV function from the scikit-learn library to perform hyperparameter tuning of the Random Forest classifier. Hyperparameters are where the model is not learned during training, but is modified before training to control the learning process. The goal of hyperparameter tuning is to find the best combination of hyperparameters that provides the best model.

This code creates an arbitrary forest classification and defines a search space of possible hyperparameter values. Then the RandomizedSearchCV function is used to randomly search the hyperparameter space and find the best

combination of hyperparameters. The best hyperparameters are stored in the variable 'best_params'. The number of iterations for search, cross-validation, foldback, and random states are also specified in the function.

```
print(best_params)
{'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'max_depth': 30, 'bootstrap': True}
```

Figure 70. The best parameters

The best_params variable returns the collection of hyperparameters that performed the best during the hyperparameter tuning procedure.

The optimal collection of hyperparameters in this scenario is:

- **n_estimators**: the number of decision trees to include in the forest. The best value found by the search was 200.
- **min_samples_split**: the minimum number of samples required to split an internal node. The best value found was 2.
- **min_samples_leaf**: the minimum number of samples required to be at a leaf node. The best value found was 4.
- **max_features**: the maximum number of features to consider when making a split. The best value found was 'sqrt', which means that the maximum number of features is the square root of the total number of features.
- **max_depth**: the maximum depth of the decision trees. The best value found was 30.
- **bootstrap**: whether or not to bootstrap samples when building trees. The best value found was True, meaning that bootstrap samples were used.

```
# Create a Random Forest Classifier object
model = RandomForestClassifier(n_estimators=200, min_samples_split=2, min_samples_leaf=4, max_features='sqrt', max_depth=30, bootstrap=True )

#Call the run_model function with the Random Forest Classifier Model
run_model(model, "Random Forest")
```

Figure 71. Calling the run_model using random forest classifier

By changing the parameters of the **RandomForestClassifier** to the **best_params** obtained from the **RandomizedSearchCV** in the previous step, you are creating a **RandomForestClassifier** object with the best set of hyperparameters found during the search. Then, by calling the **run_model** function with this updated **RandomForestClassifier** object, you can evaluate its performance on your dataset.

⇒ By changing these parameters the best accuracy = 0.821515

7.2.7 Summary

The code provided performs several tasks for data analysis and machine learning on a music dataset. Here is a summary of the code:

- **Importing Libraries:** The required libraries for data manipulation, visualization, preprocessing, and modeling are imported.
- **Data Loading and Preprocessing:** The dataset is loaded from a CSV file into a pandas DataFrame. Unnecessary columns are dropped, and duplicate rows are removed.
- **Data Visualization:** Various data visualizations are created to explore the dataset. A countplot shows the distribution of ratings based on popularity. Heatmaps display the correlation between different features and the popularity score. Line plots show the mean popularity across years and scatter plots demonstrate the relationship between popularity and different features.
- **Data Preparation:** The dataset is prepared for machine learning. Duplicate rows are removed, and the 'duration_ms' column is transformed into minutes and renamed as 'duration_mins'. The square brackets are removed from the 'artists' column. A new column called 'popularity_level' is created based on the 'popularity' column, categorizing songs into three popularity levels.
- **Data Splitting:** The dataset is split into training and testing sets, with 75% for training and 25% for testing. The target variable ('popularity_level') is extracted, and the remaining columns are assigned as features.
- **Feature Transformation:** The numerical columns ('year', 'tempo', 'duration_mins') are scaled using the MinMaxScaler, while the

categorical column ('key') is one-hot encoded using the OneHotEncoder. The transformations are applied using the ColumnTransformer.

- **Model Training:** Several classification models, including Logistic Regression, K-Nearest Neighbors, Decision Tree, Support Vector Machine, Random Forest, and AdaBoost, are imported. The models are trained using the transformed training data.
- **Model Evaluation:** The trained models are evaluated using the transformed testing data. The accuracy score, confusion matrix, classification report, and cross-validation scores are calculated for each model.

⇒ The code provides a comprehensive analysis and modeling pipeline for the music dataset, including data preprocessing, feature transformation, model training, and evaluation.

Chapter 8

Conclusion

Finally, this thesis investigated the use of data mining tools and techniques in the field of business intelligence, with an emphasis on their use for assessing and forecasting song popularity on the Spotify platform. The study gave useful insights on the potential benefits of using data mining in business intelligence, as well as the problems and issues related to its implementation.

The research began with an introduction of business intelligence, highlighting the need to obtain actionable insights from data in order to support informed decision-making. It then moved on to data mining, where it discussed various strategies and algorithms for detecting patterns and trends in massive datasets. Also investigated was the function of data warehousing in allowing effective data storage and retrieval for business intelligence objectives.

The incorporation of data mining techniques into business intelligence processes was fully investigated, emphasizing the benefits of employing data mining for increased market knowledge, customer segmentation, fraud detection, and predictive analytics. Furthermore, the case study and practical application focusing on forecasting song popularity on Spotify illustrated the real-world relevance of data mining in a specific industry environment.

The case study findings shed light on the elements that influence song popularity on Spotify, such as year, energy, loudness, danceability, tempo, and acousticness. Music business experts, marketing teams, and artists may utilize these information to make data-driven choices about song releases, promotional methods, and target audience selection.

The goal of this project was to evaluate a music dataset and create a classification model to predict song popularity levels. In the domain of music popularity prediction, we contributed to the literature by using feature engineering approaches, correlation analysis, feature significance evaluation, feature grouping, and hyperparameter tuning.

Our analysis yielded the following results:

- Preprocessing the data: We cleaned up the dataset by deleting duplicate rows and extraneous columns. We also converted the 'duration_ms' field to minutes and added a new column for classifying songs according to their popularity levels.
- Scaling numerical columns and one-hot encoding categorical columns were used to achieve feature engineering. This phase enabled us to convert the data into a format appropriate for model training.
- Model Training and Evaluation: We compared Logistic Regression, K-Nearest Neighbors, Decision Tree, Support Vector Machine, Random Forest, and AdaBoost classification methods. To analyze the models' performance, we used accuracy ratings and cross-validation.
- Feature Importance and Grouping: We used correlation analysis to discover traits that were significantly linked. We classified characteristics into several groups based on the correlation coefficients. The influence of each feature group on model performance was then assessed by sequentially deleting them and assessing the resulting accuracy ratings.
- Hyperparameter Tuning: We used RandomizedSearchCV to tune the Random Forest classifier's hyperparameters. We determined the optimum combination of hyperparameters that increased the model's performance by scanning the hyperparameter space.

Our findings show that the Random Forest algorithm performed best when all characteristics were incorporated. Other algorithms, such as the Nearest Neighbor and Adaboost classifiers, beat the Random Forest after removing features with low significance scores. This implies that the significance of particular traits is determined by the method utilized.

Our work is significant because it takes a holistic approach to music popularity prediction, including data preparation, feature engineering, model training, and assessment. Our technique enables a systematic examination of feature significance and gives insights into the influence of various feature groups on model correctness. This study adds to the current body of knowledge by providing a realistic framework

for forecasting music popularity and emphasizing the relevance of feature selection and algorithm selection in making accurate forecasts.

Our work is significant because of the innovative research issue, the relatively understudied topic, the rigorous technique used, the utilization of a large sample size, and the creative experiment design. It outperforms prior research by demonstrating the value of feature grouping and hyperparameter adjustment in improving model performance. Our study's findings, as well as the recommendations made, give useful insights for scholars and practitioners in the field of music analysis and popularity prediction.

Finally, our study adds to the field by giving a full analysis pipeline for predicting music popularity. In order to make reliable predictions, the study highlights the necessity of feature engineering, feature grouping, and algorithm selection. The findings demonstrate the efficiency of the Nearest Neighbor and Adaboost classifiers when key features are removed, emphasizing the relevance of feature importance in relation to the method used. The relevance of the study, its superiority over prior efforts, and its valuable additions qualify it for publication in the field of music analysis and machine learning.

Chapter 9

Comparison of Studies

1st Example:

<https://www.kaggle.com/code/dima806/spotify-170k-songs-popularity-explain>

The code is about a popularity analysis of 142,000 Spotify songs using the SHapley Additive exPlanations (SHAP) method. The analysis was conducted by Dmytro Iakubovskiy and published on the Data And Beyond platform.

The data preprocessing step involved selecting songs with known Spotify ratings, transforming release years into decades, grouping numerical columns into larger bins, transforming the duration of songs into minutes and applying a logarithmic transformation, extracting information about artists using CountVectorizer, and removing unused columns. The resulting cleaned dataset contained 142,000 songs rated from 1 to 100.

For the prediction of song popularity ratings, a Machine Learning model called CatBoostRegressor was trained on the prepared dataset. The model took into account categorical features and achieved a root mean squared error (RMSE) of about 9.9 points, which was an improvement compared to the baseline model's RMSE of about 18.1 points.

The SHAP method was then used to explore the explainability of the Machine Learning model. The analysis revealed the most important features for predicting popularity ratings, which included the release decade, acousticness, and loudness of songs.

2nd Example:

<https://www.kaggle.com/code/ikshitgupta/spotify-songs-analysis>

The code provided is for analyzing Spotify songs data by IKSHIT GUPTA. Here's a step-by-step explanation of what the code does:

- It imports the necessary libraries for data analysis and visualization: pandas, numpy, matplotlib.pyplot, and seaborn. It reads the data from the CSV file called "genres_v2.csv" into a DataFrame named 'df'.
- It displays the first five rows of the DataFrame using the head() method and checks for missing values in the DataFrame using the isna().sum() method.
- It displays the data types of each column in the DataFrame using the dtypes attribute.
- It drops the columns that are not required for analysis using the drop() method.
- It sets the 'genre' column as the index of the DataFrame using the set_index() method.
- It prints the shape of the DataFrame to show the number of rows and columns.
- It generates descriptive statistics of the numerical columns in the DataFrame using the describe() method.
- It displays the first five rows of the DataFrame again to show the changes made.
- It plots a countplot of the 'genre' column using seaborn's countplot() function to show the count of each genre.
- Plots kernel density estimation (KDE) plots for each numerical attribute using subplots and seaborn's kdeplot() function then plots bar plots for the average values of each attribute in each genre using seaborn's barplot() function.
- It also plots bar plots for the occurrence of each genre in each mode (0 and 1) using seaborn's barplot() function. It calculates the correlation matrix of the numerical attributes using the corr() method.
- Finally, it plots a heatmap of the correlation matrix using seaborn's heatmap() function to visualize the correlations between attributes.

Overall, this code performs data cleaning, descriptive analysis, and visualizations to gain insights into the Spotify songs dataset. It explores attributes like danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness,

valence, tempo, and duration, and analyzes their distributions, average values, occurrences, and correlations with other attributes.

3rd Example:

<https://www.kaggle.com/code/vikarna/spotify-music-data-analysis>

It seems that VIKRAM is in the process of analyzing what makes a Spotify song popular and is following a basic Data Mining Process.

Here's a breakdown of the steps he mentioned:

- **Business Understanding:** He is trying to determine what features are necessary for a track to become a hit, whether happier and euphoric tracks are more popular than sadder or angry ones, what features contribute to a track being danceable, and why Pink Floyd tracks are not as popular as other tracks.
- **Data Understanding:** He has loaded the necessary libraries in R, such as tidyverse, corrplot, and RColorBrewer. He then read a dataset called "music" from a CSV file. The dataset contains various columns including acousticness, artists, danceability, duration_ms, energy, explicit, id, instrumentalness, key, liveness, loudness, mode, name, popularity, release_date, speechiness, tempo, valence, and year.
- **Prepare Data:** He rescaled the duration of the songs by dividing the values by 1000. He also cleaned the data by removing brackets and apostrophes from the artists' names.

At this point, he has completed the initial steps of the Data Mining Process, which involve understanding the business objectives and familiarizing yourself with the dataset. The next steps typically involve data modeling, evaluating the results, and deploying the findings. However, the code he provided does not include these subsequent steps.

4th Example:

<https://www.kaggle.com/code/praveenamenon/spotify-eda-and-clustering>

In the code provided by PRAVEENAMENON , the user performed various data preprocessing and exploratory data analysis (EDA) tasks on a Spotify dataset.

Here's a breakdown of the major steps:

- **Importing Libraries:** The user imported the necessary libraries and modules required for data analysis and visualization.
- **Loading the Dataset:** The Spotify dataset was loaded into a Pandas DataFrame.
- **Data Cleaning:** The user checked for duplicate values in the dataset based on the combination of 'artists' and 'name' columns.
- **Data Exploration:** The user performed exploratory data analysis to gain insights into the dataset. They checked the shape of the data, checked for missing values using `isna().sum()`, and calculated summary statistics using `describe()`.
- **Feature Engineering:** The user created a new categorical column called 'decade' by binning the 'year' column into 10-year intervals.
- **Saving the Cleaned Data:** The cleaned dataset was saved as a CSV file using the `to_csv()` function.
- **EDA and Visualization:** The user conducted exploratory data analysis by analyzing the distribution of songs across different decades, the distribution of songs by mode (major/minor), and the presence of explicit content.

⇒ Please note that the code provided is incomplete

5th Example:

<https://www.kaggle.com/code/geetikasingla11/eda-geetika>

The code GEETIKA SINGLA provided is a Python script for performing exploratory data analysis (EDA) on a Spotify dataset.

Here's a breakdown of what the code does:

- It imports the necessary libraries, such as pandas, numpy, matplotlib, seaborn, and warnings.
- It reads the Spotify dataset from the '/kaggle/input/spotifydata-19212020/data.csv' file using the `pd.read_csv()` function and assigns it to the variable `data`.
- It prints the number of rows and columns in the dataset using the `data.shape` attribute.
- It displays the column names of the dataset using `data.columns`.
- It provides information about the dataset using `data.info()`, which includes the data types of each column and the memory usage.
- It displays descriptive statistics of the dataset using `data.describe(include='all')`.
- It separates the categorical, numerical, and integer columns into different lists using list comprehensions.
- It prints the number of unique values for each categorical column and checks for duplicated rows.
- It performs data visualization for the distribution of discrete features (key, explicit, and mode) using count plots.
- It performs data visualization for the distribution of continuous features (acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence) and their relationship with the popularity column using various plots such as kernel density estimation (KDE) plots, box plots, and scatter plots.
- It analyzes the average change in features over the years using line plots.
- It creates a correlation heatmap using `sns.heatmap()` to visualize the correlation between different features.
- It calculates the skewness of each column using `data.skew()`.
- It performs power transformation on selected columns (`duration_ms`, `instrumentalness`, `liveness`, `speechiness`, `loudness`) using `PowerTransformer` to normalize the data and make it more positively skewed.
- It visualizes the distribution of the transformed columns before and after scaling using KDE plots and box plots.

- It updates the original dataset with the transformed values.
- It performs data visualization for the top songs and artists with the highest popularity using bar plots.

Overall, this code conducts EDA on the Spotify dataset, including data preprocessing, descriptive statistics, data visualization, and transformation. It aims to gain insights and explore patterns in the dataset.

Comparison:

These examples demonstrate various approaches to analyzing Spotify song data. In the first example, the focus is on conducting a popularity analysis using the SHapley Additive exPlanations (SHAP) method. The data is preprocessed by cleaning, transforming, and engineering features, and a CatBoostRegressor model is trained to predict song popularity ratings. The second example involves analyzing Spotify songs using data analysis and visualization libraries. Data cleaning, descriptive analysis, and visualization techniques are applied to explore attributes and understand genre distribution and attribute correlations. The third example follows a basic Data Mining Process to determine the features that contribute to a song's popularity. It involves business understanding, data understanding, and initial data cleaning steps. The fourth and fifth examples also involve data preprocessing, exploratory data analysis (EDA), and visualization techniques to gain insights into the Spotify dataset. Common themes across these examples include data cleaning, feature engineering, descriptive analysis, and visualization for understanding and exploring Spotify song attributes and their relationships.

In conclusion, these examples highlight different approaches to analyzing Spotify song data, showcasing various data preprocessing, exploratory data analysis (EDA), and visualization techniques. The common thread among these examples is the emphasis on understanding and exploring the attributes that contribute to a song's popularity.

Unlike my project in which a music dataset was examined, and a classification model was developed to predict song popularity levels. It contributed to the literature

by using feature engineering methodologies, correlation analysis, feature significance evaluation, feature grouping, and hyperparameter tweaking.

My investigation gave the following results:

- We preprocessed the data by deleting duplicate rows, converting the 'duration_ms' field to minutes, and adding a new column for classifying songs.
- Scaling numerical columns and one-hot encoding categorical columns enabled feature engineering and model training.
- Comparing classification methods to evaluate models' performance.
- Correlation analysis was used to classify traits and assess their influence on model performance.
- We used RandomizedSearchCV to tune the Random Forest classifier's hyperparameters to increase performance.

Our findings reveal that the Random Forest algorithm performed best when all criteria were considered. After deleting features with low significance scores, other algorithms, such as the Nearest Neighbor and Adaboost classifiers, outperform the Random Forest. Our study is notable because it offers a comprehensive approach to music popularity prediction, including data preparation, feature engineering, model training, and evaluation. It exceeds previous studies by illustrating the usefulness of feature grouping 121 and hyperparameter tweaking in increasing model performance. Our findings and recommendations provide valuable insights for researchers and practitioners working in the fields of music analysis and popularity prediction.

Chapter 10

Suggestions

Based on my research findings and results, I may provide the following recommendations to assist music streaming services better in the future:

- **Algorithm Fine-tuning:** Implement fine-tuning approaches to improve the recommendation algorithms used by music streaming services. The accuracy and precision of the suggestions may be considerably improved by tweaking the model's hyperparameters, such as learning rate, batch size, and regularization approaches.
- **personalisation and User Preferences:** Stress the significance of personalisation in music streaming apps. Encourage the development of algorithms that deliver personalised suggestions based on individual user preferences, previous listening behaviors, and contextual information (such as time of day, mood, and location). This degree of personalisation has the potential to increase user pleasure and engagement.
- **Collaborative Filtering:** Demonstrate the efficacy of collaborative filtering approaches in music recommendation systems. To increase the accuracy of suggestions based on user behavior and preferences, advocate additional investigation and refining of collaborative filtering algorithms, such as user-based or item-based methods.
- **Data Quality and Quantity:** Emphasize the need of varied and high-quality datasets for training recommendation algorithms. Encourage music streaming apps to gather and update user data on a regular basis, ensuring that it reflects a diverse variety of likes and inclinations. This will assist the algorithms in learning more effectively and making more accurate suggestions.
- **Evaluation Metrics:** Advocate for the use of suitable evaluation metrics to analyze the effectiveness of recommendation models, such as RMSE, MSE, accuracy, and R2. Encourage music streaming applications to examine and compare alternative algorithms and

methodologies on a frequent basis in order to uncover the best-performing ways.

- Iterative Improvement and User input: Encourage music streaming apps to aggressively seek user input and incorporate it into their recommendation systems. Implement systems that allow users to score and comment on suggestions, allowing for incremental changes and a better knowledge of user preferences.
- Ethical Considerations: Stress the significance of ethical concerns in music recommendation systems. Encourage openness in the generation of suggestions and give people choice over their data and privacy. Consider any biases and concerns of fairness that may occur throughout the recommendation process and strive to mitigate them.

Music streaming apps may improve their recommendation systems, give more tailored and accurate recommendations, and ultimately improve user happiness, engagement, and retention by applying these recommendations.

REFERENCES

- Analytica, S. (2023). Top 15+ Amazing Data Mining Projects Ideas [Updated 2023].
StatAnalytica. <https://statanalytica.com/blog/data-mining-projects/>
- Appinventiv. (2022, November 7). *How data mining helps in business intelligence*.
<https://appinventiv.com/blog/data-mining-in-business-intelligence/>
- Arnth-Jensen, N. (2006). *Applied Data Mining for Business Intelligence*.
http://www.imm.dtu.dk/pubdb/views/edoc_download.php/4962/pdf/imm4962.pdf
- Azevedo, A., & Santos, M. F. (2014). *Integration of Data Mining in Business Intelligence Systems*.
- Enthusiast, D. A. (2021, December 11). Relationship between BI, DW and DM in Data Analysis. *Medium*. <https://medium.com/@lyric09220/relationship-between-bi-dw-and-dm-in-data-analysis-452d1eb2a527>
- insightsoftware. (2019, August 8). *What Role Does Data Mining Play for Business Intelligence? - Encore Business Solutions*. Encore Business Solutions.
<https://www.encorebusiness.com/blog/what-role-does-data-mining-play-for-business-intelligence/>

Lee, P. Y. (2013). Use Of Data Mining In Business Analytics To Support Business Competitiveness. *SIM University [UniSIM]*, 17(2), 53–58.
<https://doi.org/10.19030/rbis.v17i2.7843>

Maheshwari, A. K. (2014). *Business Intelligence and Data Mining*.

Massaro, A., Vitti, V., Galiano, A., & Morelli, A. (2019). Business Intelligence Improved by Data Mining Algorithms and Big Data Systems: An Overview of Different Tools Applied in Industrial Research. *Computer Science and Information Technology*, 7(1), 1–21.
<https://doi.org/10.13189/csit.2019.070101>

Prasanna. (2022). Data Mining Advantages And Disadvantages | What is Data Mining?, Pros and Cons of Data Mining. *A Plus Topper*.
https://www.aplustopper.com/data-mining-advantages-and-disadvantages/#Disadvantages_of_Data_Mining

Simplilearn. (2023). What Is Data Mining: Definition, Benefits, Applications, and More. *Simplilearn.com*. <https://www.simplilearn.com/what-is-data-mining-article>

Şimşek, H. (2023a). Data Mining: What is it & Why do businesses need it in 2023? *AIMultiple*. <https://research.aimultiple.com/data-mining/>

Şimşek, H. (2023a). How to Apply Data Mining in Business Analytics in '23. *AIMultiple*. <https://research.aimultiple.com/data-mining-business-analytics/>

- Şimşek, H. (2023). How to Pair Data Mining & Business Intelligence in '23? *AIMultiple*. <https://research.aimultiple.com/data-mining-business-intelligence/>
- Talend. (n.d.). Business Intelligence vs. Data Mining: A Comparison. *Talend - a Leader in Data Integration & Data Integrity*. <https://www.talend.com/resources/business-intelligence-data-mining/>
- Thuraisingham, B. (2003). Web Data Mining and Applications in Business Intelligence and Counter-Terrorism. In *CRC Press eBooks*. <https://doi.org/10.1201/9780203499511>
- Tiwari, R. (2022, December 29). 5 Key Differences between Data Mining and Business Intelligence Simplified. *Learn / Hevo*. <https://hevodata.com/learn/data-mining-and-business-intelligence/>
- Tobin, D. (2020). 3 Data Mining & Business Intelligence Cases. *Integrate.io*. <https://www.integrate.io/blog/real-life-applications-of-data-mining-and-business-intelligence/>
- Van Der Lans, R. F. (2012). *Data Virtualization for Business Intelligence Systems: Revolutionizing Data Integration for Data Warehouses*. <https://dl.acm.org/citation.cfm?id=2423868>

Western Governors University. (2023, March 31). Data Mining in Business Analytics.

Western Governors University. <https://www.wgu.edu/blog/data-mining-business-analytics2005.html#close>

14 Data Mining Projects With Source Code [2023]. (2023, March 1). InterviewBit.

<https://www.interviewbit.com/blog/data-mining-projects/>