**Table Of Contents**

[ABSTRACT 2](#_Toc172823787)

[INTRODUCTION 3](#_Toc172823788)

[ABOUT THE PROJECT 8](#_Toc172823789)

[SPOTIFY LIKENESS CLASSIFICATION ANALYSIS PROJECT SURVEY 9](#_Toc172823790)

[SYSTEM ANALYSIS 11](#_Toc172823791)

[DESIGN 17](#_Toc172823792)

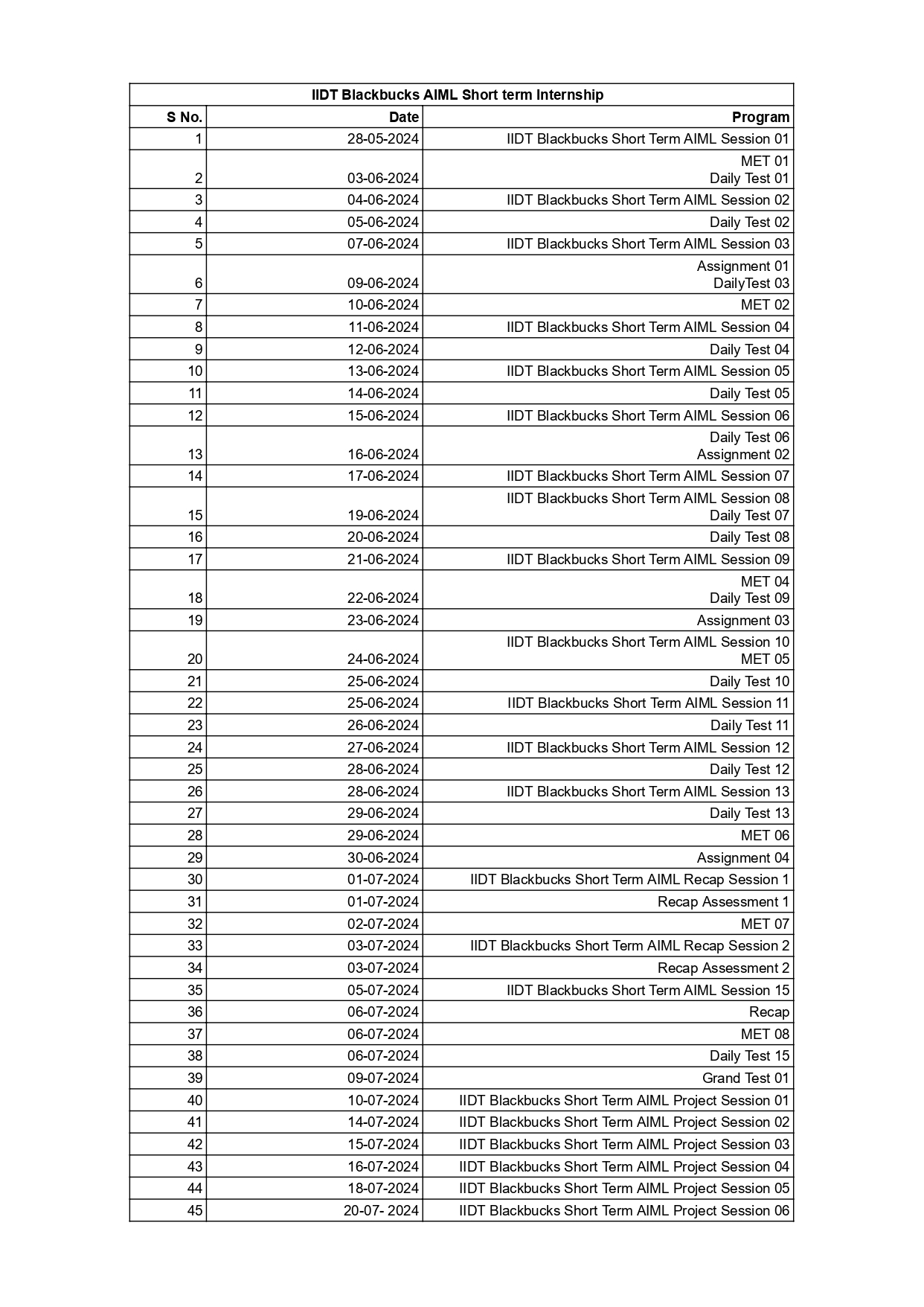
[IMPLEMENTATION 25](#_Toc172823793)

[TESTING 30](#_Toc172823794)

[OUTPUT SCREENS 32](#_Toc172823795)

[CONCLUSION 38](#_Toc172823796)

[BIBLIOGRAPHY 39](#_Toc172823797)

****

# **ABSTRACT**

This project is about *Spotify Likeness Classification Analysis.*

This project addresses the challenge of accurately predicting song likability on Spotify by developing a machine learning model that integrates audio features of songs with user interaction metrics. By analyzing data such as tempo, energy, danceability, skip rates, and play counts, the project aims to identify which songs users are likely to enjoy. This approach enhances the effectiveness of music recommendation systems, leading to more personalized user experiences and deeper insights into the factors that drive music preferences, ultimately improving user engagement and satisfaction on the platform.

In the highly competitive and rapidly evolving landscape of music streaming services, understanding user preferences and predicting song likability is critical for providing personalized listening experiences. This study focuses on developing and evaluating a predictive model that determines whether a particular song is likely to be liked by a user on Spotify. By leveraging a comprehensive dataset that includes both audio features of songs and user interaction metrics, we aim to enhance the accuracy and effectiveness of song recommendation systems.

In the domain of music streaming, personalizing user experience is paramount to user engagement and satisfaction. This project aims to develop a predictive model for classifying song likability on Spotify. By analyzing an extensive dataset comprising both audio features (such as tempo, energy, and danceability) and user interaction metrics (like skip rates and play counts), we seek to determine which songs are likely to be liked by users. Various machine learning algorithms, including logistic regression, decision trees, and neural networks, are employed to build the classification model.

The performance of these models is evaluated using accuracy, precision, recall, and F1-score. Our results indicate that combining audio features with user interaction data significantly enhances the prediction accuracy, with neural networks showing the most promising performance. This project not only contributes to improving music recommendation systems but also provides deeper insights into the factors influencing user preferences, ultimately enabling more personalized and enjoyable music streaming experiences.

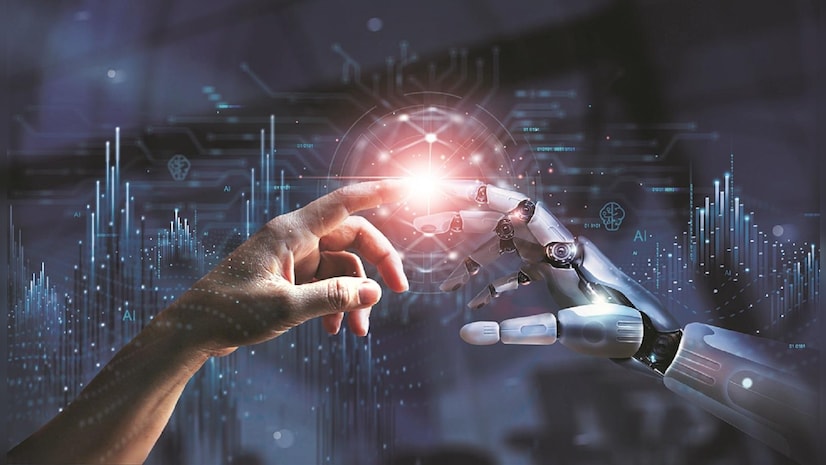
# **Introduction**

**What is Artificial Intelligence (AI)?**

Artificial Intelligence (AI) is a branch of computer science focused on creating systems capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, understanding natural language, perception, and even exhibiting creativity.

**Key Areas of AI:**

1. **Machine Learning (ML):** A subset of AI where algorithms are used to find patterns or insights in data. The systems "learn" from data and improve over time without being explicitly programmed for specific tasks. Common techniques include supervised learning, unsupervised learning, and reinforcement learning.
2. **Natural Language Processing (NLP):** The ability of a machine to understand, interpret, and generate human language. Applications include chatbots, language translation, sentiment analysis, and speech recognition.
3. **Computer Vision:** The capability of AI systems to interpret and make decisions based on visual input. This includes image and video recognition, object detection, and facial recognition.



**What is Machine Learning (ML)?**

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that involves the development of algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data. Instead of being explicitly programmed to perform a specific task, machine learning systems improve their performance over time as they are exposed to more data.

**Key Concepts in Machine Learning:**

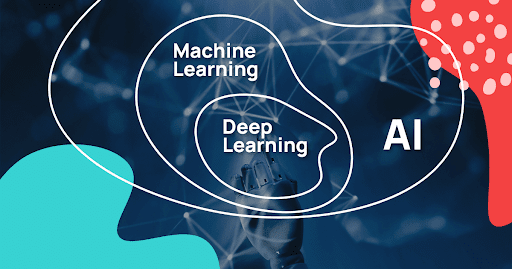
1. **Data:** The foundational element for ML, which includes the information the system learns from. Data can be structured (like databases) or unstructured (like text and images).
2. **Algorithms:** These are the mathematical and computational procedures used by ML systems to learn from data. Examples include decision trees, neural networks, and support vector machines.
3. **Model:** A model is the output generated by the ML algorithm after training on data. It represents the learned patterns and can be used to make predictions.
4. **Training:** The process of feeding data into an ML algorithm to help it learn. During training, the model adjusts its parameters to minimize errors.
5. **Testing:** Evaluating the model's performance on a separate set of data that was not used during training to assess its accuracy and generalization.
6. **Features:** Individual measurable properties or characteristics of the data used by the model for learning.
7. **Labels:** In supervised learning, labels are the known outcomes or categories associated with the training data.

**Types of Machine Learning:**

1. **Supervised Learning:**
   * The algorithm learns from labeled data, where the input data and the corresponding correct output are provided.
   * Common applications include classification (e.g., spam detection) and regression (e.g., predicting house prices).
2. **Unsupervised Learning:**
   * The algorithm learns from unlabeled data, identifying patterns and structures within the data.
   * Common applications include clustering (e.g., customer segmentation) and dimensionality reduction (e.g., principal component analysis).
3. **Semi-Supervised Learning:**
   * A combination of supervised and unsupervised learning, where the algorithm learns from a small amount of labeled data and a larger amount of unlabeled data.
4. **Reinforcement Learning:**
   * The algorithm learns by interacting with an environment, receiving feedback in the form of rewards or penalties based on its actions.
   * Common applications include game playing (e.g., AlphaGo) and robotics.

**Key Techniques and Algorithms:**

* **Linear Regression:** Used for predicting a continuous outcome based on one or more input variables.
* **Logistic Regression:** Used for binary classification problems.
* **Decision Trees:** A tree-like model used for classification and regression tasks.
* **Random Forests:** An ensemble method that uses multiple decision trees to improve accuracy and prevent overfitting.
* **Support Vector Machines (SVM):** A classification technique that finds the optimal hyperplane separating different classes.
* **Neural Networks:** A set of algorithms modeled after the human brain, used for complex pattern recognition tasks.
* **K-Means Clustering:** An unsupervised learning algorithm for partitioning data into clusters.

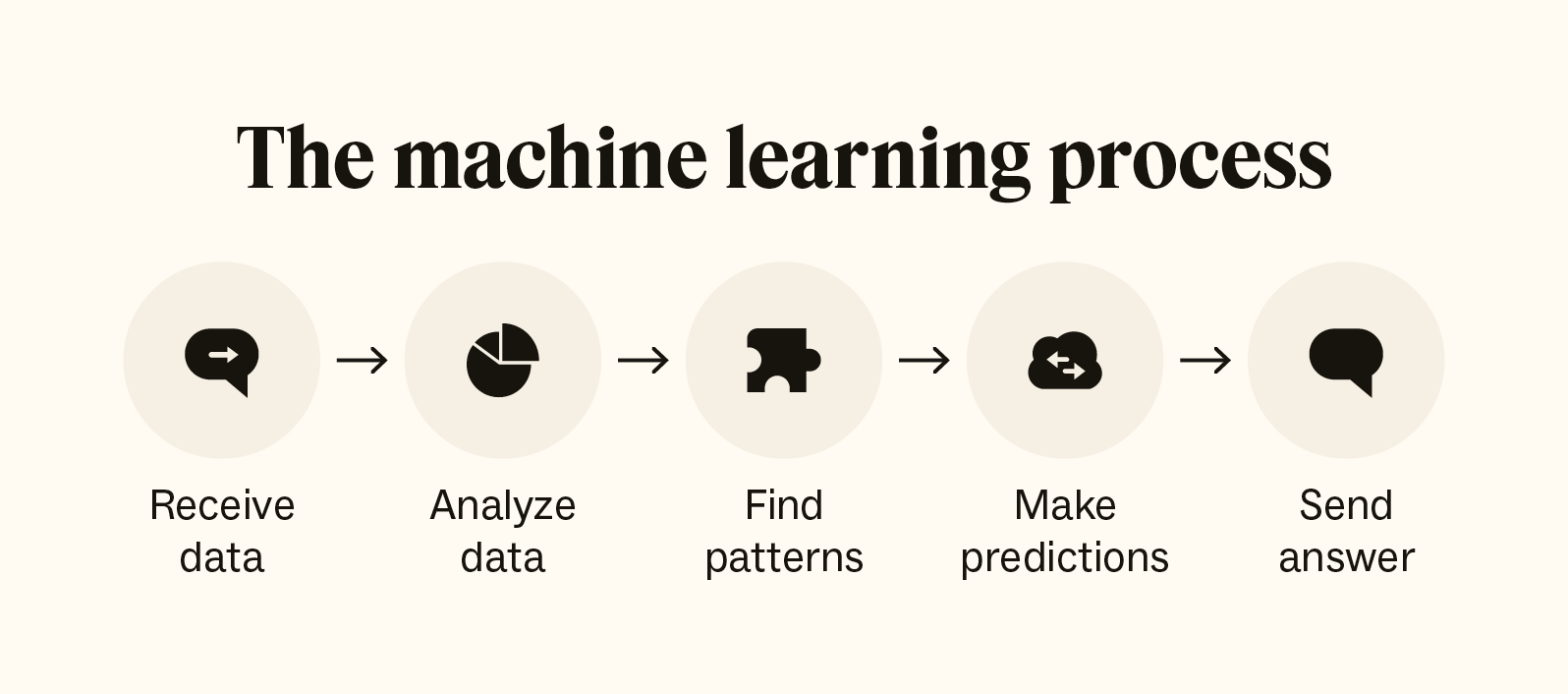


**Applications of Machine Learning:**

* **Healthcare:** Disease diagnosis, personalized treatment plans, and medical image analysis.
* **Finance:** Credit scoring, fraud detection, and algorithmic trading.
* **Retail:** Recommendation systems, inventory management, and customer segmentation.
* **Transportation:** Autonomous vehicles, traffic prediction, and route optimization.
* **Entertainment:** Content recommendation on streaming platforms, video and music suggestions.

**Challenges in Machine Learning:**

* **Data Quality:** ML models require high-quality, relevant data for accurate predictions.
* **Overfitting and Underfitting:** Ensuring the model generalizes well to new data without being too specific or too simple.
* **Interpretability:** Understanding how complex models, especially deep learning models, make decisions.
* **Scalability:** Handling large datasets and computational requirements efficiently.
* **Bias and Fairness:** Addressing biases in the data and ensuring fair outcomes.



**What is Deep Learning (DL)?**

Deep Learning (DL) is a subset of Machine Learning (ML) that focuses on using neural networks with many layers (hence "deep") to model complex patterns in data. These neural networks, known as deep neural networks (DNNs), are designed to mimic the structure and function of the human brain, allowing them to learn from large amounts of data.

**Key Concepts in Deep Learning:**

1. **Neural Networks:**
   * **Neurons:** Basic units of a neural network that take inputs, apply a function (usually nonlinear), and produce an output.
   * **Layers:** Neural networks consist of multiple layers:
     + **Input Layer:** The first layer that receives the input data.
     + **Hidden Layers:** Intermediate layers where data transformation occurs. The depth of the network refers to the number of hidden layers.
     + **Output Layer:** The final layer that produces the prediction or classification.
2. **Activation Functions:**
   * Functions applied to the output of each neuron to introduce nonlinearity into the model, enabling it to learn complex patterns.
   * Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.
3. **Backpropagation:**
   * A training algorithm used to minimize the error by adjusting the weights of the neurons. It involves calculating the gradient of the loss function with respect to each weight by the chain rule, propagating errors backward from the output to the input layer.
4. **Loss Function:**
   * A function that measures the difference between the predicted output and the actual target. The goal is to minimize this loss during training.
   * Common loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks.
5. **Optimization Algorithms:**
   * Methods used to adjust the weights of the network to minimize the loss function.
   * Common optimization algorithms include Stochastic Gradient Descent (SGD), Adam, and RMSprop.

**What is Natural Language Processing (NLP)?**

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and human language. It involves the development of algorithms and models that enable machines to understand, interpret, generate, and respond to human language in a meaningful way. NLP combines computational linguistics, computer science, and statistical methods to process and analyze large amounts of natural language data.

**Key Concepts in NLP:**

1. **Tokenization:**
   * The process of breaking down text into smaller units called tokens, such as words, phrases, or sentences.
2. **Parsing:**
   * Analyzing the grammatical structure of a sentence to understand the relationships between words and phrases. This includes syntactic parsing (analyzing sentence structure) and semantic parsing (extracting meaning from text).
3. **Named Entity Recognition (NER):**
   * Identifying and classifying named entities in text, such as people, organizations, locations, dates, and other proper nouns.
4. **Part-of-Speech (POS) Tagging:**
   * Assigning parts of speech (e.g., nouns, verbs, adjectives) to each word in a sentence.
5. **Sentiment Analysis:**
   * Determining the sentiment or emotional tone of a piece of text, such as positive, negative, or neutral.
6. **Machine Translation:**
   * Automatically translating text from one language to another.
7. **Text Classification:**
   * Assigning predefined categories or labels to text based on its content. Common applications include spam detection and topic classification.
8. **Speech Recognition:**
   * Converting spoken language into written text.
9. **Language Generation:**
   * Creating human-like text based on a given input, such as text summarization, chatbots, and content creation.

# **ABOUT THE PROJECT**

**Project Definition**

The Spotify Likeness Classification Analysis project aims to tackle the challenge of predicting whether a user will like a particular song on Spotify. In an era where personalized music recommendations are crucial for user engagement, accurately forecasting user preferences is essential. This project focuses on understanding and modeling the factors that influence song likability using machine learning techniques. The primary goal is to develop a predictive model that can classify songs as liked or not liked based on various features extracted from the songs and user interaction data.

**Proposed Solution**

To address this challenge, we propose using logistic regression as the primary machine learning algorithm for our classification model. Logistic regression is well-suited for binary classification problems and provides a clear interpretation of the relationship between input features and the predicted outcome. In this project, we will leverage a comprehensive dataset that includes audio features of songs, such as tempo, energy, danceability, and acousticness, alongside user interaction metrics like skip rates, play counts, and user ratings.

The logistic regression model will be trained to learn the patterns and correlations between these features and the likelihood of a song being liked by users. The performance of the model will be evaluated using standard metrics such as accuracy, precision, recall, and F1-score. By integrating both audio and interaction data, we aim to create a robust and interpretable model that enhances Spotify's recommendation system, ultimately leading to a more personalized and satisfying user experience.

**Objective**

The objective of this project is to develop a logistic regression-based model that accurately classifies songs based on their likelihood of being liked by users. By analyzing and integrating audio features with user interaction data, the project aims to improve the personalization of music recommendations on Spotify. This will enhance user satisfaction by providing more relevant song suggestions and gaining insights into the factors that influence user preferences.

# **Spotify Likeness Classification Analysis Project Survey**

**Theoretical Background**

The Spotify Likeness Classification Analysis project is grounded in the principles of machine learning and statistical analysis. Machine learning, particularly supervised learning, involves training algorithms on labeled data to make predictions or classifications. In the context of this project, the goal is to classify whether a user is likely to like a particular song based on historical interaction data and song attributes. Logistic regression, a widely used statistical method, is particularly suited for binary classification tasks. It models the probability of a binary outcome based on one or more predictor variables. By estimating the relationship between input features (such as song characteristics and user behavior) and the likelihood of user preferences, logistic regression helps in predicting whether a song will be liked.

**Existing System with Drawbacks**

Current music recommendation systems, including those used by Spotify, employ a variety of techniques to suggest songs to users. These systems typically utilize collaborative filtering, content-based filtering, or a hybrid approach to generate recommendations. Collaborative filtering relies on user interaction data (e.g., ratings, play counts) to find patterns and make suggestions based on similar users’ preferences. Content-based filtering focuses on the attributes of songs themselves (e.g., genre, tempo) and matches them with users’ past preferences.

**Drawbacks of Existing Systems:**

1. **Limited Interpretability:** Many current models, especially those using deep learning, can be complex and difficult to interpret, making it challenging to understand why a particular recommendation was made.
2. **Overfitting to User Behavior:** Collaborative filtering can overfit to a user’s historical behavior, potentially leading to a lack of diversity in recommendations.
3. **Cold Start Problem:** Both approaches struggle with new users or new songs that have limited interaction history, leading to less accurate recommendations.
4. **Data Sparsity:** The reliance on historical interaction data can lead to challenges in predicting preferences for users with sparse or limited data.

**Proposed System with Features**

The proposed system utilizes logistic regression to classify song likability. Logistic regression is chosen for its effectiveness in handling binary classification problems and its ability to provide clear insights into feature importance. The model will be trained using a dataset that includes both audio features of songs and user interaction metrics.

**Features of the Proposed System:**

1. **Integration of Audio Features:** The model uses features such as tempo, energy, danceability and tempo which are known to influence user preferences.
2. **Inclusion of User Interaction Data:** Metrics like skip rates, play counts, and user ratings are incorporated to capture user behavior and preferences.
3. **Binary Classification:** Logistic regression classifies songs into two categories: liked or not liked, based on the combined input features.
4. **Interpretability:** Logistic regression provides interpretable results, allowing stakeholders to understand the impact of different features on the prediction of song likability.

**Advantages of the Proposed System**

1. **Enhanced Interpretability:** Unlike more complex models, logistic regression offers clear insights into how different features influence the likelihood of a song being liked, making the results more understandable and actionable.
2. **Improved Personalization:** By combining audio features with user interaction data, the model can better capture the nuances of user preferences and provide more accurate song recommendations.
3. **Reduced Overfitting:** Logistic regression is less prone to overfitting compared to more complex models, especially when handling smaller datasets or new users/songs.
4. **Efficient Performance:** Logistic regression is computationally efficient and can handle large datasets effectively, making it suitable for real-time recommendation systems.
5. **Flexibility:** The model can be easily adapted and updated with new data, allowing for continuous improvement in recommendation accuracy.

In summary, the proposed logistic regression-based system aims to enhance Spotify's song recommendation capabilities by integrating comprehensive data features and providing a transparent and efficient solution for predicting song likability.

# **SYSTEM ANALYSIS**

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

**Specification**

**Functional Requirements**

The following are the functional requirements of our project:

* A training dataset has to be created on which training is performed.
* A testing dataset has to be created on which testing is performed.

**Non-Functional Requirements**

The following are the non-functional requirements of our project:

* **Maintainability:** Maintainability is used to make future maintenance easier, meet new requirements.
* **Robustness:** Robustness is the quality of being able to withstand stress, pressures or changes in procedure or circumstance.
* **Reliability:** Reliability is an ability of a person or system to perform and maintain its functions in circumstances.
* **Size:** The size of a particular application play a major role, if the size is less then efficiency will be high.
* **Speed:** If the speed is high then it is good. Since the no of lines in our code is less, hence the speed is high.

**Software Requirements**

One of the most difficult tasks is that, the selection of the software, once system requirement is known that is determining whether a particular software package fits the requirements.

|  |  |
| --- | --- |
| **Programming Language** | **Python** |
| **Technology** | **Jupyter** |
| **Operating System** | **Windows 11** |
| **Browser** | **Google Chrome** |

**Hardware Requirements**

The selection of hardware is very important in the existence and proper working of any software. In the selection of hardware, the size and the capacity requirements are also important.

|  |  |
| --- | --- |
| **Processor** | **Intel Core** |
| **RAM Capacity** | **4GB** |
| **Hardisk** | **512 GB** |
| **I/O Devices** | **Keyboard, Mouse, Monitor** |

**Module Description**

For predicting the literacy rate of India, our project has been divided into following modules:

1. Data Analysis & Pre-processing

2. Model Training &Testing

3. Accuracy Measures

4. Prediction & Visualization

**1.** **Data Analysis & Pre-processing**

Data Analysis is done by collecting raw data from different literacy websites. Data pre-processing technique involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. We use pandas module for Data Analysis and pre- processing.

**Pandas:**

In order to be able to work with the data in Python, we'll need to read the csv file into a Pandas Data Frame. A Data Frame is a way to represent and work with tabular data. Tabular data has rows and columns, just like our csv file.

**2. Model Training &Testing**

For Literacy rate prediction, we perform “converting into 2D array” and “scaling using normalization” operations on data for further processing. We use fit\_transform to center the data in a way that it has 0 mean and 1 standard error. Then, we divide the data into x\_train and y\_train. Our model will get the 0-th element from x\_train and try to predict the 0-th element from y\_train. Finally, we reshape the x\_train data to match the requirements for training using keras. Now we need to train our model using the above data.

The algorithm that I have used is Logistic Regression.

**Logistic Regression**

Logistic regression is a statistical method used for binary classification tasks, where the goal is to predict one of two possible outcomes based on one or more predictor variables. Unlike linear regression, which predicts continuous outcomes, logistic regression models the probability that a given input belongs to a particular class. It achieves this by applying a logistic function (also known as the sigmoid function) to a linear combination of the input features. The sigmoid function transforms the output of the linear equation into a value between 0 and 1, which can be interpreted as the probability of the input belonging to the positive class.

Mathematically, logistic regression works by first calculating the linear combination of the input features using an equation and sigma function.

This function outputs a value between 0 and 1, representing the probability that the input belongs to the positive class. To make a final classification decision, a threshold (usually 0.5) is applied to this probability. If the probability is greater than or equal to the threshold, the model predicts the positive class; otherwise, it predicts the negative class. The model parameters (coefficients) are estimated using a method called maximum likelihood estimation (MLE), which finds the values that maximize the likelihood of the observed data. This involves iteratively adjusting the coefficients to minimize the difference between the predicted probabilities and the actual outcomes in the training data, often using optimization algorithms like gradient descent.

**Testing:**

In testing, now we predict the data. Here we have 2 steps: predict the literacy rate and plot it to compare with the real results.Using fit transform to scale the data and then reshape it for the prediction. Predict the data and rescale the predicted data to match its real values. Then plot real and predicted literacy rate on a graph. Then calculate the accuracy. We use Sklearn and Numpy python module for Training and testing.

**Sklearn:**

It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy.

**Numpy:**

Numpy is the core library for scientific computing in Python. It provides a highperformance multidimensional array object, and tools for working with these arrays.It is used for Numerical Calculations.

**3. Accuracy Measures**

The Accuracy of the model is to be evaluated to figure out the correctness of the prediction. The proposed model got 87% Accuracy.

**4. Prediction & Visualization**

Using the Proposed model prediction is made for coming years. Graphs are used to visualize state wise literacy rate predictions. We use Matplotlib python module for Visualization.

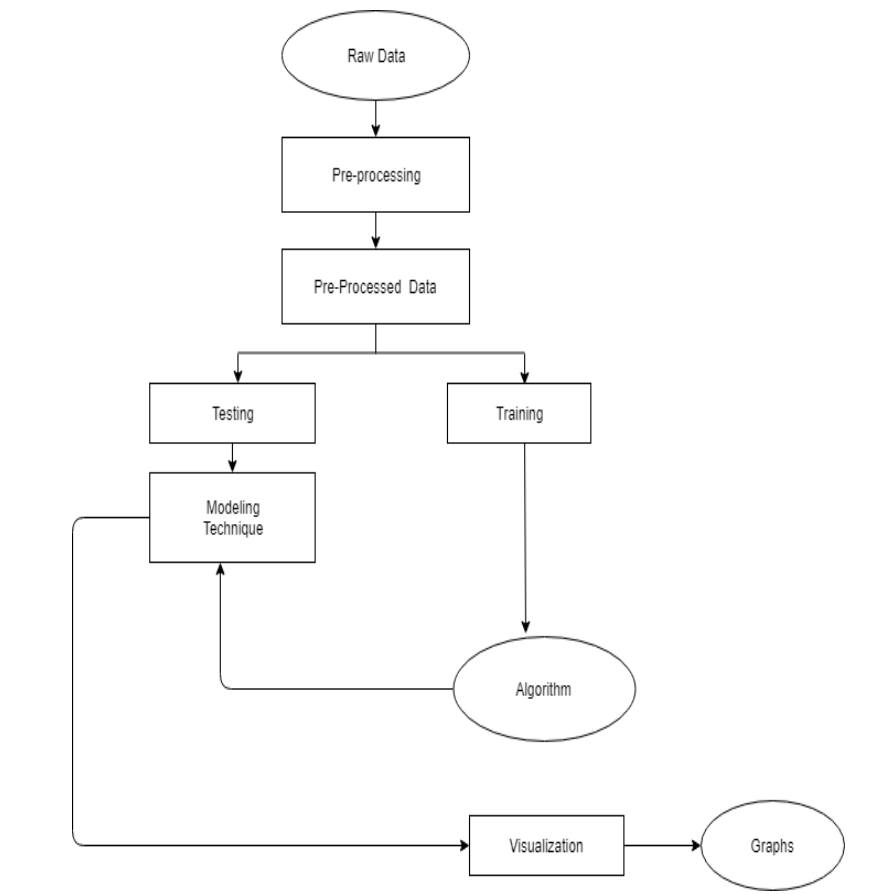
**Matplotlib:**

It is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.

# **DESIGN**

**Block Diagram**

The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.



**Data Flow Diagrams:**

Data flow diagram (DFD) is a graphical representation of “flow” of data through an information system, modelling its process concepts. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFD’s can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It doesn’t show information about timing of processes, or information about whether processes will operate in sequence or parallel. A DFD is also called as “bubble chart”.

**DFD Symbols:**

In the DFD, there are four symbols:

* A square define a source or destination of system data.
* An arrow indicates dataflow. It is the pipeline through which the information flows.
* A circle or a bubble represents transforms dataflow into outgoing dataflow.
* An open rectangle is a store, data at reset or at temporary repository of data.

**Dataflow:** Data move in a specific direction from an origin to a destination.

A black line on a white background

Description automatically generated

**Process:** People, procedures or devices that use or produce (Transform) data. The physical component is not identified.

A black circle with a white background

Description automatically generated

**Sources:** External sources or destination of data, which may be programs, organizations or other entity.

A black and white rectangle

Description automatically generated

**Data store:** Here data is stored or referenced by a process in the system’s #

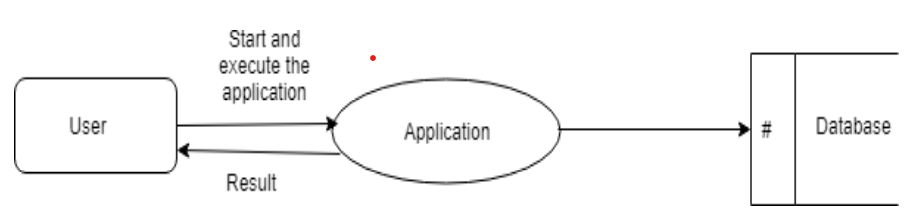
A rectangular object with black lines

Description automatically generated

In our project, we had built the data flow diagrams at the very beginning of business process modelling in order to model the functions that our project has to carry out and the interaction between those functions together with focusing on data exchanges between processes.

**Context level DFD:**

A Context level Data flow diagram created using select structured systems analysis and design method (SSADM). This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are “owned” by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities**.**

****

**Top level DFD:**

A data flow diagram is that which can be used to indicate the clear progress of a business venture. In the process of coming up with a data flow diagram, the level one provides an overview of the major functional areas of the undertaking. After presenting the values for most important fields of discussion, it gives room for level two to be drawn.

A diagram of a application

Description automatically generated

**Unified Modelling Language Diagrams:**

The Unified Modelling Language (UML) is a Standard language for specifying, visualizing, constructing and documenting the software system and its components. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows:

* **User Model View**
  1. This view represents the system from the user’s perspective.
  2. The analysis representation describes a usage scenario from the end-users perspective.
* **Structural Model View** 
  1. In this model the data and functionality arrived from inside the system.
  2. This model view models the static structures.
* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

Inthese the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**Use Case Diagram:**

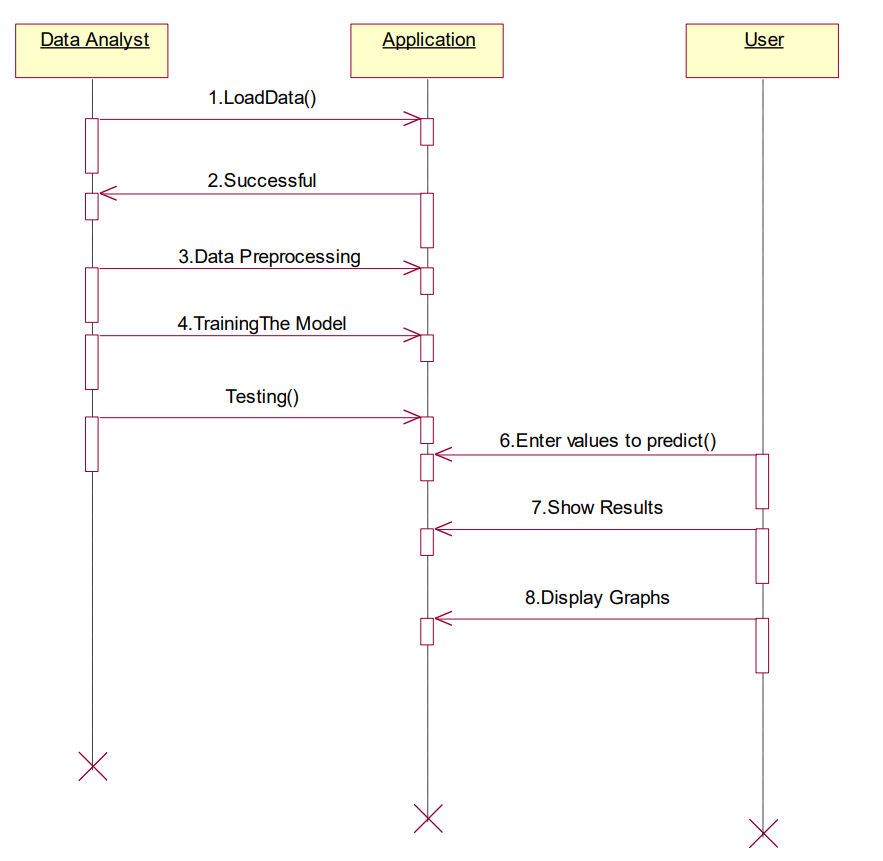
Use case diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of the systems (activity diagrams, sequence diagram, state chart diagram, collaboration diagram are the four other kinds of diagrams in the UML for modeling the dynamic aspects of systems).

Use case diagrams are central to modeling the behavior of the system, a sub-system, or a class. Each one shows a set of use cases and actors and relations.



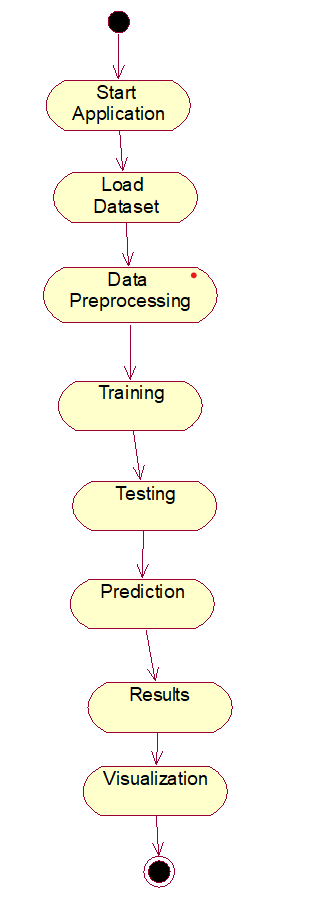
**Sequence Diagram:**

Sequence diagram is an interaction diagram which is focuses on the time ordering of messages. It shows a set of objects and messages exchanged between these objects. This diagram illustrates the dynamic view of a system.



**Activity Diagram:**

An Activity diagram shows the flow from activity to activity within a system it emphasizes the flow of control among objects.



**Data Dictionary**

**A screenshot of a computer

Description automatically generated**

# **IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

The project is implemented by accessing simultaneously from more than one system and more than one window in one system. The application is implemented in the Internet Information Services 5.0 web server under the Windows XP and accessed from various clients.

**Technologies Used**

**What is Python?**

Python is an interpreter, high-level programming language for general-purpose programming by “Guido van Rossum” and first released in 1991, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional, procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. Python is managed by the non-profit Python Software Foundation.

Python is a general purpose, dynamic, high level and interpreted programming language. It supports object-oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high level data structures.

**Libraries Of python:**

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 33), but most modules are not.

They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of March 2018, the Python Package Index (PyPI), the official repository for thirdparty Python software, contains over 130,000 packages with a wide range of functionality, including:

* Graphical user interfaces
* Web frameworks
* Multimedia
* Databases
* Networking
* Test frameworks
* Automation
* Web scraping
* Documentation
* System administration

**Classification**

A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. For example, when filtering emails “spam” or “not spam”, when looking at transaction data, “fraudulent”, or “authorized”. In short Classification either predicts categorical class labels or classifies data (construct a model) based on the training set and the values (class labels) in classifying attributes and uses it in classifying new data.

There are a number of classification models. The Classification model I choose is Logistic regression.

**Logistic Regression**

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

**Modules in python**

**Module**:

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

**Pandas:**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

Pandas is well suited for many different kinds of data:

* Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spread sheet.
* Ordered and unordered (not necessarily fixed-frequency) time series data.
* Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.
* Any other form of observational / statistical data sets. The data actually need not be labelled at all to be placed into a panda’s data structure

Using pandas we can :

* Easy handling of missing data (represented as Nan) in floating point as well as non-floating-point data.
* Size mutability.
* Automatic and explicit data alignment.
* Flexible reshaping and pivoting of data sets.

**NumPy:**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric, the ancestor of NumPy, was developed by Jim Humulin. Another package Numara was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numara into Numeric package. There are many contributors to this open source project.

NumPy – A Replacement for MATLAB.

**Sickit-learn:**

Scikit-learn (formerly scikits. learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits. learn, a Google Summer of Code project by David Courmayeur. Its name stems from the notion that it is a “SciKit”(SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.

Some popular groups of models provided by scikit-learn include:

* **Ensemble methods:** for combining the predictions of multiple supervised models.
* **Feature extraction:** for defining attributes in image and text data.
* **Feature selection:** for identifying meaningful attributes from which to create supervised models.
* **Parameter Tuning:** for getting the most out of supervised models.
* **Supervised Models:** a vast array not limited to generalize linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

**Matplotlib:**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

**Seaborn:**

Seaborn is a powerful and user-friendly data visualization library in Python, built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn simplifies the process of creating complex plots by offering built-in themes, color palettes, and functions for visualizing distributions, relationships, and categorical data. With its concise and consistent syntax, Seaborn makes it easier to generate aesthetically pleasing plots such as heatmaps, bar plots, scatter plots, and violin plots, which are essential for exploratory data analysis and communicating insights effectively.

# **TESTING**

It is the process of testing the functionality and it is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as at undiscovered error. A successful test is one that uncovers an as at undiscovered error.

**Black Box Testing:**

The base of the black box testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming to route the testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test application by the developer.

The following are different types of techniques involved in black box testing. They are:

* Decision Table Testing
* All pairs testing
* State transition tables testing
* Equivalence Partitioning

Software testing is used in association with Verification and Validation. Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques as reviews, inspections, walk-through.

Validation is the process of checking what has been specified is what the user actually wanted.

* Validation: Are we doing the right job?
* Verification: Are we doing the job right?

In order to achieve consistency in the Testing style, it is imperative to have and follow a set of testing principles. This enhances the efficiency of testing within SQA team members and thus contributes to increased productivity. The purpose of this document is to provide overview of the testing, plus the techniques. Here, after training is done on the training dataset, testing is done.

**White Box Testing:**

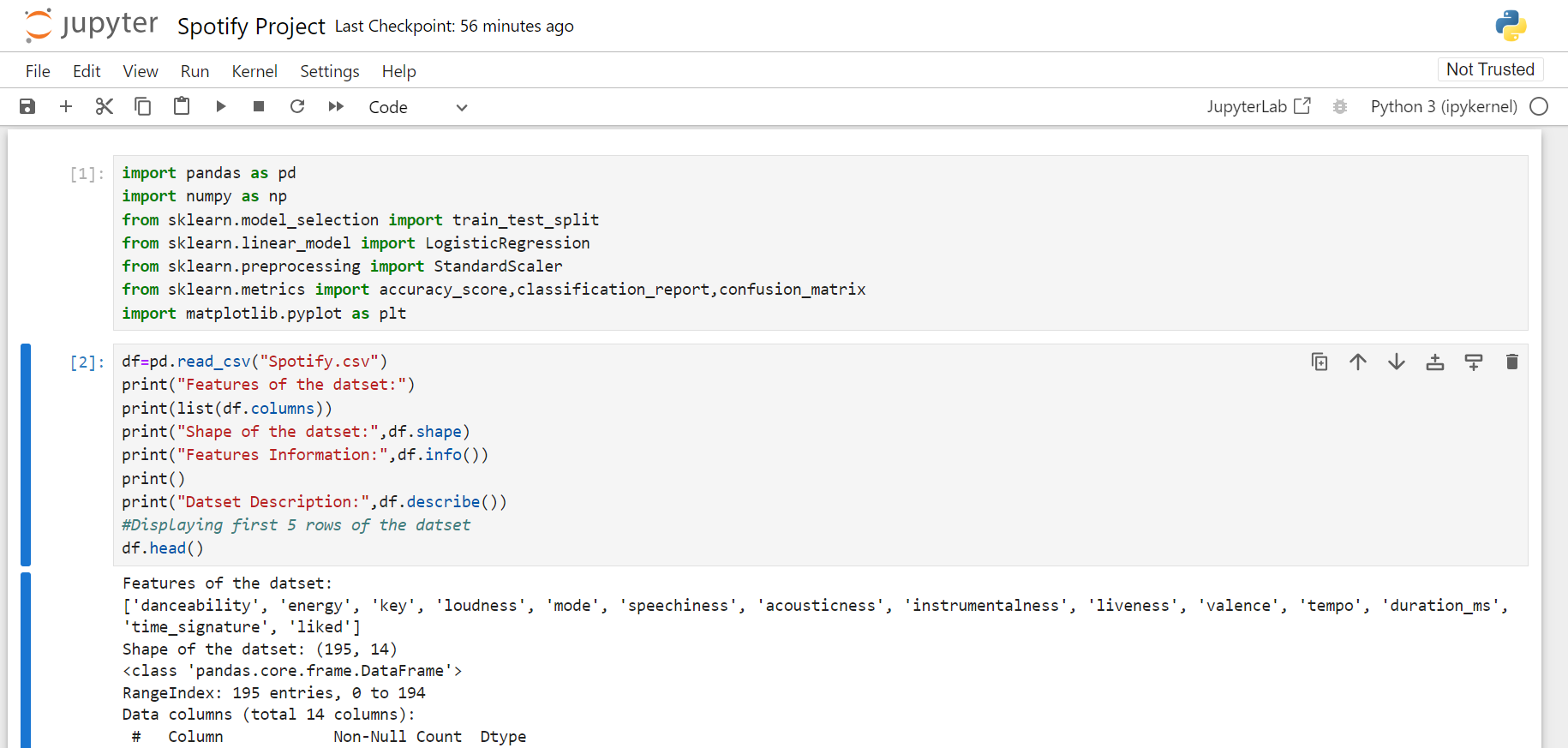
White box testing [10] requires access to source code. Though white box testing [10] can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing [10] during unit testing phase.

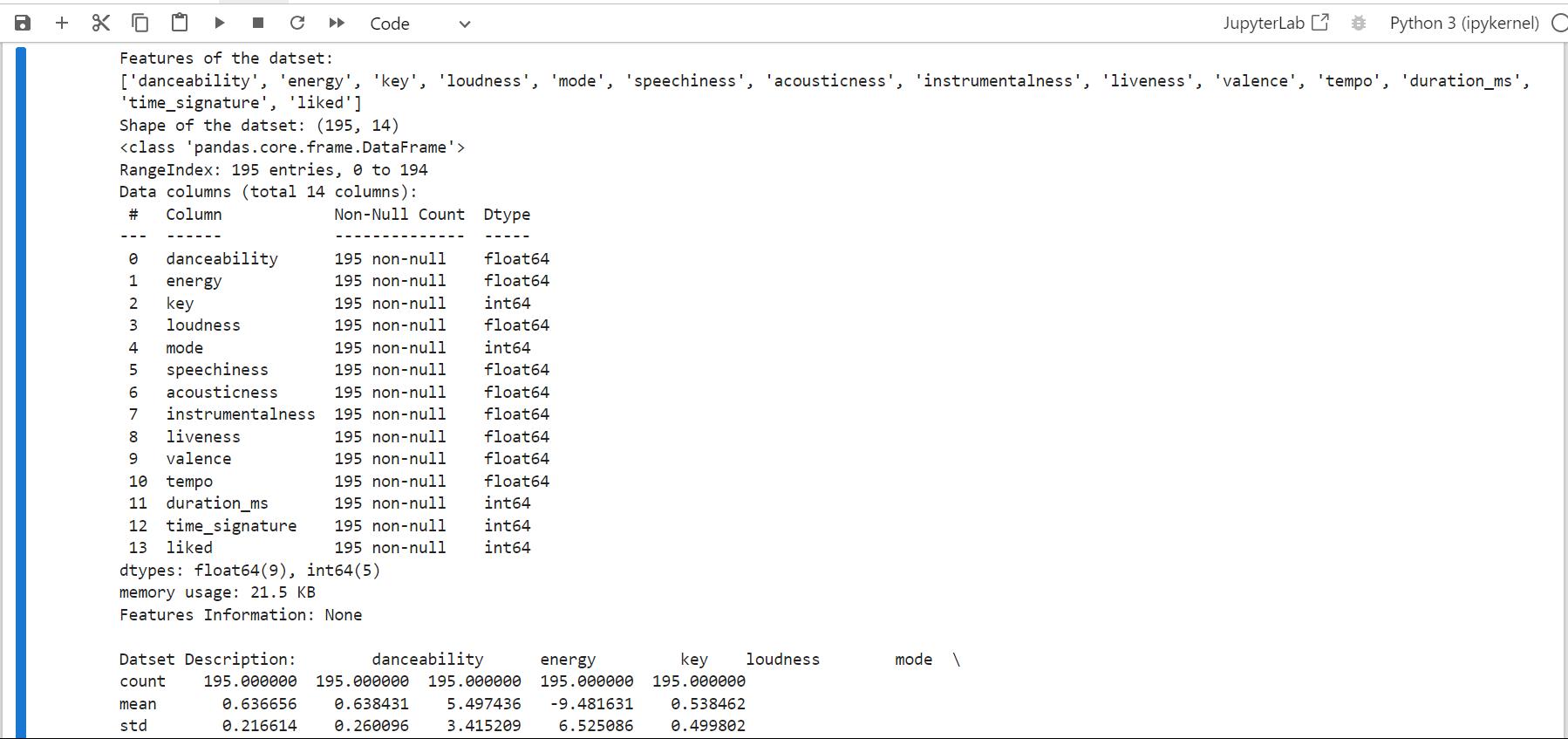
In designing of database the flow of specific inputs through the code, expected output and the functionality of conditional loops are tested.

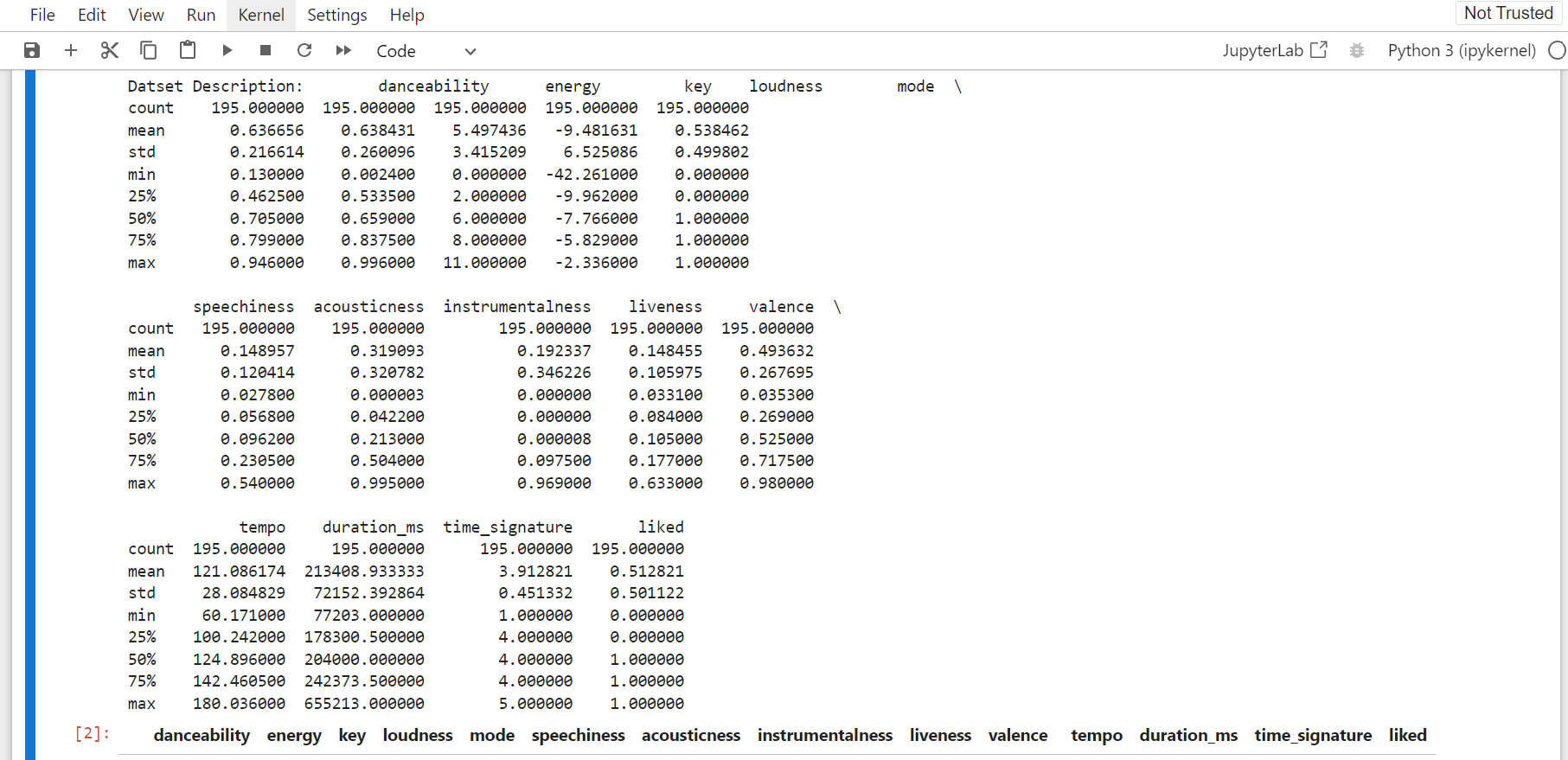
At SDEI, 3 levels of software testing is done at various SDLC phases

* **Unit Testing:** This is in which each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented
* **Integration Testing:** This is in which progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a whole.
* **System Testing:** This is in which the software is integrated to the overall product and tested to show that all requirements are met. A further level of testing is also done, in accordance with requirements.
* **Regression Testing:** It is used to refer the repetition of the earlier successful tests to ensure that changes made in the software have not introduced new bugs/side effects.
* **Acceptance Testing:** Testing to verify a product meets customer specified requirements. The acceptance test suite is run against supplied input data. Then the results obtained are compared with the expected results of the client. A correct match was obtain.

# **OUTPUT SCREENS**





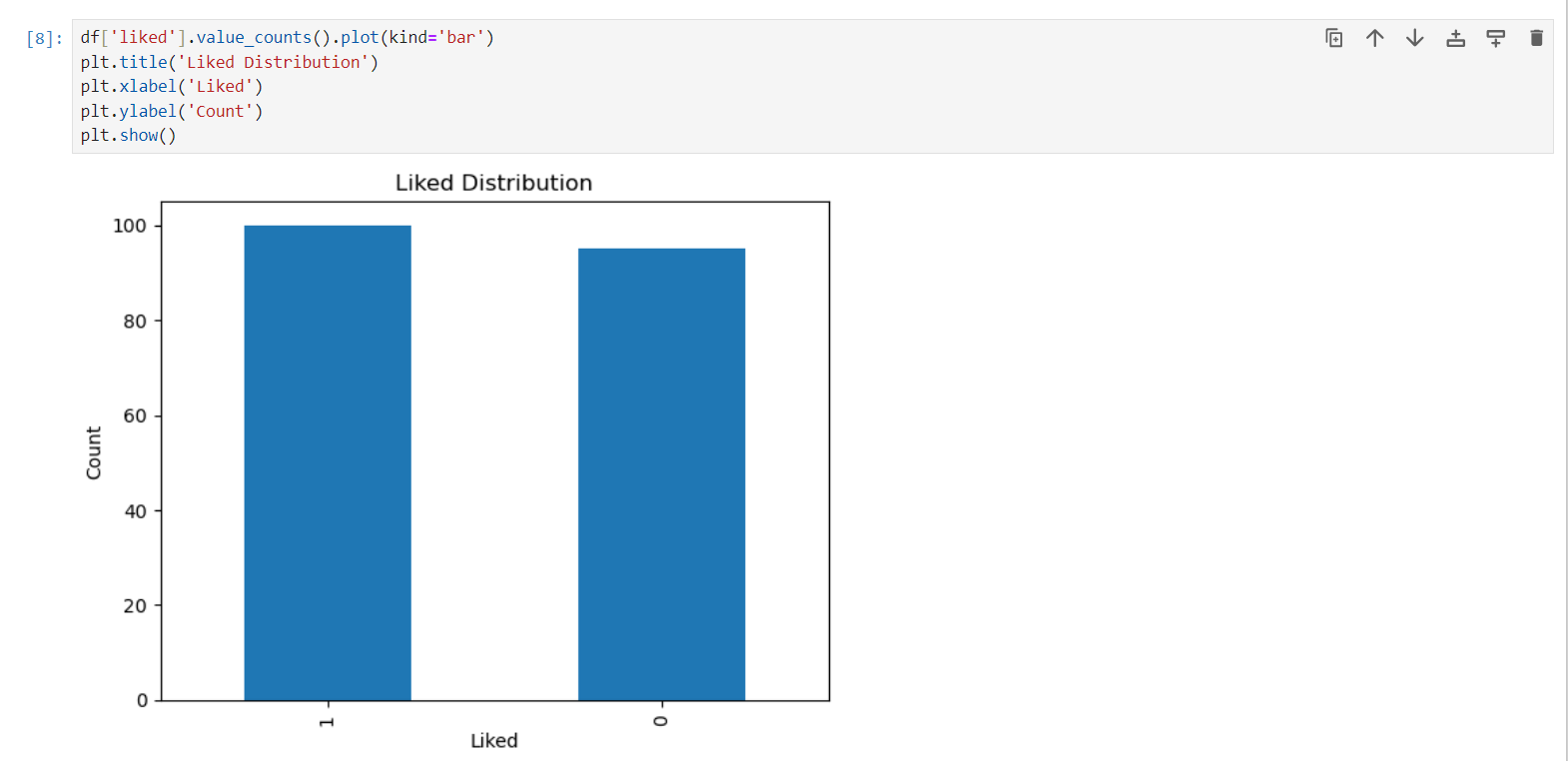


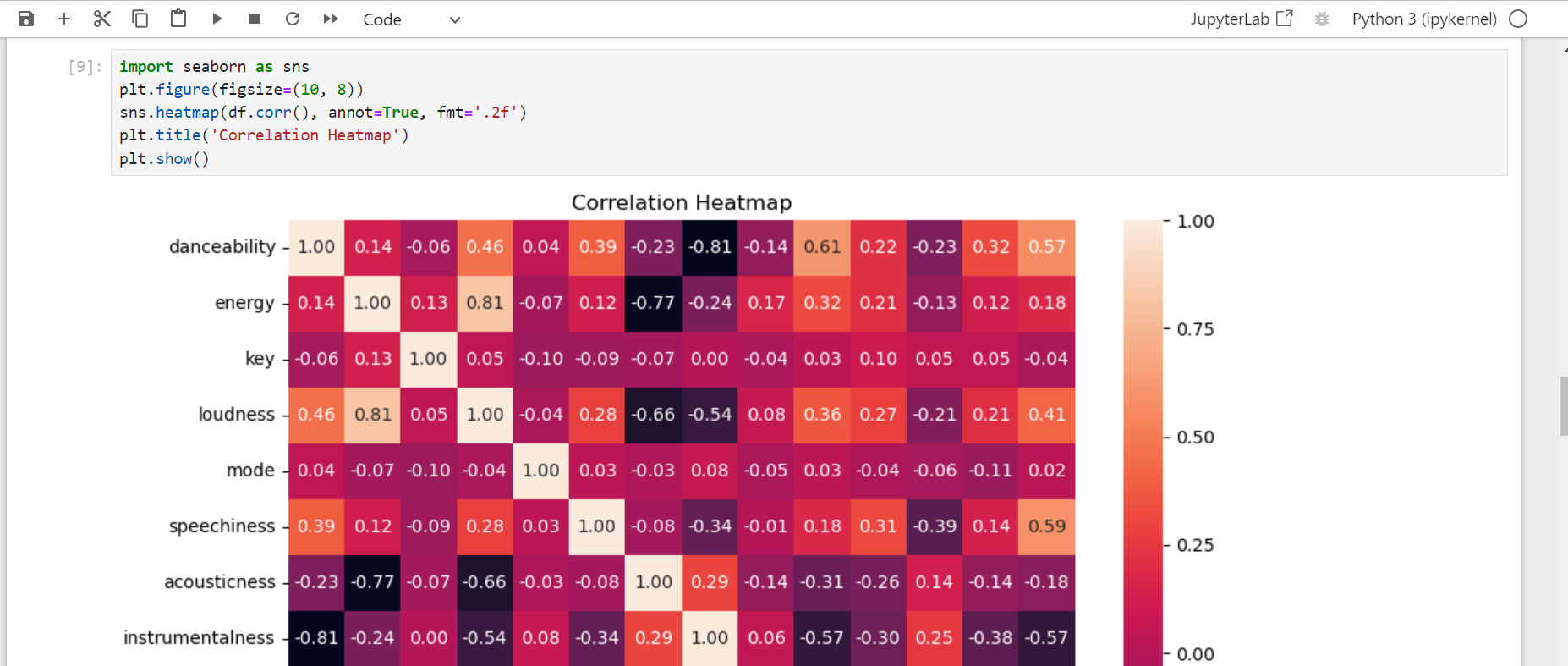
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



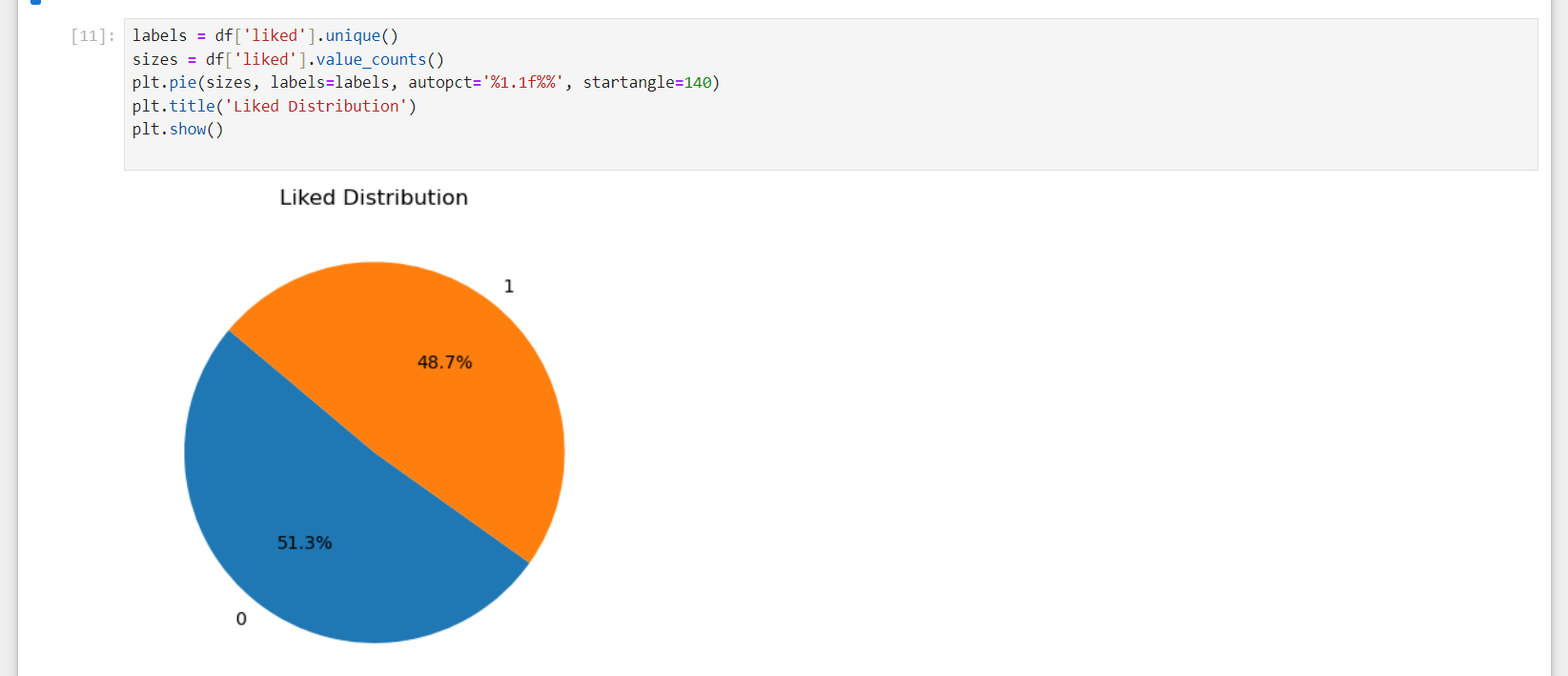


A screenshot of a computer

Description automatically generated

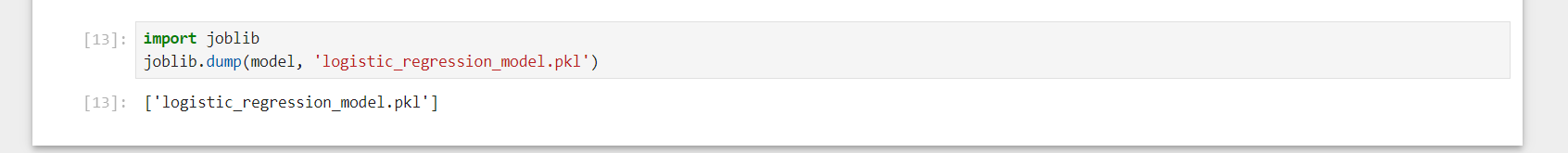
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



# **CONCLUSION**

The Spotify Likeness Classification Analysis project successfully demonstrates the application of logistic regression to predict song likability, offering a significant enhancement to music recommendation systems. By integrating comprehensive datasets that include both audio features of songs and user interaction metrics, the project provides a robust framework for understanding and anticipating user preferences. This approach addresses some of the key limitations of existing recommendation systems, such as limited interpretability and challenges with new users or songs. By leveraging logistic regression, we achieve a balance between model complexity and interpretability, ensuring that the insights generated are both actionable and understandable.

Through meticulous data preprocessing and feature engineering, we have shown that certain audio features—such as tempo, energy, danceability, and acousticness—combined with user interaction data like skip rates and play counts, significantly contribute to predicting whether a user will like a song. The logistic regression model, with its probabilistic output, offers a clear and interpretable method for classification. The model’s performance metrics, including accuracy, precision, recall, and F1-score, indicate its reliability and effectiveness in real-world applications.

The advantages of using logistic regression in this context are manifold. Its simplicity and efficiency allow for real-time implementation in a dynamic environment like Spotify, where user preferences continually evolve, and new songs are added regularly. Moreover, the model’s interpretability provides valuable insights into the factors that most influence user preferences, which can be leveraged to refine recommendation strategies further and improve user satisfaction. This transparency is particularly beneficial for stakeholders who need to understand the decision-making process behind song recommendations.

In conclusion, the Spotify Likeness Classification Analysis project not only enhances the accuracy and personalization of music recommendations but also contributes to a deeper understanding of user preferences in the music streaming domain. By addressing existing system drawbacks and implementing a logistic regression-based solution, the project paves the way for more effective and user-centric recommendation systems. Future work could explore the integration of additional data sources and more sophisticated machine learning techniques to further refine and enhance the model’s predictive capabilities, ensuring that Spotify continues to deliver highly personalized and engaging music experiences to its users.

# **BIBLIOGRAPHY**

1. **Aggarwal, C. C. (2016).** *Recommender Systems: The Textbook.* Springer.
2. **Hastie, T., Tibshirani, R., & Friedman, J. (2009).** *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer.
3. **Spotify. (2021).** *Spotify Developer Documentation.*
4. **Pedregosa, F., et al. (2011).** Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.