

SPOTIFY MUSIC RECOMMENDATION SYSTEM

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

MAHENDRA DATTA

BY

ANUSIKA RANI



**SCHOOL OF ENGINEERING & I.T,
ARKA JAIN UNIVERSITY, JAMSHEDPUR
2020-2024**

In association with



(ISO9001:2015)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

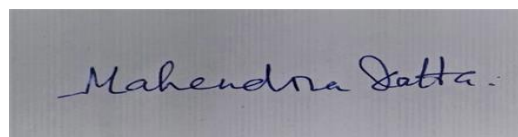
1. Title of the Project: **SPOTIFY MUSIC
RECOMMENDATION SYSTEM**
2. Project Members: **ANUSIKA RANI**
3. Name of the guide: **Mr. MAHENDRA DATTA**
4. Address: Ardent Computech Pvt. Ltd
(An ISO 9001:2015 Certified)
SDF Building, Module #132, Ground Floor, Salt
Lake City, GP Block, Sector V, Kolkata, West
Bengal, 700091

Project Version Control History

Version	Primary Author	Description of Version	Date Completed
Final	ANUSIKA RANI	Project Report	28 TH JULY,2022

Signature of Team Member

Signature of Approver



Date: 28/07/22

Date: 28/07/22

For Office Use Only

MR. MAHENDRA DATTA

Approved

Not Approved

Project Proposal Evaluator

DECLARATION

I hereby declare that the project work being presented in the project proposal entitled “**SPOTIFY MUSIC RECOMMENDATION SYSTEM**” in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALLAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **MR. MAHENDRA DATTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:28/07/22

Registration no.:AJU/201336

Name of the Student: ANUSIKA RANI

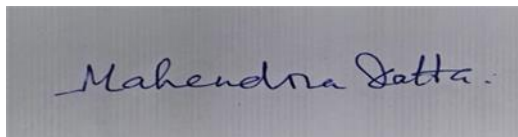
Signature of the students:

CERTIFICATE

This is to certify that this proposal of minor project entitled “**SPOTIFY MUSIC RECOMMENDATION SYSTEM**” is a record of bonafide work, carried out by *ANUSIKA RANI* under my guidance at **ARDENT COMPUTECH PVT LTD**, submitted to **ARKA JAIN University, Jharkhand**.

In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®**. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor



MR. MAHENDRA DATTA

Project Engineer

Assistant Dean

MR. ASHWINI KUMAR

School of Engineering & IT

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to **Mr. MAHENDRA DATTA**, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

- Overview
- History of Python
- Environment Setup
- Basic Syntax
- Variable Types
- Functions
- Modules
- Packages
- Artificial Intelligence
 - Deep Learning
 - Neural Networks
 - Machine Learning
- Machine Learning
 - Supervised and Unsupervised Learning
 - NumPy
 - SciPy
 - Scikit-learn
 - Pandas
 - Regression Analysis
 - Matplotlib
 - Clustering
- SPOTIFY MUSIC RECOMMENDATION SYSTEM
 1. code
 2. conclusion

3. Future Scope
4. bibliography

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic datatypes and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

*If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");***

However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print

continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
    print "True"
else:
    print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file [-]][arg]...
```

Options and arguments (and corresponding environment variables):

- c cmd: program passed in as string(terminates option list)
- d : debug output from parser (also PYTHONDEBUG=x)
- E : ignore environment variables (such as PYTHONPATH)
- h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10          # An integer assignment
weight=10.60        # A floating point
name="Ardent"       # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types –

- ☐ String
- ☐ List
- ☐ Tuple
- ☐ Dictionary

□ Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	int(x [,base]) Converts x to an integer. base specifies the base if x is a string
2	long(x [,base]) Converts x to a long integer. base specifies the base if x is a string.
3	float(x) Converts x to a floating-point number.
4	complex(real [,imag]) Creates a complex number.
5	str(x) Converts object x to a string representation.
6	repr(x) Converts object x to an expression string.
7	eval(str) Evaluates a string and returns an object.
8	tuple(s) Converts s to a tuple.
9	list(s) Converts s to a list.

FUNCTIONS

Defining a Function

- `def function name(parameters):`
 `"function_docstring"`
 `function suite`
 `return [expression]`

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

Function definition is here

```
def change me(mylist):  
    "This changes a passed list into this function"  
    mylist.append([1,2,3,4]);  
    print"Values inside the function: ",mylist  
    return
```

Now you can call changeme function

```
mylist=[10,20,30];  
change me(mylist);  
print" Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]  
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                # This is global variable.
```

Function definition is here

```
def sum( arg1, arg2 ):
```

```
# Add both the parameters and return them."
```

```
total= arg1 + arg2;    # Here total is local variable.
```

```
print"Inside the function local total: ", total
```

```
return total;
```

```
# Now you can call sum function
```

```
sum(10,20);
```

```
Print"Outside the function global total: ", total
```

When the above code is executed, it produces the following result –

Inside the function local total: 30

Outside the function global total: 0

MODULES

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.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, *support.py*

```
def print_func( par ):  
    print"Hello : ", par  
    return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
Import module1 [, module2 [... moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
def Pots ():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function *Isdn ()*
- *Phone/G3.py* file having function *G3 ()*

Now, create one more file *__init__.py* in *Phone* directory –

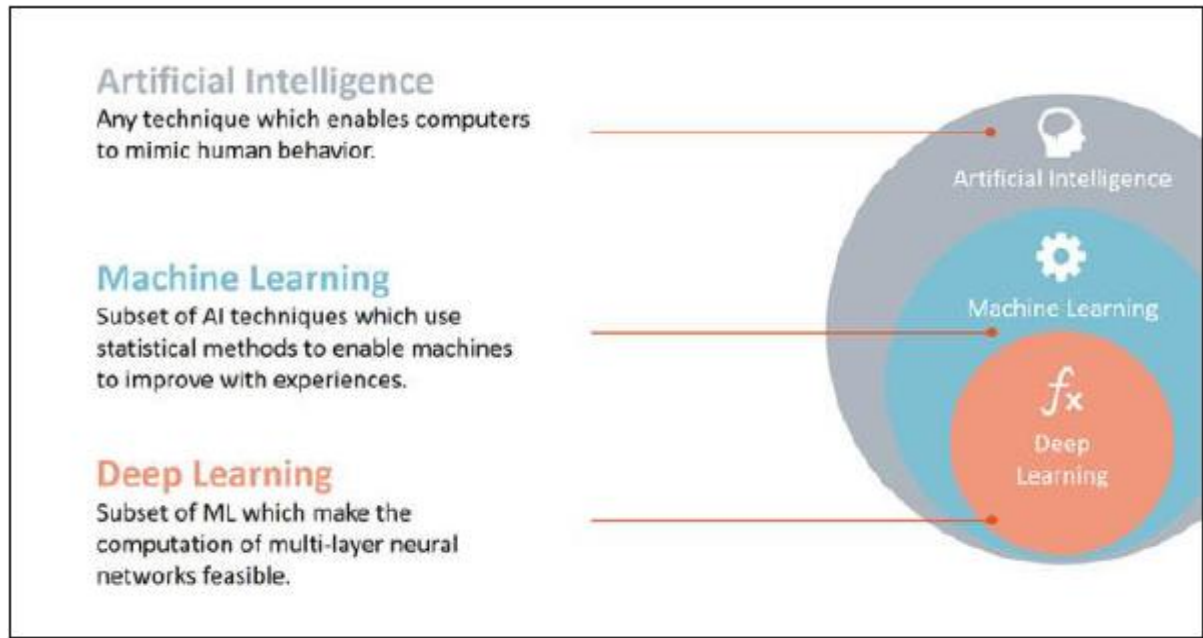
- *Phone/__init__.py*

To make all of your functions available when you've imported *Phone*, you need to put explicit import statements in *__init__.py* as follows –

```
from Pots import Pots  
from Isdn import Isdn  
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is “*The science and engineering of making intelligent machines, especially intelligent computer programs*”.

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems – The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.

To Implement Human Intelligence in Machines – Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as:-

Gaming – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing – It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems – There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems – These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information

Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

Speech Recognition – Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots – Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

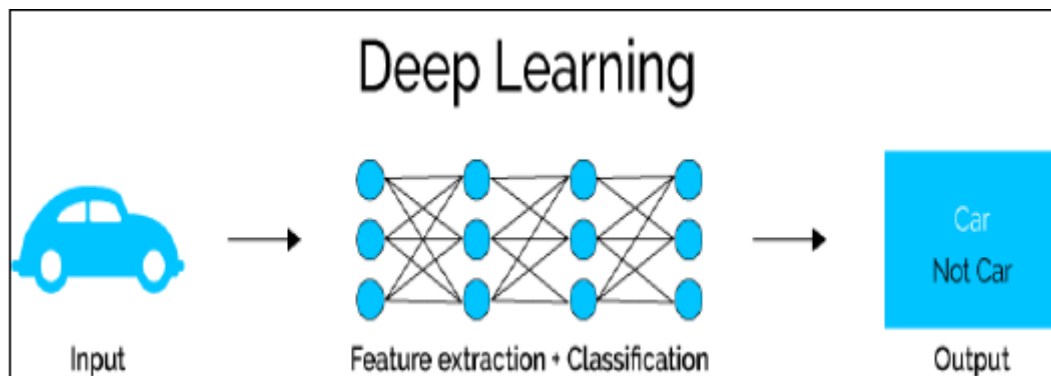
Application of AI

Deep Learning

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

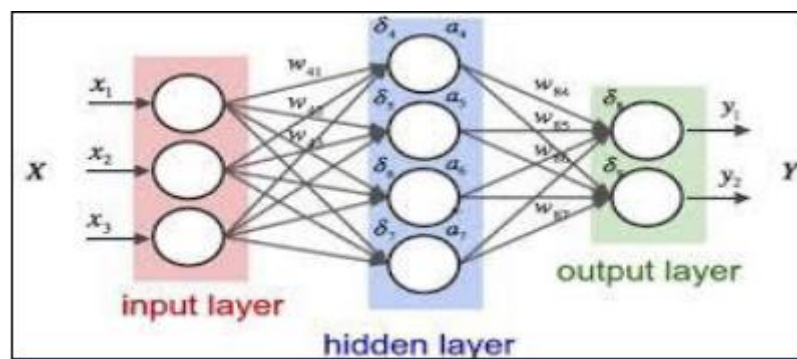
Deep learning is a class of machine learning algorithms that:

- Use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- Learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- Learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- Use some form of gradient descent for training via backpropagation.



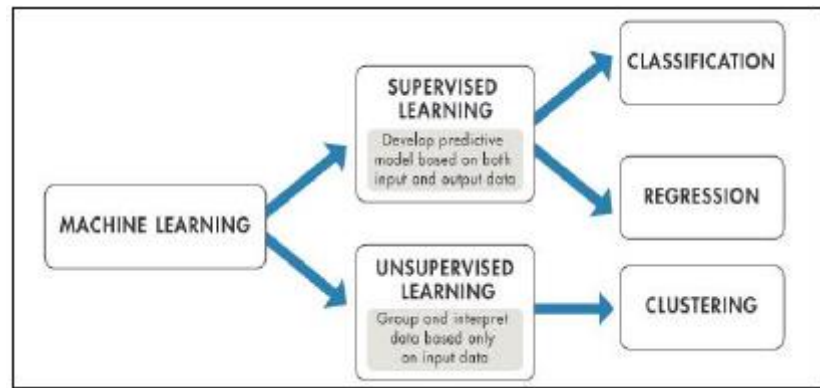
NEURAL NETWORKING

Artificial neural networks (ANNs) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance on) tasks by considering examples, generally without task-specific programming



An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another.

MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.^[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],  
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

Import numpy as np

```
a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])
```

```
print 'Our array is:'
```

```
Print a
```

```
print '\n'
```

```
print 'The items in the second column are:'
```

```
print a[:,1]
```

```
print '\n'
```

```
print 'The items in the second row are:'
```

```
print a[1...]
```

```
print '\n'
```

```
print 'The items columns 1 onwards are:'
```

```
print a[:,1:]
```

OUTPUT

Our array is:

```
[[1 2 3]
```

```
[3 4 5]
```

```
[4 5 6]]
```

The items in the second column are:

```
[2 4 5]
```

The items in the second row are:

```
[3 4 5]
```

The items column 1 onwards are:

```
[[2 3]
```


[4 5]
[5 6]]

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

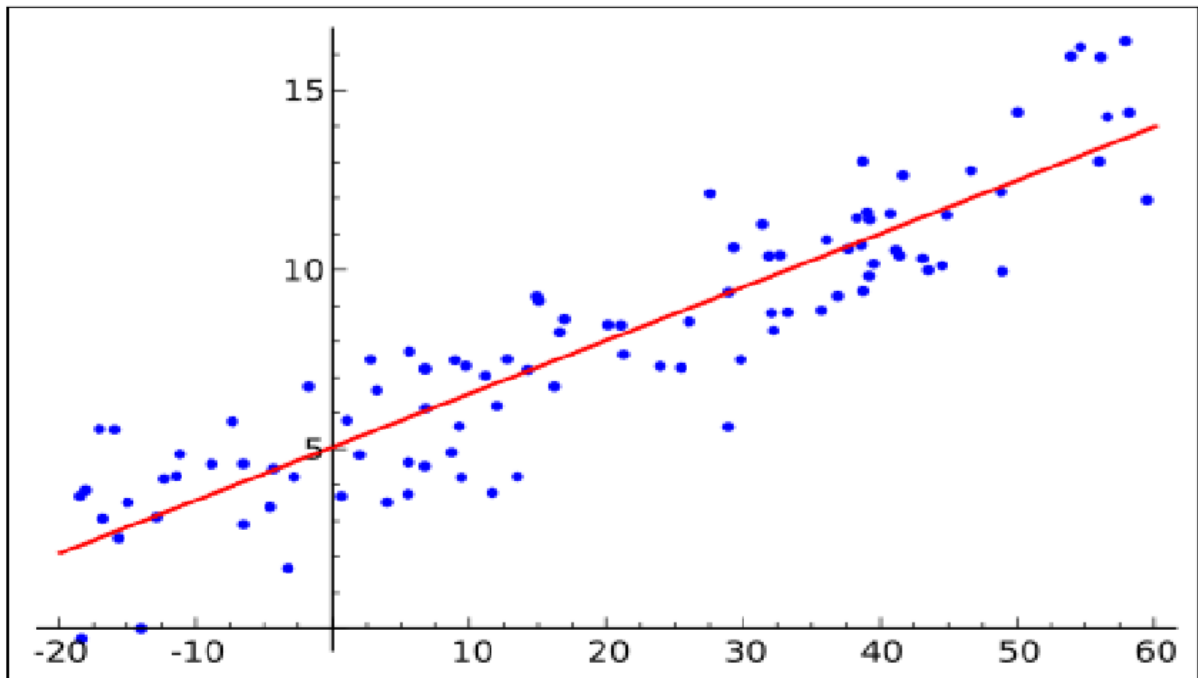
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-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 Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model^[1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x .

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted $E(y | x)$, and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function $E(y | x)$ is linear in the unknown parameters that are estimated from the data.

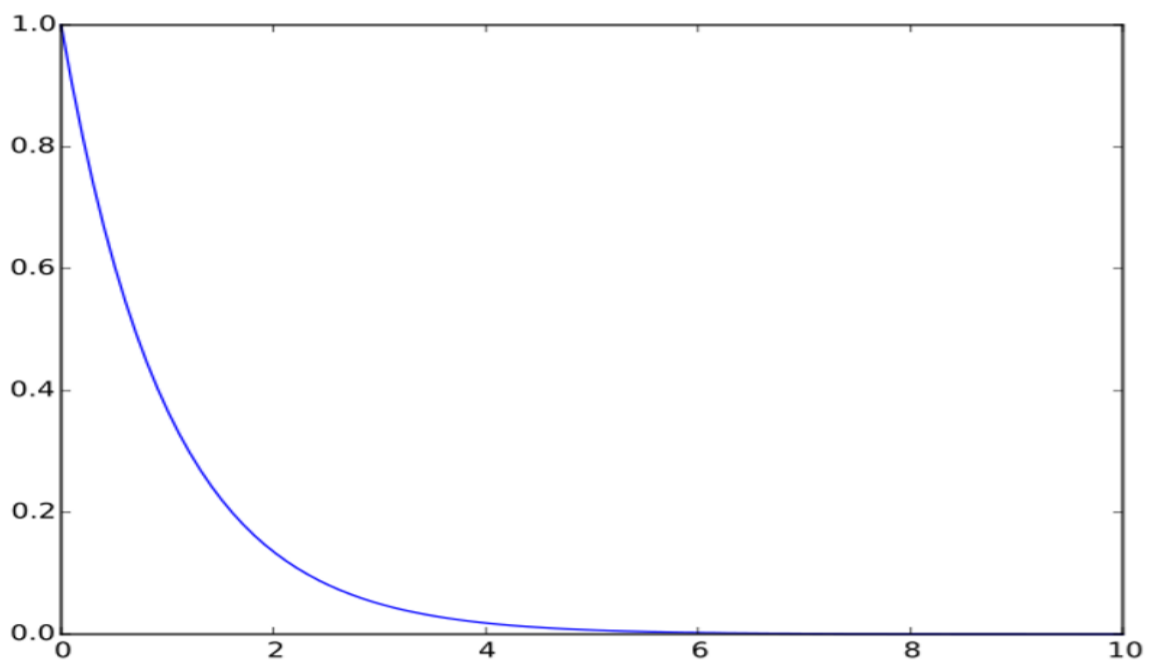
MATPLOTLIB

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.

EXAMPLE

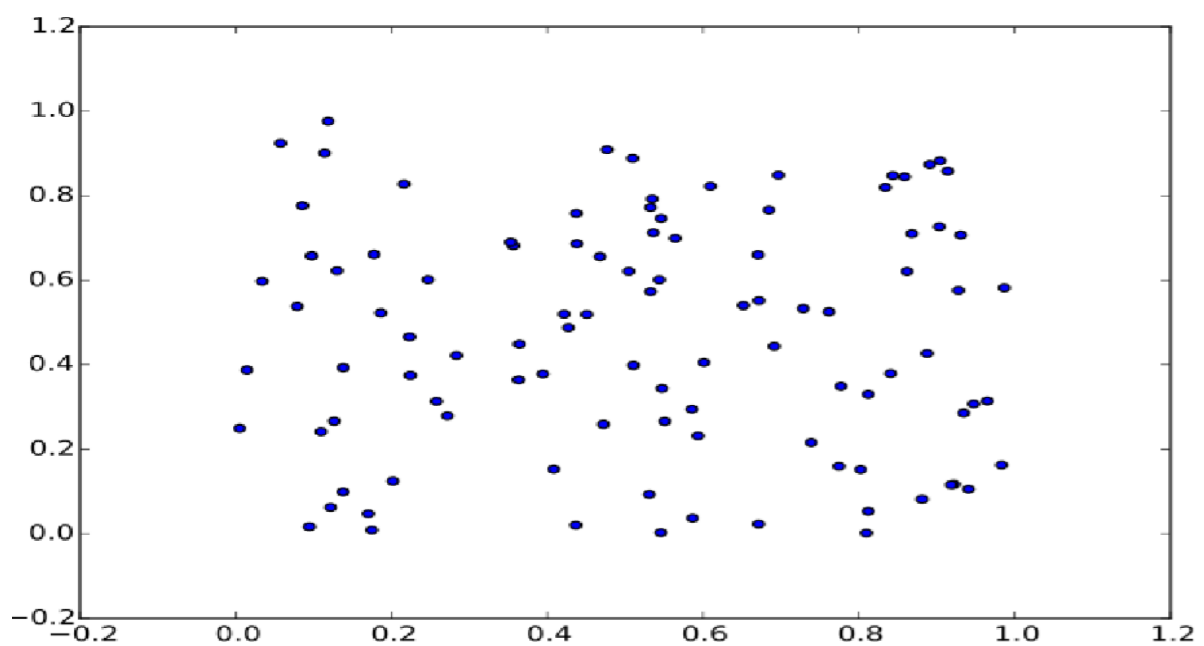
➤ LINE PLOT

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot (a,b)
>>>plt.show ()
```



➤ **SCATTER PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimportrand
>>>a=rand(100)
>>>b=rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Lasso Linear Model and Bayesian Ridge Linear Model.

Training: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

SPOTIFY MUSIC RECOMMENDATION SYSTEM

CODE AND EXPLANATION

Music recommender system

One of the most used machine learning algorithms is recommendation systems. A **recommender** (or recommendation) **system** (or engine) is a filtering system which aim is to predict a rating or preference a user would give to an item, eg. a film, a product, a song, etc.

Which type of recommender can we have?

There are two main types of recommender systems:

- Content-based filters
- Collaborative filters

Content-based filters predicts what a user likes based on what that particular user has liked in the past. On the other hand, collaborative-based filters predict what a user like based on what other users, that are similar to that particular user, have liked.

Import Libraries

```
import numpy as np
import pandas as pd
from typing import List, Dict
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Dataset

```
songs = pd.read_csv('songdata.csv')
songs.head()
```

This dataset contains name, artist, and lyrics for 57650 songs in English.

output: -

	artist	song	link	text
0	ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face \nAnd...
1	ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please \nTouch me gentl...

	artist	song	link	text
2	ABBA	As Good As New	/a/abba/as+good+as+new_20003033.html	I'll never know why I had to go \nWhy I had t...
3	ABBA	Bang	/a/abba/bang_20598415.html	Making somebody happy is a question of give an...
4	ABBA	Bang-A-Boomerang	/a/abba/bang+a+boomerang_20002668.html	Making somebody happy is a question of give an...

Because of the dataset being so big, we are going to resample only 5000 random songs.

```
songs = songs.sample(n=5000).drop('link', axis=1).reset_index(drop=True)
```

We can notice also the presence of \n in the text, so we are going to remove it.

```
songs['text'] = songs['text'].str.replace(r'\n', '')
```

After that, we use TF-IDF vectorizer that calculates the TF-IDF score for each song lyric, word-by-word.

Here, we pay particular attention to the arguments we can specify.

```
tfidf = TfidfVectorizer(analyzer='word', stop_words='english')
```

```
lyrics_matrix = tfidf.fit_transform(songs['text'])
```

How do we use this matrix for a recommendation?

We now need to calculate the similarity of one lyric to another. We are going to use **cosine similarity**.

We want to calculate the cosine similarity of each item with every other item in the dataset. So we just pass the lyrics matrix as argument.

```
cosine_similarities = cosine_similarity(lyrics_matrix)
```

Once we get the similarities, we'll store in a dictionary the names of the 50 most similar songs for each song in our dataset.

```
similarities = {}
```

```
for i in range(len(cosine_similarities)):
```

```
    # Now we'll sort each element in cosine_similarities and get the indexes of the songs.
```

```
    similar_indices = cosine_similarities[i].argsort()[::-50:-1]
```


After that, we'll store in similarities each name of the 50 most similar songs.

Except the first one that is the same song.

```
similarities[songs['song'].iloc[i]] = [(cosine_similarities[i][x], songs['song'][x],  
songs['artist'][x]) for x in similar_indices][1:]
```

We can use that similarity scores to access the most similar items and give a recommendation.

For that, we'll define our Content based recommender class.

```
class ContentBasedRecommender:
```

```
    def __init__(self, matrix):
```

```
        self.matrix_similar = matrix
```

```
    def _print_message(self, song, recom_song):
```

```
        rec_items = len(recom_song)
```

```
        print(f'The {rec_items} recommended songs for {song} are:')
```

```
        for i in range(rec_items):
```

```
            print(f"Number {i+1}:")
```

```
            print(f"{recom_song[i][1]} by {recom_song[i][2]} with {round(recom_song[i][0], 3)}  
similarity score")
```

```
            print("-----")
```

```
    def recommend(self, recommendation):
```

```
        # Get song to find recommendations for
```

```
        song = recommendation['song']
```

```
        # Get number of songs to recommend
```

```
        number_songs = recommendation['number_songs']
```

```
        # Get the number of songs most similars from matrix similarities
```

```
        recom_song = self.matrix_similar[song][:number_songs]
```

```
        # print each item
```

```
        self._print_message(song=song, recom_song=recom_song)
```

Now, instantiate class

```
recommendations = ContentBasedRecommender(similarities)
```

Then, we are ready to pick a song from the dataset and make a recommendation.

```
recommendation = {  
    "song": songs['song'].iloc[10],  
    "number_songs": 4  
}  
  
recommendations.recommend(recommendation)
```

output: -

```
The 4 recommended songs for John Hardy are:  
Number 1:  
Dear John by Kirsty Maccoll with 0.271 similarity score  
-----  
Number 2:  
Dear John by Styx with 0.264 similarity score  
-----  
Number 3:  
Dear John by Hank Williams with 0.188 similarity score  
-----  
Number 4:  
Jaw, Knee, Music by NOFX with 0.162 similarity score  
-----
```

And we can pick another random song and recommend again:

```
recommendation2 = {  
    "song": songs['song'].iloc[120],  
    "number_songs": 4  
}  
  
recommendations.recommend(recommendation2)
```

output: -

```
The 4 recommended songs for I'm Not Like Everybody Else are:  
Number 1:  
Violent Pornography by System Of A Down with 0.664 similarity score  
-----  
Number 2:  
Everybody's Trying To Be My Baby by Johnny Cash with 0.543 similarity score  
-----  
Number 3:  
Summertime Blues by Eddie Cochran with 0.529 similarity score  
-----  
Number 4:  
Everybody Wants You by Ice Cube with 0.451 similarity score  
-----
```

CONCLUSION

We have collected the raw data from online sources. Then we take this raw data and format it.

We have first extracted the data according to the features we need . a lyrics matrix is created by using the extracted data . here in this algorithm cosine similarity is used. Once we got the similarities, we have stored in a dictionary the names of the 50 most similar songs for each song in our dataset.

We can use that similarity scores to access the most similar items and give a recommendation.

FUTURE SCOPE

The data taken was limited. The project could be extended to more number of songs. The music was recommended only by taking the limited list of song data in account, it can be predicted by looking at the bigger set of data as well.

The error can be minimized as well using other algorithms.

BIBLIOGRAPHY:

the csv files used for data extraction was obtained from Kaggle

THANK YOU