DATA COMPRESSION TECHNIQUES PREDICTION USING MACHINE LEARNING

A PROJECT REPORT

ABSTRACT

Data compression is the conversion of a file into another file of smaller size than the original file size. The compression of data is used to conserve storage space, to speed communication. More focus needs to be put on data compression. There are several types of data compression algorithms that each have different properties. Some of these algorithms are best suited for a specific filetype or size. Different data compression algorithms give different efficiency when applied to a same file. Hence choosing the best machine learning algorithm for a given file is very important.

In this project we are applying machine learning algorithm to predict which data compression algorithm is best suited for a given file. In this project we use three different types of machine learning algorithms these are Huffman coding, LZW, Shannon Fano. The project helps us to determine which out of these algorithms can be used to get an efficient compression of the file.

CHAPTER 1
INTRODUCTION

1.1 Overview

Data compression is used to modify, encode or decode the data in such a way that it reduces the storage space of the file in the system. This is a compression technique which is used to reduce the size of data by compressing it using compression algorithms. There are two different types of data one is Lossy and another one is Lossless. Regarding our project, in a data compression process the original text file is converted into a compressed file, such that the original text file can be completely recovered from the compressed file by decompressing it. The compressed text file will be recovered without any loss of original information contained in the file. This process is more efficient if you want to save the storage space in your device. For example, if am storing a 50MB file, it is preferable to first compress it into a smaller size file and then to save the file. These compressed files can be easily exchanged from one person to another person over the internet since they upload and download much faster than the original file. It is useful to transfer a text file without any lack in the transmission. It will also reduce the transmission cost. And, we can also decompress the original file from the compressed file at any time. So, in this project we have aimed to reduce the storage size and increase the transmission speed of a text file by compressing the original file and then by decompressing the compressed file.

1.2 Objective

Data compression is one of the important feature in the area of file storage and transmission. Our objective is to build a data compression process by using machine learning techniques. In this project, we have discussed about several techniques for compressing and predicting the suitable compression technique for the given text files.

1.3 Problem Statement

The major problem in the existing data compression techniques is it depends on the third-party software's. And, it takes more time to compress a file. It also increases the complexity in transferring the file from one person to another person. Some other disadvantages in existing system are lack of data security, loss in data accuracy, consuming more time, etc. So, we have worked on different data compression algorithms to improve the data compression technique.

1.4 Research Objectives

This research work is based on reducing the size of a text file by compressing it into a smaller size file without any loss of original data in the file. The primary idea behind this is to create self-awareness among people on storage consumption and provide them with ideas and suggestions to cut down the usage of unnecessary storage space.

CHAPTER 2

LITERATURE STUDY

2.1 Reducing Symbol Search Operations on Stream

This paper [1] provides a detailed overview of dictionary-based lossless data compression algorithms precisely replace the standard input data pattern with a compressed symbol, and also to decompress.

Drawback:

Data compression mechanism treats data streams with very low latency to avoid overhead in the path

Proposed system advantages:

The proposed system overcomes the challenges of improving and updating the compression algorithms regularly.

2.2 Next Generation Sequencing Data

This paper [2] explains that the large number of research papers and compression algorithms proposed for the compressing of data generated by sequencing machines, gzip is by far the most commonly used compression algorithm in the industry.

Drawback:

The proposed compression algorithms are the slow speed of either compression or decompression where it takes more time than the usual time needed for the compression or decompression of the data.

Proposed system advantages:

In this system we propose achieving speed improvements and 6%-11% more efficient compression ratios for the compression and decompression.

2.3 Compression Using Recurrent Neural Networks

Neural networks are known as the universal function approximators with the ability to learn arbitrarily complex mappings and display excellent performance in predicting a task, we use neural network predictors to improve the methods for compressing sequential data. We combine recurring neural network predictors with an arithmetic coder and compress wide variety of digital documents and genomic datasets without any loss. [3]

Drawback:

It looks up for the efficient compression mechanisms to enable better storage, transmission and processing of such data.

Proposed system advantages:

This proposed system provides a better solution for storage, transmission and processing of the data.

2.4 Optimize Genomics Data Compression

In this paper [4] we discuss the definition of data compression in the workflows of genomics and evaluate the efficiency & cost of different compression algorithms in the tool of genomics analysis, we present a new method of hardware acceleration that efficiently compresses DNA sequences with reduced computation time and CPU usage.

Drawback:

Data management have become one of the most challenging task in data compression techniques. This paper fails to maintain a proper record on the input data taken for the compression which may lead to loss of data accuracy.

Proposed system advantages:

Our proposed system aims to improve the efficiency of data accuracy and storage for the betterment of users.

2.5 Fractal Image Compression Method

In this paper [5] we are discussing about the JPEG compression method is one of the most widely used methods for compression of lossy images today. A system for compression of fractal images is proposed to address the disadvantages. JPEG compression provides decent results on small sized images but it suffers from big sized images. But the process of fractal compression prevents these disorders. As the size of the images increases, the method of fractal compression is observed to be more suitable.

Drawback:

As this paper says, when the size of the image increases, both accuracy and compression ratio decrease and it takes a long time for decoding.

Proposed system advantages:

This proposed system gives good results on both small-sized and large-sized text documents.

2.6 Accelerate Data Compression in File System

In this paper work [6] we are researching the essence of data compression in BTRFS, setting up the test bench to quantify the gain and cost when enabling data compression in BTRFS, and designing a new method of hardware acceleration to offload the compression and decompression workloads from the

BTRFS software stack. Our experiment shows that the hardware offloading solution is capable of providing BTRFS data compression with the least CPU overhead compared to ZLIB and LZO software implementation, while still achieving a very good compression ratio and disk write performance.

Drawback:

It needs advanced system for compressing and decompressing the data and it consumes more power since the CPU is very intensive.

Proposed system advantages:

The proposed system can work with the basic CPU and it consumes very less amount of power for compression and decompression purposes.

2.7 Compression technique with encryption-based method

This paper [7] proposes a new technique of compression and compares its efficiency to the common Huffman adaptive encoding. The main goal of our approach was to optimally shrink the original data without losing data integrity. We used compression ratio, compression time, compression factor and a percentage saving for comparison and analysis. A Windows executable framework was designed to encourage and ease the use of the program. Alternatively, adding an encryption algorithm (Caesar Cipher) would ensure the security and privacy of files. This compression can handle large files quickly, and without wasting much of the resources of the machine.

Drawback:

This system compares the file with only one algorithm so that the system will not provide efficient compression technique.

Proposed system advantages:

Where as in our proposed system we compare the file between three algorithms to find which one is more efficient for the given type of file.

2.8 Visualization-Oriented Trajectory Method

This paper [8] proposes a visualization-oriented trajectory data compression method to overcome the problem of current trajectory data compression methods overly follow the optimum trajectory data compression rate which makes it difficult to maintain the similar shape between the compressed trajectory and the original trajectory.

Drawback:

In this system the originality of the file may change comparing between the compressed file and original file.

Proposed system advantages:

In our proposed system it provides the exact data in the file even after compressing the original file. Data accuracy is maintained perfectly in the proposed system comparing to other compression techniques.

2.9 Dynamic Online Performance Optimization

For high-bandwidth applications such as science simulations and sensing applications, compression is necessary to the resource burden such as storage, network transmission, and more recently I/O. This research proposes an online data compression streaming algorithm that takes into account the generally concave pattern of the compression ratio curve, and optimizes key operation parameters. We show that the proposed algorithm successfully adapts one of IDEALEM's main parameters to the optimum value and yields near maximum compression ratios for time series data. [9]

Drawback:

Existing system compression methods minimize the quality between original data and reconstructed data, which either reduces the compression performance or reconstruction quality.

Proposed system advantages:

Our compression technique maximizes the compression performance and provides with the best quality of data even after compressing and decompressing the data.

2.10 Power System Operational Data Compression

Differential Binary Encoding Algorithm (DBEA) and Extended-DBEA (E-DBEA) were the two lossless data compression techniques developed to compress the operational data of the power system and obtained a high compression ratio (CR) for most practical data sets. While for the majority of available data DBEA can offer high compression ratio (CR) than E-DBEA, it fails when transient data sets have been found. [10]

Drawback:

The cost of data compression and data transmission is high in this kind of techniques comparing to other data compression techniques.

Proposed system advantages:

In our proposed system the cost of compression, decompression and transmission of the data is very less comparing to other data compression methods.

CHAPTER 3

3.1 Functional Requirements

The functional requirements for our system are:

- System should be able to fetch and display all the data.
- System should be able to easily installable with fewer modifications to the existing one.
- System should be able to store the compresses data.
- System should be able to calculate the difference between size of the compressed data from original data.
- System should be able to transfer data from one network to another.

3.2 Non-Functional Requirements

The Non-functional Requirements for our system are:

- The system should be made by keeping all privacy laws in mind.
- The system should be scalable.
- Systems should be portable in nature.
- The system should be readily available 24*7.
- Future versions and maintenance should in no way affect the hardware requirement of the existing system.

3.3 HARDWARE REQUIREMENTS

Physical computing resources, often called hardware, are the most basic collection of specifications specified by any os or software program. A checklist

of hardware specifications, particularly in the case of operating systems, is sometimes followed by a list of hardware compatibility. The basic specifications for hardware are as follows,

1. PROCESSOR NAME: PENTIUM IV
2. RAM: 8 GB
3. PROCESSOR: 2.4 GHZ
4. MAIN MEMORY: 8GB RAM
5. PROCESSING SPEED: 600 MHZ
6. HARD DISK DRIVE: 1TB
7. KEYBOARD :104 KEYS

3.4 SOFTWARE REQUIREMENTS

The specifications for applications deal with the concept of resource criteria and prerequisites that ought to be implemented on a device to ensure that a program works. Before the device is installed such specifications must be implemented separately. The basic specifications for apps are as follows,

1. FRONT END: PYTHON
2. IDE: ANACONDA
3. OPERATING SYSTEM: WINDOWS 10

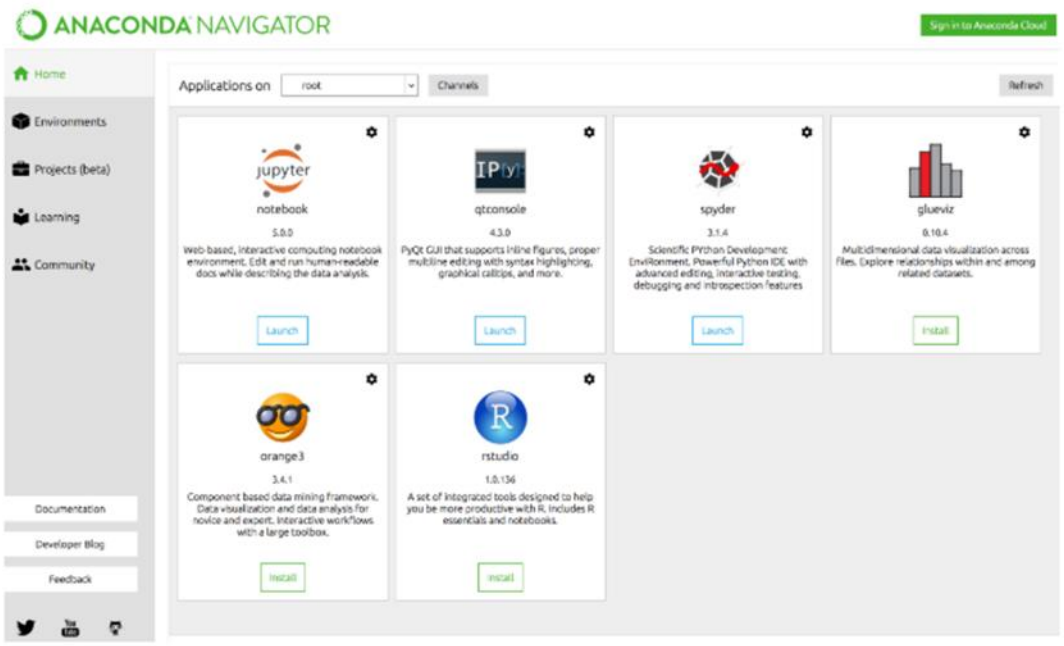
3.4.1 Python Language

It is an object-oriented programming language which Guido Rossum created in 1989. Preferably suitable for fast prototyping of complex applications. This provides interfaces to other operating system calls and collections, which can be expanded to either C or C++. The Python

programming language is used by several major corporations including NASA, Facebook, Twitter, BitTorrent etc. Python programming is commonly utilized in Computer Science in Artificial Intelligence, Natural Language Processing, Neural Networks and other specialized sectors. Python focuses heavily on code readability.

3.4.2 Anaconda Navigator

Anaconda Navigator is a customizable user interface (GUI) for desktops used in the Anaconda release. It helps us to operate applications included in the Anaconda delivery and control conda products, environments and channels quickly without the use of command-line commands. It's available for Windows, Linux, macOS and.



Anaconda Navigator

FIGURE 3.4.1

3.5 SYSTEM DESIGN

3.5.1 SYSTEM ARCHITECTURE

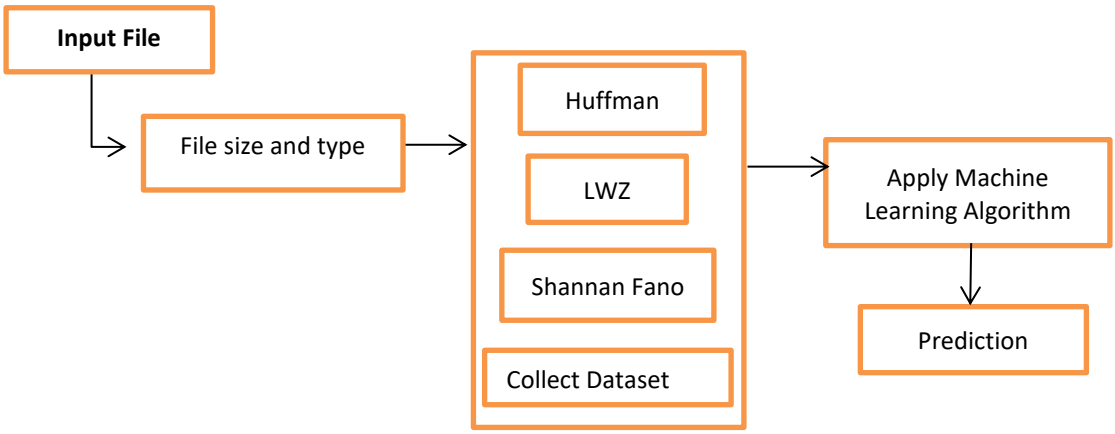


FIGURE 3.5.1

3.5.2 DATA FLOW DIAGRAM

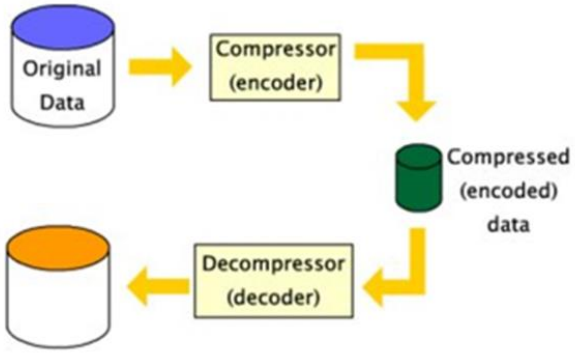


FIGURE 3.5.2

3.6 UML DIAGRAMS

UML is a popular language used to describe, model, create, and document artifacts on the operating system. The Object Management Group (OMG)

developed UML, and in January 1997 the OMG submitted UML 1.0 specification draft.

3.6.1 USE CASE DIAGRAM:

these are a series of use cases, participants and relationships between them. We reflect a perspective of a system's usage case.

A case of use is a basic aspect of a system. Usage case study is then used to describe the application's connection to its internal / external controllers. Those inspections are called performers.

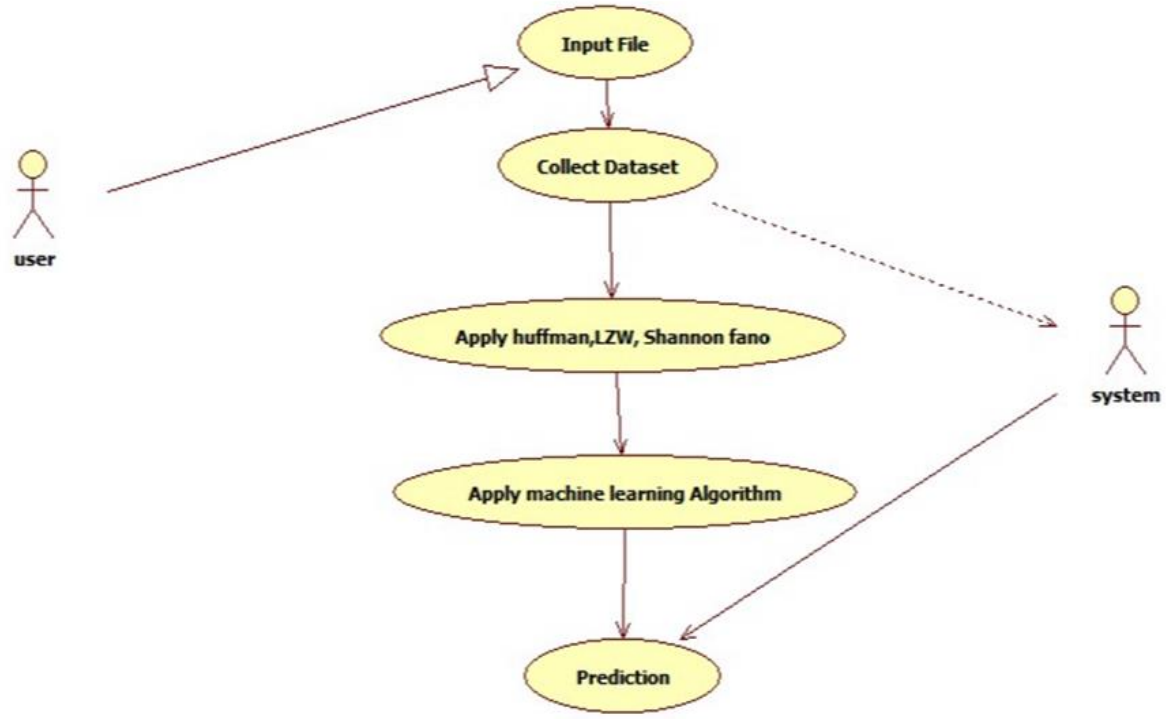


FIGURE 3.6.1

3.6.2 SEQUENCE DIAGRAM:

A series diagram is an overview of an occurrence. From the description it is clear that the diagram deals with such series, which are the collection of communications passing from one person to another.

Contact among the modules of a device is highly important from the implementation and execution point of view. Series diagram is used to depict the number of calls inside a system to conduct a specific function.

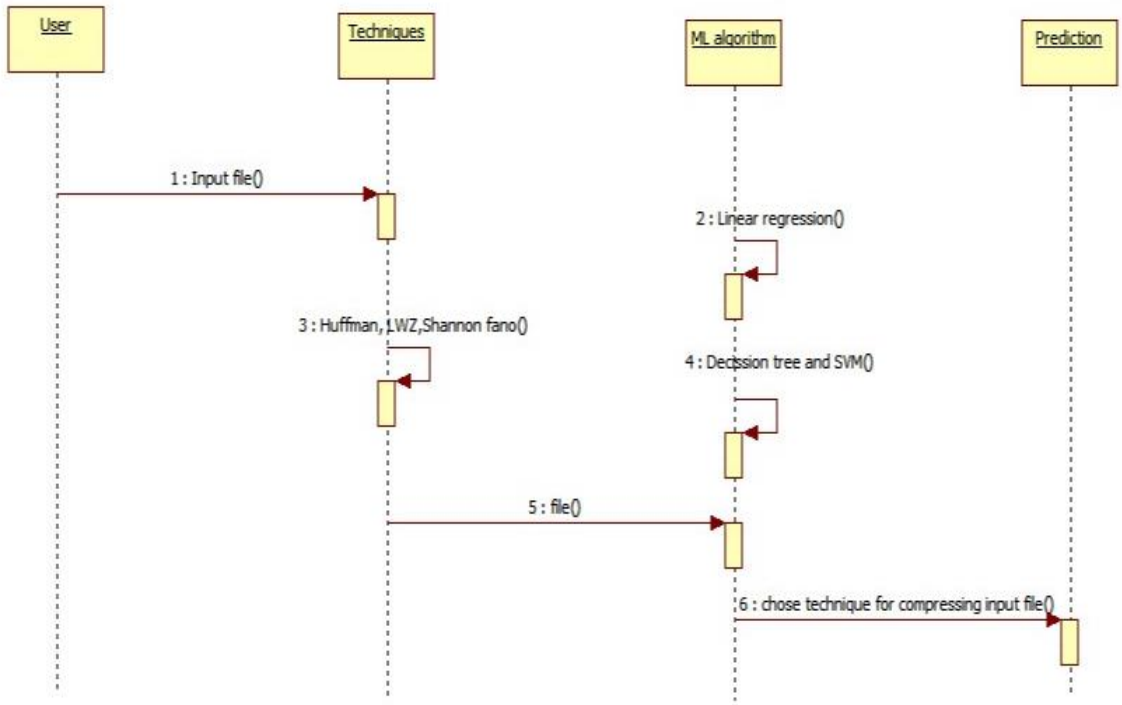


FIGURE 3.6.2

3.6.3 ACTIVITY DIAGRAM:

Action diagram defines power flow within a device. It consists of the events and the relations. The flow may be either continuous, concomitant or branched. Activities are nothing but machine features. Numbers of operation diagrams are able to show the whole process inside a framework. Task diagrams are used for visualizing power flow within a device. It is just to get an understanding about how the program should operate as it's implemented.

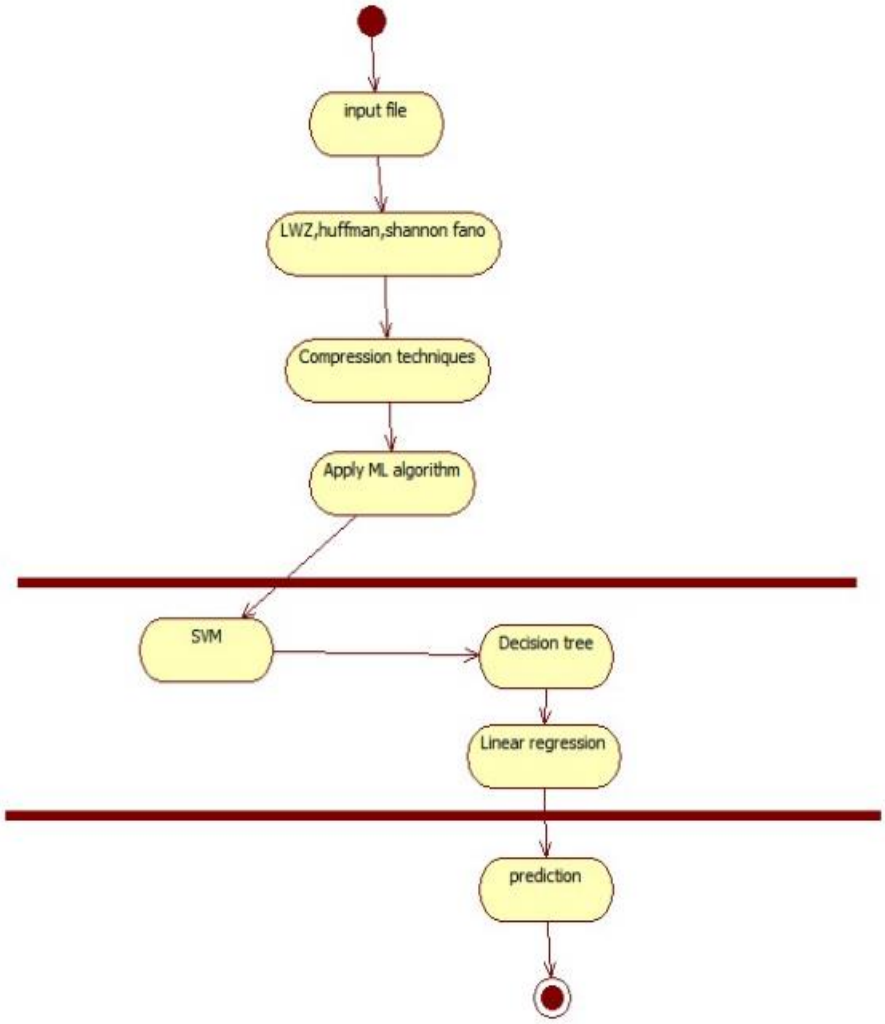


FIGURE 3.6.3

3.6.4 CLASS DIAGRAM:

Group diagrams are the diagrams found most commonly in UML. Class Diagram is comprised of groups, frameworks, partnerships, and partnership. Unit diagrams are simply the object-oriented perspective of a method, which is static in design. Within a hierarchy diagram Active hierarchy is used to reflect the system's competitiveness. Class diagram is the direction of an entity in a scheme. So, it is commonly used for purposes of growth. This is the most commonly known machine design diagram at the moment.

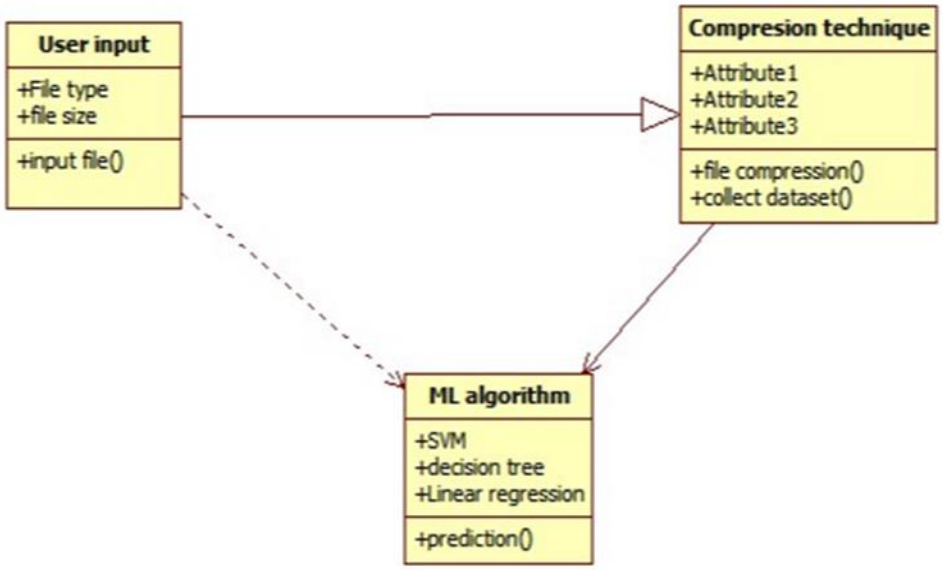


FIGURE 3.6.4

The diagrams above are concept UML diagrams. UML diagrams are prepared to provide a better and more realistic view of the operation. A single diagram isn't adequate to cover all machine components. UML describes various diagram styles which will make up much of a system's elements.

4.4 Machine Learning

4.4.1 linear regression

A linear regression model's goal is to find a relationship among one or more features (autonomous variables) and a continuous target (dependent variable). This is called Uni-variate Linear Regression when there is just a feature and if there are multiple aspects it is called Multiple Linear Regression.

Simple linear regression is a method to estimating an outcome using a single function.

The two variables are considered to be related linearly. Therefore, as a feature of an attribute / independence variable, we seek to find as reliable as possible a mathematical formula which forecasts the result value.

Let's call the dataset where for each feature x we have an answer value y:

TABLE 4.4.1

x	0	1	2	3	4	5	6	7	8	9
y	1	3	2	5	7	8	8	9	10	12

In general terms, we describe: X as just a function vector, i.e. x = [x 1, x 2, ..., x n], Y as an answer vector, i.e. y = [y 1, y 2, ..., y n] of n occurrences (in the illustration before, n=10).

the scatter plot for the table is: -

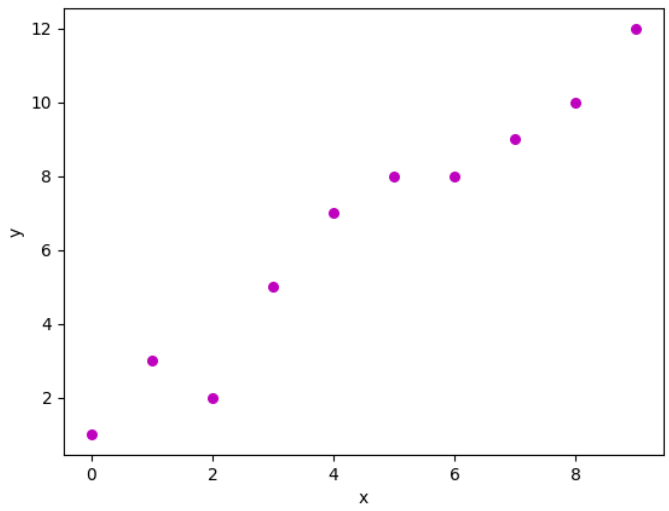


FIGURE 4.4.1

Still, the challenge is finding a line that best suits well into the scatter plot above, so that we can foresee the explanation to any new product values. (i.e. x valuation not seem portray in the sample) It is considered a line of regression. The equation to the regression line as replicated as follows:

$H(x_i) = \beta_0 + \beta_1 x_i$

Here, H(xi)reflects the expected reaction value for observation of ith.

β0 and β1are coefficients of regression and represent y-intercept and regression line slope, respectively.

4.4.2 SVM

In machine learning, support vector machines (SVM, often support vector networks) are controlled training systems with similar optimization algorithms that represent data used instead of identification and correlation analysis. The SVM learning method builds a template which assigns fresh instances to one or other subset, rendering it an un-probabilistic conditional vector classifier, offering a set of learning cases, each of which is classified as having connections to one or the other category two. The SVM template is indeed a depiction of the cases as space points, mapped to differentiate the instances in various categories by a simple distance which is as large as possible. Then fresh instances were inserted onto the same room but are supposed to correspond to a class based on the aspect of the vacuum into which they have sunk.

4.4.3 DECISION TREE

A decision tree is by far the most powerful and growing approach for classifying and forecasting. The Decision tree may be a leaf-like block diagram where each inner node is an entity check, every branch is a test result, so each branch node (term node) is a category name.

Development of Algorithm:

A tree is "learned" via splitting its root set into subgroups that depend on a measure of the attribute's value. This loop, dubbed lexical partitioning subsection, is duplicated in a nested way. The recursion becomes done once the node subset either has the same target variable value, or the projections are no longer given meaning by separating. Developing a decision tree classifier needs no expertise or requirements definition, and is therefore ideal for exploring explorative details. Decision trees can accommodate data of the large dimensions. Tree classifier has excellent consistency in general decision taking. Decision tree inference is a standard inductive method for studying classification information.

Decision Tree Representation:

Decision trees classify cases by ordering it from either the core to every node vertex down the ladder providing classification for the case. An example is described by starting at the root node of the tree, testing the attribute specified by that node, then moving down the division corresponding to the value of the attribute. Afterwards this process is repeated for subtree inserted in the new node.

For example,

In a case the leftmost branch of the decision tree will be arranged and then counted as opposite. In another terms it can be assumed the decision tree reflects a disconnection on the attribute values of instances on the conjunctions of constraints.

```
0.4386422976501306
[[168   0   0]
 [146   0   0]
 [ 69   0   0]]

enter input file size:
```

```
0.4386422976501306
[[168   0   0]
 [146   0   0]
 [ 69   0   0]]
enter input file size: 13.2

enter file type: (0-txt,1-img,2-pdf,3-docx) 0
```

```
0.4386422976501306
[[168   0   0]
 [146   0   0]
 [ 69   0   0]]
enter input file size: 13.2
enter file type: (0-txt,1-img,2-pdf,3-docx)0
  original file type
0          13.2    0
['HUFFMAN']
```

FIGURE 4.4.2

The above image shows the prediction and the code for the following is added in the appendix.

CHAPTER 5

SYSTEM TESTING

5.1 Testing Objectives

The main objective of performing testing on our system is to find any vulnerabilities or bottlenecks which expose test cases that may compromise the functional working or integrity of our system. Testing is done in many different methods; each method of testing inspects the working a different area of system. It may be related to one specific part of the system, or multiple parts of the system working together. There are different types of testing mechanisms. Each test type addresses a specific testing requirement.

5.2 Types of Testing

5.2.1 Unit Testing

This type of testing is done to ensure the lowest individually identifiable pieces of the software in the system are tested for functional correctness and expected performance. Unit tests are used to execute basic tests at unit level (of a single unbreakable part) and test the working of a specific flow in the application with

different types of system configuration. For our project unit testing was run on each of the components: Huffman coding, LZW compression, Shannon Fano compression.

5.2.2 Functional Testing

Testing involves checking the sanity of the data that goes in and out of the system, and the processing it undergoes inside the system. Various type of inputs is passed in to check consistency of the system and its ability to handle exception cases.

Functional testing is cantered on the following items:

• Input Parameters: Providing different type of valid input data to test different corner cases.

• Data Output: Various types of output data corresponding to different types of input

must be observed.

• Functional: Individual functions with well-defined input and output data must be identified and each function must be tested individually.

• Garbage/Invalid Input Parameters: Checking the performance of the system/ flow when the input provided does not match the expected format.

5.2.3 Integration Testing

Integration testing is done on our system to ensure individual components are working properly in sync with one another.

5.2.4 Black Box Testing

This type of testing involves observing the working of the system from an outsider or user perspective without having any knowledge about the internal technical workings. The focus is mainly on user experience and making sure that the high-level functions work as expected by the user.

5.2.5 System Testing

System testing encompasses testing the entire system with varying configuration setups and asserting that the system performs as per requirements in each of the scenarios. In our case, system testing would involve the entire flow right from Huffman coding compression, LZW compression, Shannon Fano Compression, prediction of the compression. The varying parameters could be compression ratio, compression speed and compressed data.

CHAPTER 6

6.1 CONCLUSION

In this paper the suggested method focuses specifically on reducing the ability of machine learning to compact data techniques. In specific, we have implemented numerous techniques for compacting and forecasting the correct technique using machine learning algorithms. As a consequence, it is an effective form of data compression to represent data by linear regression, divide and conquer technique, divide data by time, and use Euclidean Distance machine learning technologies in conquering phase. SVM classifiers and Decision tree classifiers have been used. There are three algorithms used namely Huffman coding, LZW coding and Shannon Fano to compress text file and given the result that which technique is best.

6.2 FUTURE ENHANCEMENT

In the future we will try to add more data compression technique algorithms and also try to increase the type of data for compression. the machine learning code can be rewritten for better accuracy and prediction. The additional use of more classifiers will help the prediction to be much easier and accurate

CHAPTER 7

REFERENCES

[1] S. Yamagiwa, R. Morita and K. Marumo, "Bank Select Method for Reducing Symbol Search Operations on Stream-Based Lossless Data Compression," 2019 Data Compression Conference (DCC), Snowbird, UT, USA, 2019, pp. 611-611.

[2] C. Constantinescu and G. Schmidt, "Fast and Efficient Compression of Next Generation Sequencing Data," 2018 Data Compression Conference, Snowbird, UT, 2018, pp. 402-402.

[3] M. Goyal, K. Tatwawadi, S. Chandak and I. Ochoa, "DeepZip: Lossless Data Compression Using Recurrent Neural Networks," 2019 Data Compression Conference (DCC), Snowbird, UT, USA, 2019, pp. 575-575.

[4] W. Li, "Optimize Genomics Data Compression with Hardware Accelerator," 2017 Data Compression Conference (DCC), Snowbird, UT, 2017, pp. 446-446.

[5] F. ARTUĞER and F. ÖZKAYNAK, "Fractal Image Compression Method for Lossy Data Compression," 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 2018, pp. 1-6.

[6] W. Li and Y. Yao, "Accelerate Data Compression in File System," 2016 Data Compression Conference (DCC), Snowbird, UT, 2016, pp. 615-615.

[7] K. Sharma and K. Gupta, "Lossless data compression technique with encryption-based approach," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-5.

[8] Y. Zhou, M. Huang, F. Jiang, C. Jiang and B. Shan, "A Visualization-Oriented Trajectory Data Compression Method," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 2019, pp. 3432-3435.

[9] J. K. Gibson, D. Lee, J. Choi and A. Sim, "Dynamic Online Performance Optimization in Streaming Data Compression," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 534-541.

[10] S. J. Sarkar, P. K. Kundu and G. Sarkar, "Development of Combined Differential Binary Encoding Algorithm for Power System Operational Data Compression," 2018 10th International Conference on Electrical and Computer Engineering (ICECE), Dhaka, Bangladesh, 2018, pp. 109-112.

CHAPTER 8

APPENDIX

MACHINE LEARNING CODING

```
import numpy as np
from sklearn import linear_model
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
```

In [3]:

```
df=pd.read_csv(r"C:\Users\ASUS\Downloads\final project\Book2.csv")
df
```

In [4]:

```
x = df.iloc[:, :-1].values
print(x)
y = df.iloc[:, 2].values
print(y)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/3, random_stat
e=0)
```

In [5]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion = 'entropy')
```

```
clf.fit(x_train, y_train)
y_pred =  clf.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test, y_pred))
original_ff=float(input("enter input file size: "))
ftype=input("enter file type: (0-txt,1-img,2-pdf,3-docx)")
columns = ['original file','type']
values = np.array([original_ff,ftype])
pred = pd.DataFrame(values.reshape(-1, len(values)),columns=columns)
# print(pred.dtype)
print(pred)

prediction = clf.predict(pred)
print(prediction)
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


clf = SVC(kernel='linear')
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test, y_pred))

original_ff=float(input("enter input file size: "))
ftype=input("enter file type: (0-txt,1-img,2-pdf,3-docx)")
columns = ['original file','type']
values = np.array([original_ff,ftype])
pred = pd.DataFrame(values.reshape(-1, len(values)),columns=columns)
# print(pred.dtype)
print(pred)
prediction = clf.predict(pred)
print(prediction)
```