

DEVELOPMENT OF UNSUPERVISED AND SEMI-SUPERVISED ALGORITHMS FOR CLUSTERING PROTEIN SEQUENCES

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

BY

Shruti Jain

EN16CS301251

Under the Guidance of

Prof. (Dr.) N. Hemachandra

Prof. (Dr.) Ruchi Patel



Department of Computer Science & Engineering

Faculty of Engineering

MEDI-CAPS UNIVERSITY, INDORE- 453331

MAY 2020

DEVELOPMENT OF UNSUPERVISED AND SEMI-SUPERVISED ALGORITHMS FOR CLUSTERING PROTEIN SEQUENCES

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

BY

Shruti Jain

EN16CS301251

Under the Guidance of

Prof. (Dr.) N. Hemachandra

Prof. (Dr.) Ruchi Patel



Department of Computer Science & Engineering

Faculty of Engineering

MEDI-CAPS UNIVERSITY, INDORE- 453331

MAY 2020

Report Approval

The project work “**Development of Unsupervised and Semi-Supervised Algorithms for Clustering Protein Sequences**” is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation:

Affiliation:

External Examiner

Name:

Designation:

Affiliation:

Declaration

I hereby declare that the project entitled “**Development of Unsupervised Algorithms for Clustering Protein Sequences**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science and Engineering’ completed under the supervision of **Prof. (Dr.) N. Hemachandra, Industrial Engineering and Operations Research**, Indian Institute of Technology, Bombay and **Prof. (Dr.) Ruchi Patel, Department of Computer Science and Engineering**, Faculty of Engineering, Medi-Caps University, Indore, is an authentic work.

Further, I declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Shruti Jain

EN16CS301251

Certificate

We, **N. Hemachandra** and **Ruchi Patel** certify that the project entitled **“Development of unsupervised and semi-supervised algorithms for clustering protein sequences”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Ms. Shruti Jain**, is the record carried out by her under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

Prof. (Dr.) Ruchi Patel

Department of Computer Science and
Engineering

Medi-Caps University, Indore

Prof. (Dr.) N. Hemachandra

Industrial Engineering and Operations
Research

Indian Institute of Technology, Bombay

Dr. Suresh Jain

Head of the Department

Computer Science and Engineering

Medi-Caps University, Indore

Acknowledgement

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal**, who has provided me with this internship opportunity to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Sunil K Somani**, Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda**, Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to explore domains of working as an intern. I would also like to thank our Head of the Department **Prof. (Dr.) Suresh Jain** for his continuous encouragement.

I express my heartfelt gratitude to my project guide and Principal Investigator, **Prof. N Hemachandra, Industrial Engineering and Operations Research, IIT Bombay** without whose continuous guidance and support the completion of this project would not have been possible.

I express my gratitude **Prof. Milind Atrey, Dean (R&D), IIT Bombay** for providing me this golden opportunity of being a part of the Research Internship Awards 2019-20 at IIT Bombay. I would like to thank **Prof. Harsh C Phuleria and Prof. Sunita Srivastava, IRCC Internship Coordinators at IIT Bombay** for their continuous support throughout the internship.

I would also like to extend my gratitude to our project coordinator **Prof. (Dr.) Ruchi Patel** who extended her support and necessary inputs important for successful submission of this project.

Shruti Jain (EN16CS301251)

B.Tech. IV Year

Department of Computer Science and Engineering

Faculty of Engineering

Medi-Caps University, Indore

Abstract

Technological advances have resulted in a phenomenal reduction in the cost and time of whole genome sequencing. Consequently the size of protein database is increasing steadily. In last six months, on average 300,000 sequences were added to the database every day. This presents a major challenge of assigning function to these sequences. One approach is to sort these sequences into orthologous groups using machine learning approach. Specifically, the project aims to develop methods for clustering proteins sequences. This is expected to facilitate protein function annotation. This will also help to know shared ancestry between proteins.

A Protein molecule is made from a long chain of twenty amino acids, each linked to its neighbour through a covalent peptide bond. Proteins are also known as polypeptides. Each type of protein has a unique sequence of amino acids. The final protein structure ultimately depends on this sequence. It plays role in determining the function of a protein. A protein sequence is a sequence of amino acids and each amino acid is represented as an alphabet. Amino acid sequences are represented with their corresponding 1 letter code, for example, code for alanine is (A), arginine is (R) and so on.

This project aims to study the methods available for clustering data available in Machine Learning and implement them according to the nature of dataset. Proteins can be made from 20 different kinds of amino acids, and the structure and function of each protein are determined by the kinds of amino acids used to make it and how they are arranged. Understanding this relationship between amino acid sequence and protein function is a long-standing problem in molecular biology with far-reaching scientific implications. Proteins are the key molecules that facilitate most biological processes within a cell. Therefore, the discovery, annotation and characterization of them, is of great importance.

The number of unique entries in all protein sequence databases together exceeds now about a million. However, biological evolution lets proteins fall into families, thus imposing a natural grouping. A protein family contains sequences that are evolutionarily related. Generally, this is reflected by sequence similarity. Therefore, one aims at organizing the set of all protein sequences into clusters based on their sequence similarity. Clustering a large set of sequences offers several advantages. A frequent problem is the identification of sequences that are similar to a new query sequence where a newly identified protein sequence can be put into an existing group or form a new group which can help to annotate the possible functionality and nature of this new protein and can help scientists to further carry out their studies.

The project considers the large size of data which is continuously evolving indicating a streaming data behavior and hence also trying to implement the stream data clustering on protein sequences.

Keywords: Clustering, protein sequences, data stream

Table of Contents

		Page No.
	Report Approval	ii
	Declaration	iii
	Certificate	iv
	Acknowledgement	v
	Abstract	vi
	Table of Contents	vii
	List of figures	viii
	Abbreviations	ix
	List of tables	x
Chapter 1	Introduction	
	1.1 Introduction	1
	1.2 Literature Review	2
	1.3 Objectives	4
	1.4 Significance	4
	1.5 Research Design	5
	1.6 Data Source	5
Chapter 2	System Requirement Analysis	
	2.1 Information gathering	6
	2.2 System Feasibility	7
Chapter 3	System Analysis	
	3.1 Information Flow Diagram	9
	3.2 Sequence Diagram	10
Chapter 4	Design	
	4.1 Architectural Design	11
	4.2 Modules and procedures used	14
Chapter 5	Testing	
	Testing objectives	21
	Testing scope	22
	Testing principles	22
	Test cases	25
Chapter 6	Results and discussions	29
Chapter 7	Summary and conclusion	36
Chapter 8	Future Scope	37
	Appendices	38
	Bibliography and References	42

List of Figures

Figure No.	Figure Name	Page No.
1.1	Growing nature of data	5
2.1	Steps in Machine learning application	7
3.1	Information Flow Diagram	9
3.2	Sequence diagram	10
4.1	Control hierarchy	13
5.1	Clusters for k=3	25
5.2	Clusters for k=4	26
5.3	Clusters for k=5	26
5.4	Clusters for k=6	27
5.5	Graph for elbow method	27
5.6	Graph for silhouette score	28
5.7	Clusters labels for sample data	28
5.8	Cluster for 100 proteins using kmeans	29
5.9	Spectral Clustering	29
5.10	Decreasing SSE	30
5.11	3d plot	30
5.12	kmeans clustering for 1000 proteins	31
5.13	Decreasing SSE for 1000 proteins	31
6.1	Dropping the malicious entries	32
6.2	Shape of vector	32
6.3	Cluster labels	33
6.4	Clusters for Kmeans	33
6.5	Silhouette scores for kmeans	34
6.6	Dendrogram for keywords	34
6.7	Dendrogram for similarity matrix	35
6.8	Silhouette score plot	35

Abbreviations

Abbreviation	Description
HMM	Hidden Markov Model
BLOSUM	Blocks Substitution Matrix
PAM	Point accepted mutation
IDE	Integrated Development Environment
SSE	Sum of squared errors

List of tables

Table No.	Title	Page no.
4.1	Sklearn algorithms	15

Chapter -1

Introduction

1.1 Introduction

The term machine learning refers to a set of topics dealing with the creation and evaluation of algorithms that facilitate pattern recognition, classification, and prediction, based on models derived from existing data. Recent advancements in technology are enabling us to store an incredible amount of data. Initially, size of data was perceived as a problem to be solved. In fact, we had reached a point in which we were able to store too much data without being able to make the best use of it. Nowadays, what was considered first as a problem, has now become an open door to a world of innovations. We can store bulk of data. Now what poses as a problem is processing of the data and deriving useful results from it.

Biology is a subject which makes wide use of biological databases to try to tackle many different challenges such as understanding the treatment for diseases and cellular function. Datasets of biological data can be created from amino-acid sequences, nucleotides, macromolecular structures and so on. Computational biology and bioinformatics is an interdisciplinary field that develops and applies computational methods to analyse large collections of biological data, such as genetic sequences, cell populations or protein samples, to make new predictions or discover new biology.[1] The computational methods that are used include analytical methods, mathematical modelling and simulation.

A machine learning algorithm is a computational method based upon statistics, implemented in software, able to discover hidden non-obvious patterns in a dataset, and moreover to make reliable statistical predictions about similar new data. As explained by Kevin Yip and colleagues:- “The ability[of machine learning]to automatically identify patterns in data [...] is particularly important when the expert knowledge is incomplete or inaccurate, when the amount of available data is too large to be handled manually, or when there are exceptions to the general cases” [2]. This is clearly the case for computational biology and bioinformatics. Machine learning has thus been applied to multiple computational biology problems so far helping scientific researchers to discover knowledge about many aspects of biology.

Use of Machine Learning in Computational Biology is now becoming more and more important. Two main paradigms exist in the field of machine learning: supervised and unsupervised learning. Both have potential applications in biology. In supervised learning, objects in a given collection are classified using a set of attributes, or features. The result of the classification process is a set of rules that prescribe assignments of objects to classes based solely on values of features. In a biological context, examples of object-to-class mappings are tissue gene expression profiles to disease group, and protein sequences to their secondary structures. The features in these examples are the expression levels of individual genes measured in the tissue samples and the presence/absence of a given amino acid symbol at a given position in the protein sequence, respectively. In unsupervised learning, no predefined class labels are available for the objects under study. In this case, the goal is to explore the data and discover similarities between objects. Similarities are used to define groups of objects, referred to as clusters. In other words, unsupervised learning is intended to unveil natural groupings in the data.

1.2 Literature Review

Biological sequences databases are growing constantly in large sizes, and handling of this data becomes tedious process. Protein sequences fall in such a category. There is an innate nature of this data, biological evolution, which lets proteins fall into some families, introducing a natural grouping. This is generally reflected in sequence similarities.

The aim of clustering protein sequences is to get a biologically meaningful partitioning. One of the simplest well-studied and computationally cheap methods to construct a clustering of data points is single linkage clustering. Starting with the pair of data points of least distance, one incrementally merges single data points or already existing clusters. Such a hierarchical clustering can be viewed as a tree, called the single linkage tree. The leaves represent the individual data points, while the root of this tree corresponds to just one large cluster representing the whole data set. All other layers in between can be seen as cluster sets at different levels of similarity. However, it is not clear which layers give a meaningful partitioning of the data. They should be chosen so that they neither produce small trivial clusters nor form huge uninformative clusters. Several approaches already deal with the problem of partitioning a protein sequence database into protein families.

Automatically generated cluster sets like ProtoMap, ProtoNet, or CluSTr typically provide a hierarchical classification at several different levels of similarity. Other methods, like iProClass or PIRSF include further knowledge, e.g., from domain based classifications, or require manual interaction. [4]

Many of the existing clustering methods can be used for clustering sequences if a suitable distance (or similarity) metric is available. A distance metric takes a pair of sequences and returns a real number which denotes the distance between the given sequences.

The most popular distance metric for sequence data is the Levenshtein distance, or edit distance. Other metric that can be used is based on keyword similarity, a sequence can be represented as a vector, in which each component corresponds to the frequency of one of the q -length segments. Then the similarity between two sequences can be calculated. Kernel-based sequence similarity metrics can also be used. Probabilistic models, such as HMM are also used for finding similarity metrics. For a given set of sequences to be clustered, such a method trains one HMM for each of the sequences. Then, the similarity between two sequences can be obtained from the similarity (or distance) between the corresponding HMMs [5]

Algorithms can be compared on the basis of various factors which include clustering method (concept/type), Linkage strategy, time complexity, outlier detection, number of clusters, feasibility, data sources, number of sequences, similarity measure, scoring matrix[Appendix blossom], suitability for large datasets.[6]

Some existing approaches include hierarchical, spectral, partition-based, density based. Spectral methods allow one to study global properties of a dataset by making only pairwise similarity measurements between data points. A new initialization method for the K-means algorithm has been proposed to solve problems associated with random selection. In the new initialization method, we try to choose suitable initial points, which are well separated and have the potential to form high-quality clusters.

We can use any clustering method for clustering biological sequences. Nevertheless, there have been many clustering methods that are explicitly proposed for clustering biological sequences. A subsequence-based clustering method mines a set of frequent subsequences from each of the sequences and uses them as features for clustering the sequences. The idea of such clustering is similar to the task of document clustering using the “bag-of-words” representation of a document. A traditional sequence mining method returns a large number of subsequences that are frequent, but all such subsequences are not good features for clustering. So, a good clustering method needs to choose a subset of these sub-sequences as features so that when projected on the feature-set the similarity between a pair of sequences is computed correctly. Different algorithms of sequence clustering vary in the way they choose the subsequence feature set. The advantages of subsequence-based clustering methods is that they are typically fast compared to other sequence clustering methods.

A graph-based sequence clustering method represents the sequences in a similarity graph, in which a vertex represents a sequence and an edge represents the similarity relation between the corresponding pair of sequences. In such a representation, a partition of the similarity graph represents a clustering of the input sequences. The crucial requirement in a graph-based sequence clustering method is to obtain the similarity graph in an efficient manner. The probabilistic approach is popular for sequence modeling. For instance, the earliest approaches

for modeling protein families use profile-HMM, a hidden Markov model-based probabilistic approach.

The UniProt Archive is a comprehensive and non-redundant database that contains most of the publicly available protein sequences in the world. Protein families can be characterized by molecules which share significant sequence similarity. Notably, this biological problem is very difficult to solve and most available clustering techniques fail in the case of eukaryotic proteins, which contain large numbers of protein domains. Nevertheless, ongoing efforts in detecting the best and more accurate protein clustering are still a very active research field. Several tools today, follow various methodologies and strategies to perform protein clustering. Outstanding tools such as the CD-HIT, UCLUST, kClust and the newly developed MMSEQ or LinClust follow a k-mer and dynamic programming-based sequence alignment approach whereas tools such as the MCL clustering algorithm and others a network topology based clustering.

1.3 Objectives

- The size of protein database is increasing steadily. This presents a major challenge of assigning function to the sequences.
- To develop methods for clustering protein sequences.
- To facilitate protein function annotation.
- To help in knowing shared ancestry between proteins.

1.4 Significance

There is an increasing demand for the development of automatic and accurate techniques for protein sequences comparison, classification, and functional prediction. An important problem in today's genomics is that of grouping together homologous proteins when only sequence information is available. This problem is difficult since sequence similarity is a very noisy measure.

This project is of utility in aiding to solve this problem by forming clusters of proteins. Sequence clustering is an essential task in biological data analysis, because the most important building blocks of living organisms, such as DNA, RNA, mRNA, polypeptides, and proteins, have a linear structure and can be represented as sequences. Protein clustering is used to construct meaningful and stable groups of similar proteins to be used for analysis and functional annotation.

Proteins are the key molecules that facilitate most biological processes within a cell. Therefore, the discovery, annotation and characterization of them, is of great importance.[3]

1.5 Research Design

The methods and algorithms will be studied through literature reviews, observations and expert discussions in exploratory manner. Aim is to develop ideas and get insights in proceeding the project. It will be carried out on a requirement basis and will be majorly based on reading and implementation to obtain results and compare them at last.

Literature reviews will include reading the existing research papers on clustering methods and papers on how to use protein sequences for machine learning tasks. For this online a search of online library will be used. Observations depend on the implementation and existing results. Expert discussions will help to understand what protein sequences represent and how are they useful for knowing the functions. The sequences are the primary structure and this primary structure is further used for secondary and tertiary structures of proteins.

1.6 Source of Data

A CSV file of sequence data set was obtained from:-

<https://www.kaggle.com/shahir/protein-data-set>

A dataset of FASTA file format was obtained from:-

www.uniprot.org

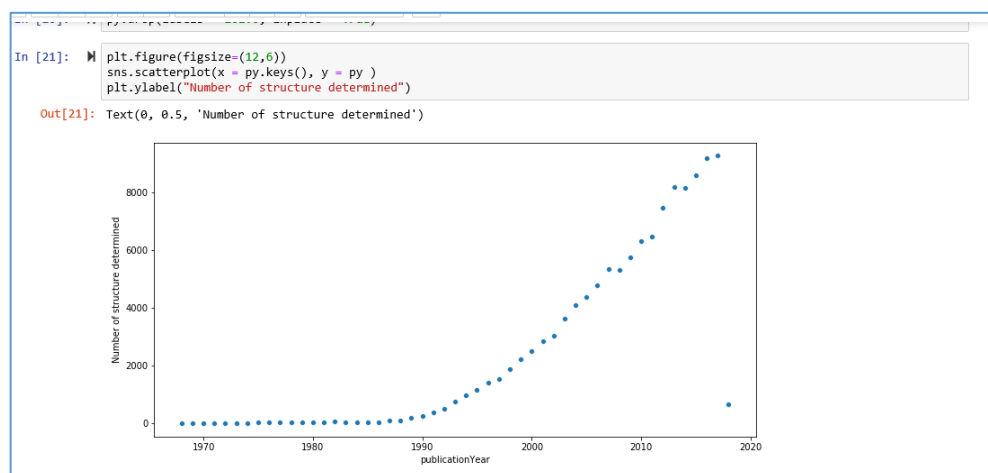


Fig 1.1 Growing nature of data

Chapter –2

System Requirement Analysis

2.1 Information Gathering

A requirement is a vital feature of a new System which may include processing or capturing the data, controlling the activities of business, producing information and supporting the management.

Information gathering is very key part of the feasibility analysis process. Information gathering is both art and a science. It is a science because it requires a proper methodology and tools in order to be effective. It is an art too, because it requires a sort of mental dexterity to achieve the best results.

For the project a task analysis and domain analysis approach is followed. Also use cases are analysed and prototypes on some data, which is easy to process, is performed. After discussion with the principal investigator, the project required to be able to work with stream data. It was observed by understanding the nature of protein sequence database which is continuously growing.

To understand the basics of clustering, I read various research papers and survey papers to know what tools and techniques are used to achieve the environment requires for the project. By that knowledge I got to know what libraries and biotechnology software tools and repositories I can use to assist the project. Also it was important to learn the concepts necessary to develop the algorithms which were based on statistical methods.

On discussions with the biosciences team we found that the existing problem with primary structure in the form of protein sequences is that it doesn't consider the three dimensional molecular structure of protein which is responsible for the behavior of the protein molecule. So it was necessary to understand the utility of protein sequences in determining the function of a protein. After discussions it was concluded that protein structures can help in annotations as per previous researches carried out in the same direction. With the hope to obtain interesting results the project has to be carried out. The sequential nature of data is also taken into consideration for the application of algorithms.

Machine learning helps us find patterns in data—patterns we then use to make predictions about new data points. To get those predictions right, we must construct the data set and transform the data correctly.

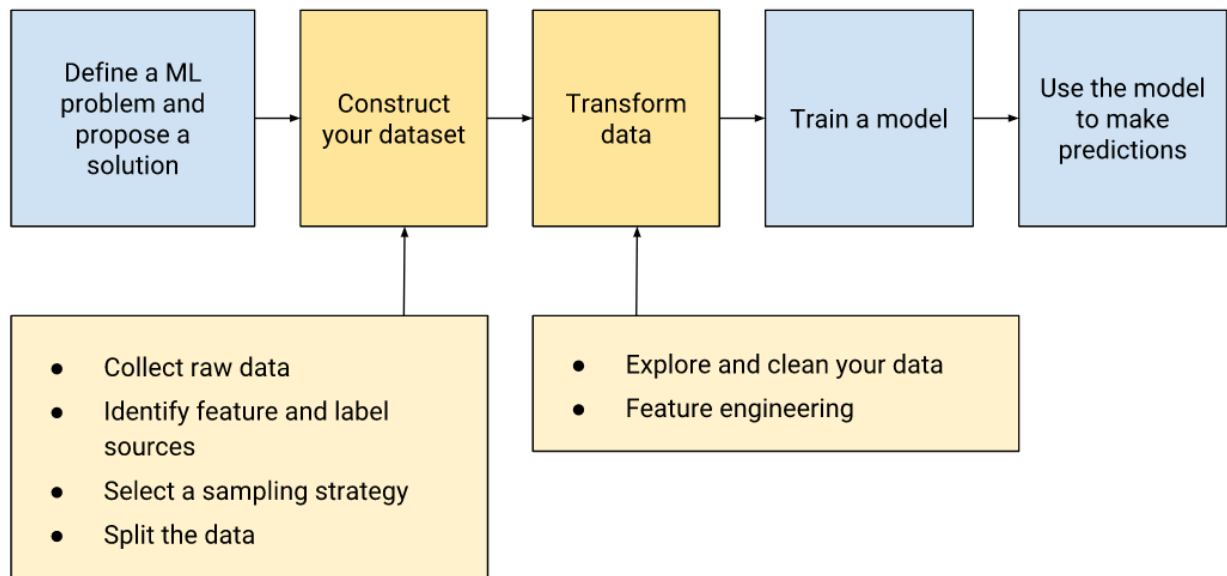


Figure 2.1: Steps in Machine learning application

However, while collecting data, it's helpful to have a more concrete definition of quality. Certain aspects of quality tend to correspond to better-performing models: reliability, feature representation, minimizing skew.

Representation is the mapping of data to useful features. Converting non-numeric features into numeric. You can't do matrix multiplication on a string, so we must convert the string to some numeric representation.

2.2 System Feasibility

Feasibility is defined as the practical extent to which a project can be performed successfully. Information such as resource availability, cost estimation for software development, benefits of the software are considered. It specifies how the project is deployed to the platform and what the requirements for deploying the project are.

2.2.1 Resource Availability (Technical)

The resources required for this project are a dataset of protein sequences which are available on various resources online and are free of cost. Some of which include UniProt, SwissProt, NCBI repository. There is also a requirement of study material since it is an interdisciplinary project, to understand the terms in biosciences and to be able to utilize the data set. Other than these resources we need a computer system with high processing speed that can handle large amount of data and perform mathematical computations on it. All these resources are easily available to develop this project with a little compromise on the processing speed as the kind of processing speed we require to develop this project can only be provided by GPU but the installed graphics card can work as a makeshift GPU for the developmental scope of this project

2.2.3 Benefits of the Software

The number of unique entries in all protein sequence databases together exceeds now about a million. However, biological evolution lets proteins fall into families, thus imposing a natural grouping. A protein family contains sequences that are evolutionarily related. Generally, this is reflected by sequence similarity. Therefore, one aims at organizing the set of all protein sequences into clusters based on their sequence similarity. A frequent problem is the identification of sequences that are similar to a new query sequence where a newly identified protein sequence can be put into an existing group or form a new group which can help to annotate the possible functionality and nature of this new protein and can help scientists to further carry out their studies.

2.3 Platform Specification

2.3.1 Hardware

- A laptop or PC with preferably 64 bit OS
- 8 GB RAM (or more)

2.3.2 Software

- Visual Studio Code with Python interpreter (intellicode)
- Spreadsheet software like Microsoft Excel
- Python Scikit, numpy, sklearn, pandas, matplotlib libraries
- BLAST and FASTA processing software online at ncbi
- R Studio
- Jupyter Notebook

Chapter - 3

System Analysis

System Analysis is the act, process, or profession of studying an activity (such as a procedure, a business, or a physiological function) typically by mathematical means in order to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently.

3.1 Information Flow Representation

An **information flow diagram (IFD)** is a diagram that shows how information is communicated (or "flows") from a *source* to a *receiver* or *target* (e.g. $A \rightarrow C$), through some medium. The medium acts as a bridge, a means of transmitting the information.. The concept of IFD was initially used in radio transmission. The return paths can be two-way or bi-directional: information can flow back and forth. An IFD can be used to model the information flow throughout an organisation. An IFD shows the relationship between internal information flows within an organisation and external information flows between organisations. It also shows the relationship between the internal departments and sub-systems. An IFD usually uses "blobs" to decompose the system and sub-systems into elemental parts. Lines then indicate how the information travels from one system to another. IFDs are used in businesses, government agencies, television and cinematic process.

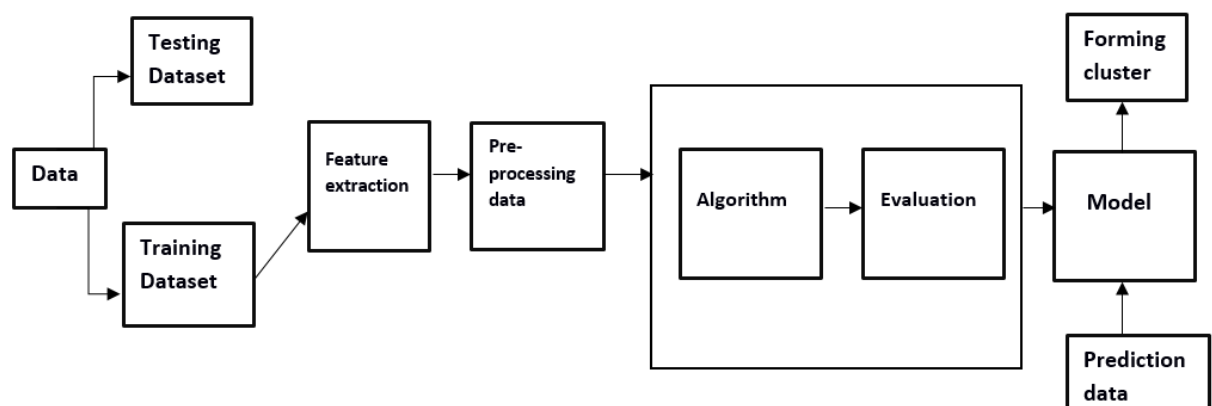


Figure 3.1 Information flow diagram

3.1.1 Sequence Diagram

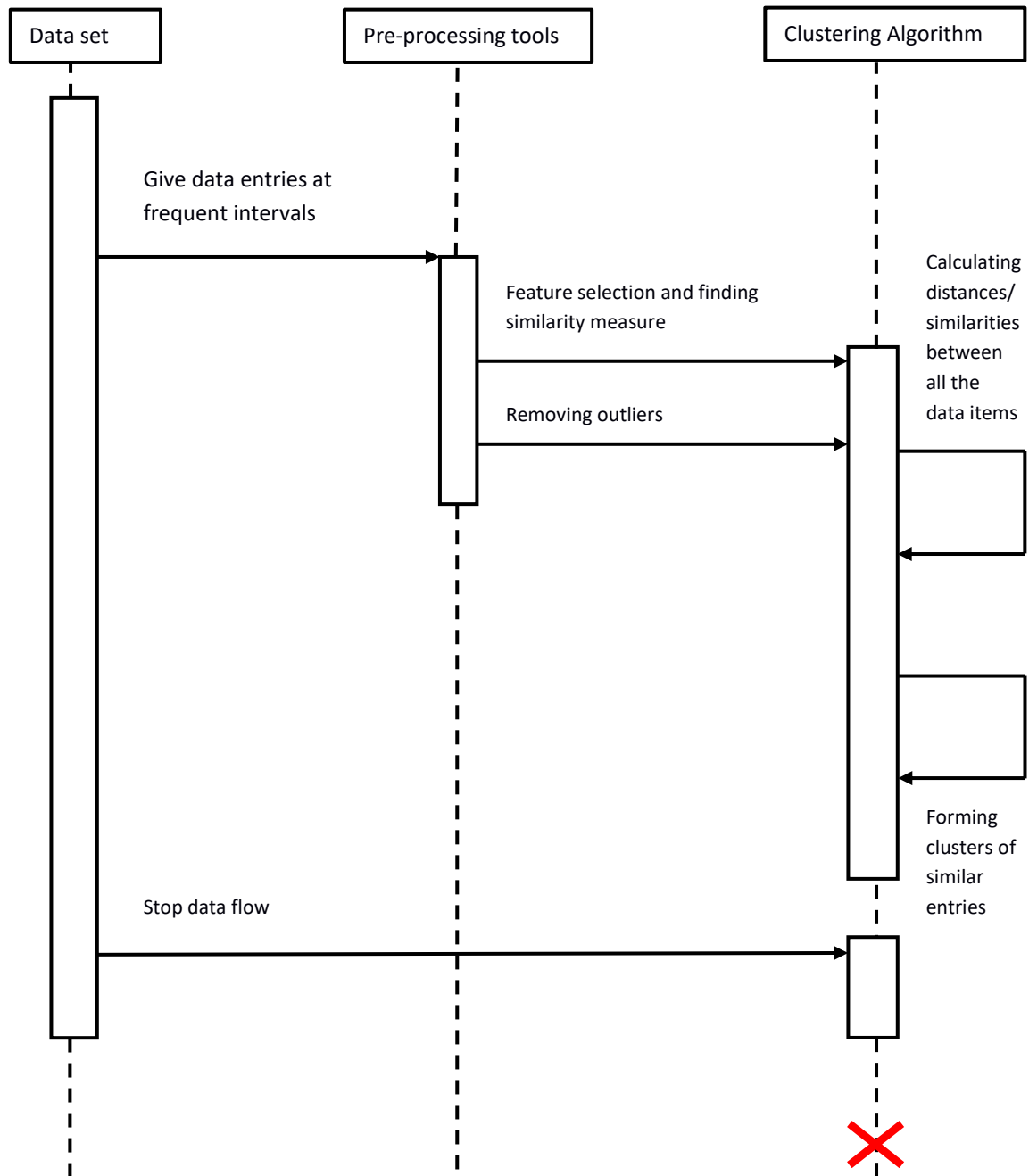


Figure 3.2 Sequence diagram

Chapter - 4

Design

Software architecture involves the high level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability.

A software architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.

A software architecture can be defined in many ways –

- **UML (Unified Modeling Language)** – UML is one of object-oriented solutions used in software modeling and design.
- **Architecture View Model (4+1 view model)** – Architecture view model represents the functional and non-functional requirements of software application.
- **ADL (Architecture Description Language)** – ADL defines the software architecture formally and semantically

4.1 Architectural Design

IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectural styles. Each style will describe a system category that consists of :

- A set of components (eg: a database) that will perform a function required by the system.
- Computational modules required
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

In this project the following can be used:

- Protein sequence dataset
- Python ML Modules, FASTA sequence generator

4.1.1 Architectural Context Diagram

The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.

A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

In order to produce the context diagram and agree on system scope, the following must be identified:

- external entities: The protein dataset
- data-flows: processed information after every function or algorithm is implemented

4.1.2 Control Hierarchy

Control hierarchy, also called program structure, represents the organization of program components (modules) and implies a hierarchy of control. It does not represent procedural aspects of software such as sequence of processes, occurrence or order of decisions, or repetition of operations; nor is it necessarily applicable to all architectural styles.

Different notations are used to represent control hierarchy for those architectural styles that are amenable to this representation. The most common is the treelike diagram that represents hierarchical control for call and return architectures.

The control relationship among modules is expressed in the following way: A module that controls another module is said to be superordinate to it, and conversely, a module controlled by another is said to be subordinate to the controller.

The principle of hierarchical control is always to start an action, then forget about it at the initiating level until the next level down signals its completion. At that point, a status variable is updated in the initiating system, so that the level above can, by a simple 'read status' message,

find out what state the system is in; while a device or system is in transit from one well-defined state to another, its status is 'undefined'.

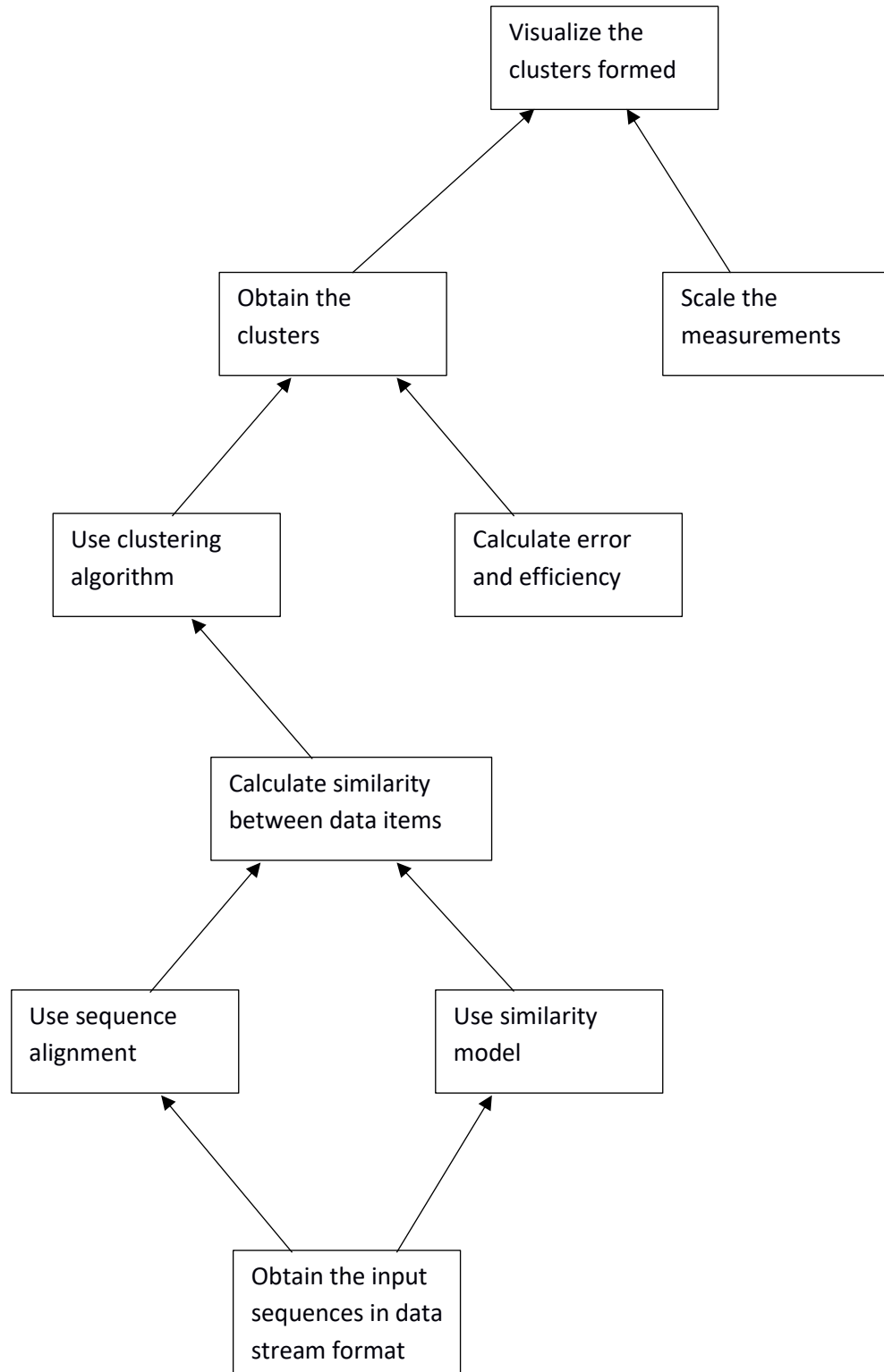


Figure 4.1 Control hierarchy for the project

4.2 Modular and Procedural Approach

Python provides various inbuilt modules for machine learning purposes. In this project we have used various python modules and libraries for various purposes. Since this is an NLP extensive project the first library that we used was the natural language toolkit or the NLTK. Secondly for mathematical computations and clustering purposes we used, pandas, numpy, sk-learn and other libraries. For the development of GUI we used tk-inter or the interface toolkit.

4.2.1 Modules Used

4.2.1.1 sklearn.cluster

The sklearn.cluster module gathers popular unsupervised clustering algorithms.[7]

Algorithm	Description
Cluster.AffinityPropagation()	AffinityPropagation creates clusters by sending messages between pairs of samples until convergence
Cluster.AgglomerativeClustering()	The AgglomerativeClustering object performs a hierarchical clustering using a bottom up approach: each observation starts in its own cluster, and clusters are successively merged together.
Cluster.Birch()	he Birch builds a tree called the Clustering Feature Tree (CFT) for the given data
Cluster.DBSCAN()	The DBSCAN algorithm views clusters as areas of high density separated by areas of low density
Cluster.Kmeans	Partitioning algorithm, K is user defined, and k centres are taken in each iteration and k clusters are created iteratively on the basis of distance between the points till no new cluster is formed
Cluster.MeanShift()	It is a centroid based algorithm

Cluster.OPTICS()	OPTICS (Ordering Points To Identify the Clustering Structure), closely related to DBSCAN, finds core sample of high density and expands clusters from them
Cluster.SpectralClustering	<u>SpectralClustering</u> performs a low-dimension embedding of the affinity matrix between samples, followed by clustering

Table 4.1: sklearn clustering algorithms

4.2.1.2 Pandas

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. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Pandas is mainly used for machine learning in form of dataframes. Pandas allow importing data of various file formats such as csv, excel etc. Pandas allows various data manipulation operations such as groupby, join, merge, melt, concatenation as well as data cleaning features such as filling, replacing or imputing null values.

Major Features of this library are as follows:

- DataFrame 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 subsetting 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 and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.[8]

4.2.1.3 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.

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. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.[9]

4.2.1.4 Sci-kit 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. Clustering of unlabeled data can be performed with the module `sklearn.cluster`.

Each clustering algorithm comes in two variants: a class, that implements the fit method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the `labels_` attribute.

4.2.1.5 MatPlotLib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. 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+. `matplotlib.pyplot` is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In `matplotlib.pyplot` various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes.[10]

4.2.2 Internal Data Structures

While dealing with machine learning projects it becomes inevitable to deal with large chunks of data. To deal with large amounts of data it requires proper organization of data with the help of various data structures. Various data structures are used to store data in various formats. For example, an array stores homogenous data while a list can store heterogeneous data too.

To deal with the large amount of string and character data in this project I have used two distinct data structures in python library, namely list and data frames for their distinct features. While a list is a linear data structure that can store any data type in a linear form in a single row a data frame is like a two dimensional matrix that can be used to obtain the function of a table to store and represent data.

For storing the similarity matrices, numpy 2-D arrays are used which a 2x2 matrix to store the similarity measure between two protein sequences. In performing the data stream application I use generators in python. For storing the intermediate clustering results in the form centres list

and dataframes are used. In some clustering algorithm there is requirement of using tree data structure which resembles are B+ tree, which is in turn a binary tree.

This simple but elegant use of these simple data structures make the program light on the computer's memory but also makes the processing of the text faster and clustering much more easier. This also makes way for dealing with much larger data in a very small secondary memory space and as much as 57000 data-points can be processed using a secondary memory space of less than just 50 MB.

4.2.3 Algorithm Design for operations

4.2.3.1 K-means

This algorithm aims at minimizing an objective function know as squared error function given by:

$$J(V) = \sum_{i=1}^c \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2$$

where,

‘ $\|x_i - v_j\|$ ’ is the Euclidean distance between x_i and v_j .

‘ c_i ’ is the number of data points in i^{th} cluster.

‘ c ’ is the number of cluster centers.

The Algorithm:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

4.2.3.2 Hierarchical Clustering

Hierarchical clustering can be performed with either a *distance matrix* or *raw data*. This algorithm works by grouping the data one by one on the basis of the nearest distance measure of all the pairwise distance between the data point. Again distance between the data point is recalculated but which distance to consider when the groups has been formed. For this there are many available methods.

Some of them are:

- 1) single-nearest distance or single linkage.
- 2) complete-farthest distance or complete linkage.
- 3) average-average distance or average linkage.
- 4) centroid distance.
- 5) ward's method - sum of squared euclidean distance is minimized.

This way we go on grouping the data until one cluster is formed.

Let $X = \{x_1, x_2, x_3, \dots, x_n\}$ be the set of data points.

Algorithm:

- 1) Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.
- 2) Find the least distance pair of clusters in the current clustering, say pair $(r), (s)$, according to $d[(r),(s)] = \min d[(i),(j)]$ where the minimum is over all pairs of clusters in the current clustering.
- 3) Increment the sequence number: $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m . Set the level of this clustering to $L(m) = d[(r),(s)]$.
- 4) Update the distance matrix, D , by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new cluster, denoted (r,s) and old cluster (k) is defined in this way: $d[(k), (r,s)] = \min (d[(k),(r)], d[(k),(s)])$.
- 5) If all the data points are in one cluster then stop, else repeat from step 2).

4.3.2.3 Data Stream Clustering

A data stream is a massive sequence of objects o_1, o_2, \dots which arrive continuously over time and at a rapid rate. Each object o_i from the stream is a multidimensional vector, $o_i = \langle o_i^1, o_i^2, \dots, o_i^d \rangle$ where d is the dimensionality of the feature space.

The Stream framework by Guha et al [11] is one of the earliest methods for stream clustering. The method is based on k -median clustering and the core idea is to break the stream into batches $B_1, B_2, \dots, B_i, \dots$ of fixed size m . At each batch B_i , the k -median clustering algorithm is applied optimizing the sum of squared error SSE. After processing i batches, a total of $i \cdot k$ medians will have been stored in the prototype array, k per batch. Whenever the number of stored medians exceeds a threshold m , the k -median algorithm is applied over these stored medians, generating the new set of medians.

Algorithm Small Space(S)

1. Divide S into ℓ disjoint pieces X_1, X_2, \dots, X_ℓ .
2. For each i , find $O(k)$ centres in X_i . Assign each point X_i to its closest center.
3. Let X' be the $O(k)$ centres obtained in (2), where each center c is weighted by the number of points assigned to it.
4. Cluster X' to find k centres.

Data Stream Algorithm

1. Input the first m points; use a bicriterion algorithm to reduce these to $O(k)$ (say $2k$) points. As usual, the weight of each intermediate median is the number of points assigned to it in the bicriterion clustering. (Assume m is a multiple of $2k$.) This requires $O(f(m))$ space, which for a primal dual algorithm can be $O(m^2)$. We will see a $O(mk)$ -space algorithm later.
2. Repeat the above till we have seen $m_2 = (2k)$ of the original data points. At this point we have m intermediate medians.
3. Cluster these m first-level medians into $2k$ second-level medians and proceed.
4. In general, maintain at most m level- i medians, and, on seeing m , generate $2k$ level- $i + 1$ medians, with the weight of a new median as the sum of the weights of the intermediate medians assigned to it.
5. After seeing all the original data points (or to have a clustering of the points seen so far) we cluster all the intermediate medians into k final medians.

4.2.3.4 Sequence Matching

1. The Longest Common Subsequence (LCSS) distance is a variation of EDIT distance. EDIT distance (ED) comes from the field of string comparison and measures the number of insert, delete, and replace operations that are needed to make two strings of possibly different lengths identical to each other. The LCSS distance between two real-valued sequences S_1 and S_2 of length m and n , respectively, is computed as follows:

$$(a) D_{LCSS, \delta, \epsilon}(S_1, S_2) = 0, \text{ if } n = 0 \text{ or } m = 0$$

$$(b) D_{LCSS, \delta, \epsilon}(S_1, S_2) = 1 + D_{LCSS, \delta, \epsilon}(\text{HEAD}(S_1), \text{HEAD}(S_2))$$

$$\text{if } |S_{1,m} - S_{2,m}| < \epsilon \text{ and } |m - n| \leq \delta$$

$$(c) \max \begin{cases} D_{LCSS, \delta, \epsilon}(\text{HEAD}(S_1), S_2) \\ D_{LCSS, \delta, \epsilon}(S_1, \text{HEAD}(S_2)) \end{cases}, \text{ otherwise}$$

Where HEAD(S1) is the subsequence [S1,1,S1,2,...,S1,m-1] , δ is an integer that controls the maximum distance in the time axis between two matched elements, and ε is a real number

$0 < \varepsilon < 1$ that controls the maximum distance that two elements.

2. Similarity of two strings is usually calculated. For a given alignment A of S1 and S2 strings, let S1' and S2' denote the strings after the chosen insertion of spaces, and let l denote the length of the two strings S1' and S2' in A. The value of alignment A is defined as

$$Z_i = \sum_{i=1}^l s(S'_1(i), S'_2(i))$$

For calculating these scores BLOSUM and PAM [appendix A] scoring matrices are used.

Global Alignment is calculated using Dynamic Programming that uses Needleman-Wunsch Algorithm. The final step in the algorithm is the trace back for the best alignment. By continuing the trace one would reach to the 0th row, 0th column.

$$M_{i,j} = \text{MAX}[M_{i-1,j-1} + S_{i,j}, M_{i,j-1} + W, M_{i-1,j} + W]$$

Where,

i,j describes row and columns.

M is the matrix value of the required cell

S is the score of the required cell ($S_{i,j}$)

W is the gap alignment

Local Alignment is calculated using Smith-Waterman Algorithm. The final step for the appropriate alignment is trace backing, prior to that one needs to find out the maximum score obtained in the entire matrix for the local alignment of the by scoring it.[12]

$$M_{i,j} = \text{MAX}[M_{i-1,j-1} + S_{i,j}, M_{i,j-1} + W, M_{i-1,j} + W, 0]$$

Where,

i,j describes row and columns.

M is the matrix value of the required cell

S is the score of the required cell ($S_{i,j}$)

W is the gap alignment

Chapter - 5

Testing

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

5.1 Testing Objectives

Target of the test are –

- **Errors** – These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, is considered as an error.
- **Fault** – When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.
- **Failure** – failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

Testing can either be done manually or using an automated testing tool:

- **Manual** – This testing is performed without taking help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager.

Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.

- **Automated** This testing is a testing procedure done with aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation in concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also behaviour.

5.2 Testing Scope

There are two ends in which we had to test our software. The GUI and the algorithm. While the GUI was very simple and didn't have much scope to be fooled around with, we could focus on testing the algorithm on a wide variety of input.

The test strategy that we chose was to progress gradually. We started with a small set of data points and applied the test cases on it. After the test cases gave the desired results we increased the data points and reran the tests cases all over again. After a certain number of iterations of increasing the data points when we were satisfied with the performance of the algorithm including its processing time and accuracy of the results we tested it with our friends and colleagues and asked their reviews on the accuracy of the results.

The responses that we got were very encouraging. After testing the algorithm in front of a reasonable sample size of people we were satisfied with its performance in the current form and gave it a go ahead.

5.3 Testing Principles

Black-box testing

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise. In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Black-box testing techniques:

- **Equivalence class** – The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** – The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- **Cause-effect graphing** – In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- **Pair-wise Testing** – The behaviour of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** – The system changes state on provision of input. These systems are tested based on their states and input.

White-box testing

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as ‘Structural’ testing.

In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

The below are some White-box testing techniques:

- **Control-flow testing** – The purpose of the control-flow testing to set up test cases which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- **Data-flow testing** – This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified. Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels as discussed in the following sections. Namely it is Unit and Integration testing.

Unit Testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updation etc.

5.4 Test Cases

On implementing K-means with different values of K the following results were obtained:-

K=3

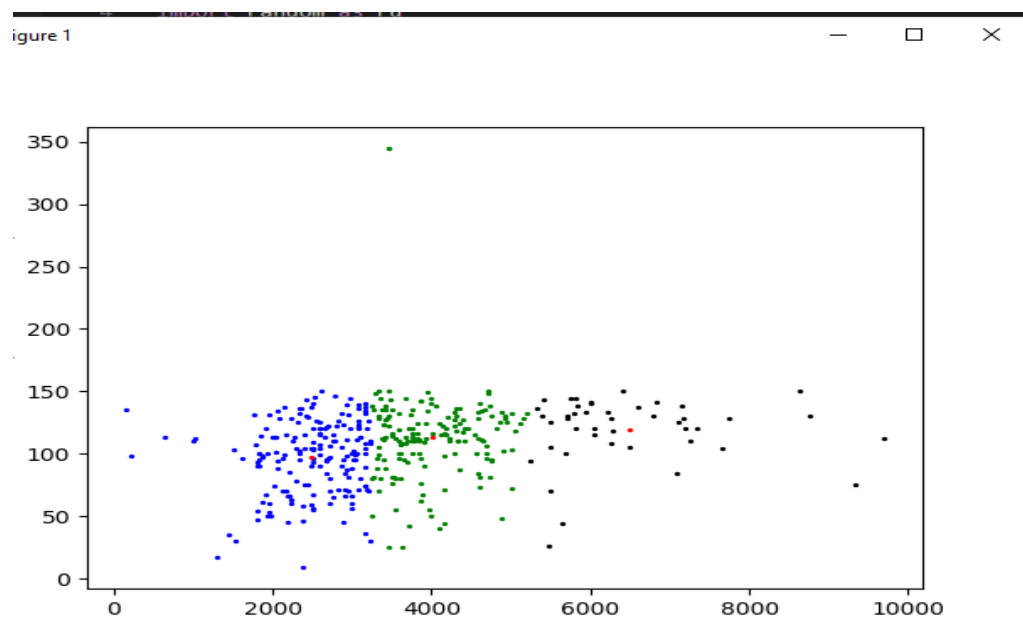


Figure 5. 1 Clusters for k=3

K=4

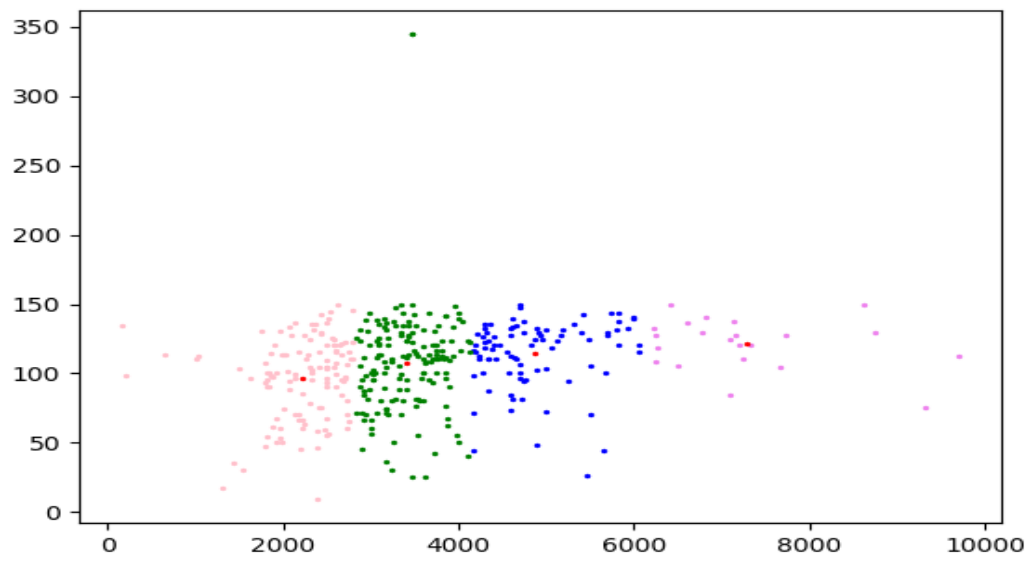


Figure 5. 2 Clusters for k=4

K=5

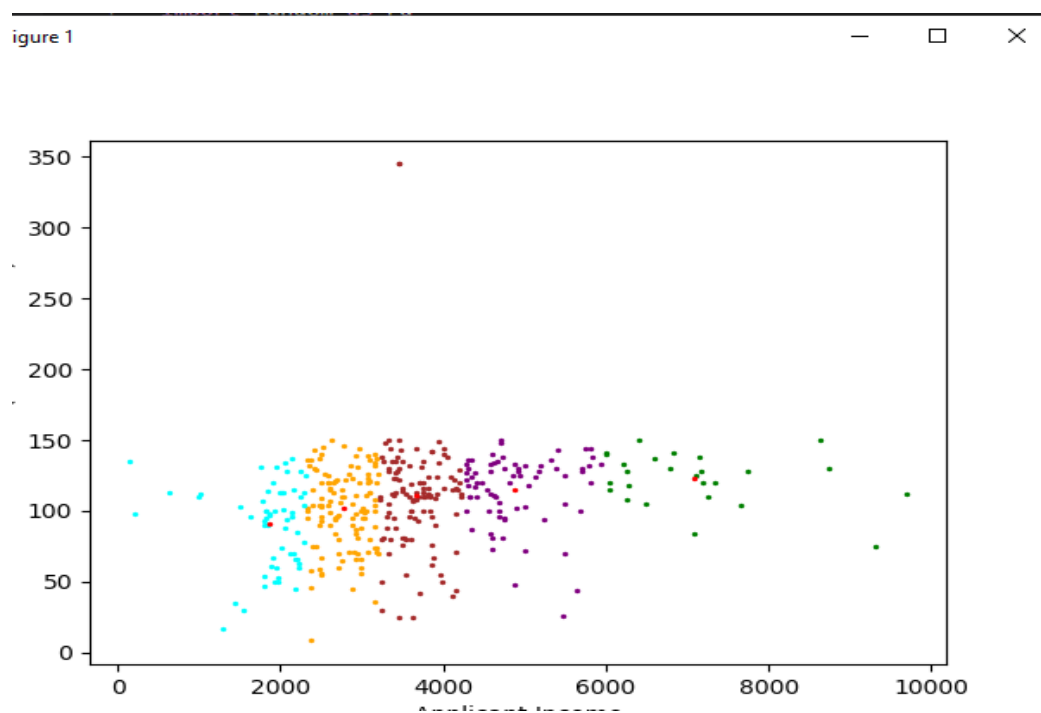


Figure 5. 3 Clusters for k=5

K=6

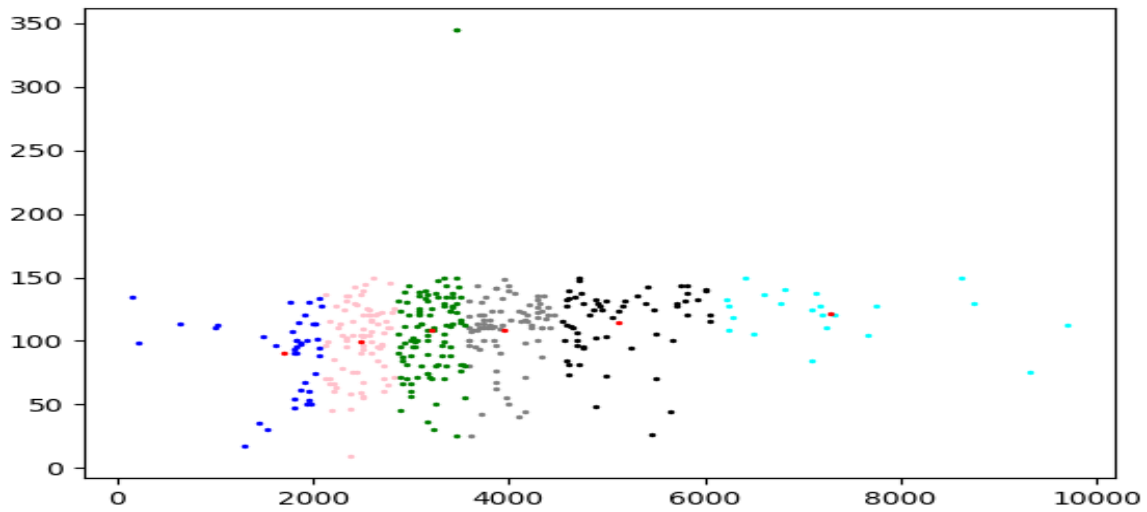


Figure 5. 4 Clusters for k=6

To calculate optimum value of K, Elbow method was used:-

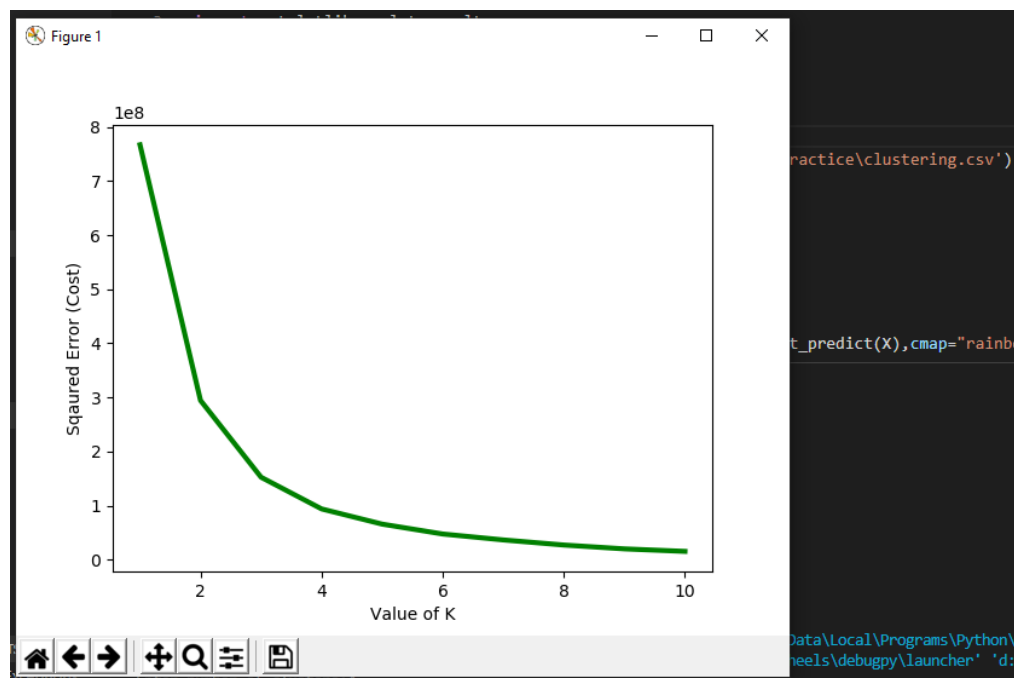


Figure 5. 5 Graph for elbow method showing decreasing SSE

Average Silhouette method is also use to know optimum value of K:-

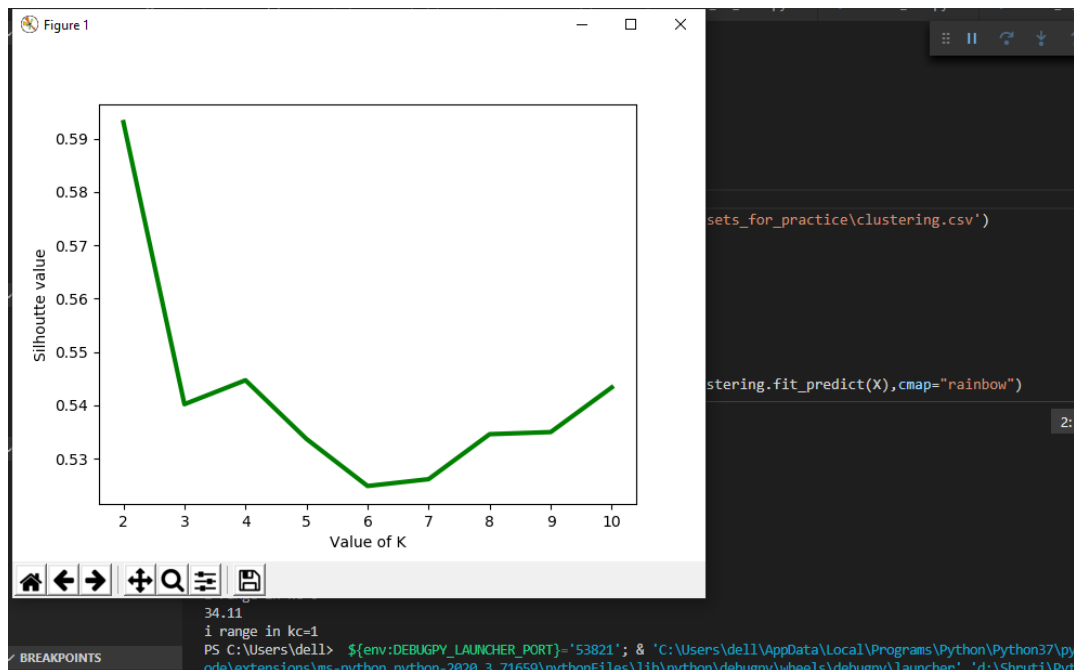


Figure 5. 6 Graph for Silhouette score

On applying Needleman-Wunsch algorithm to 7 protein sequences (samples) the following similarity score was obtained:-

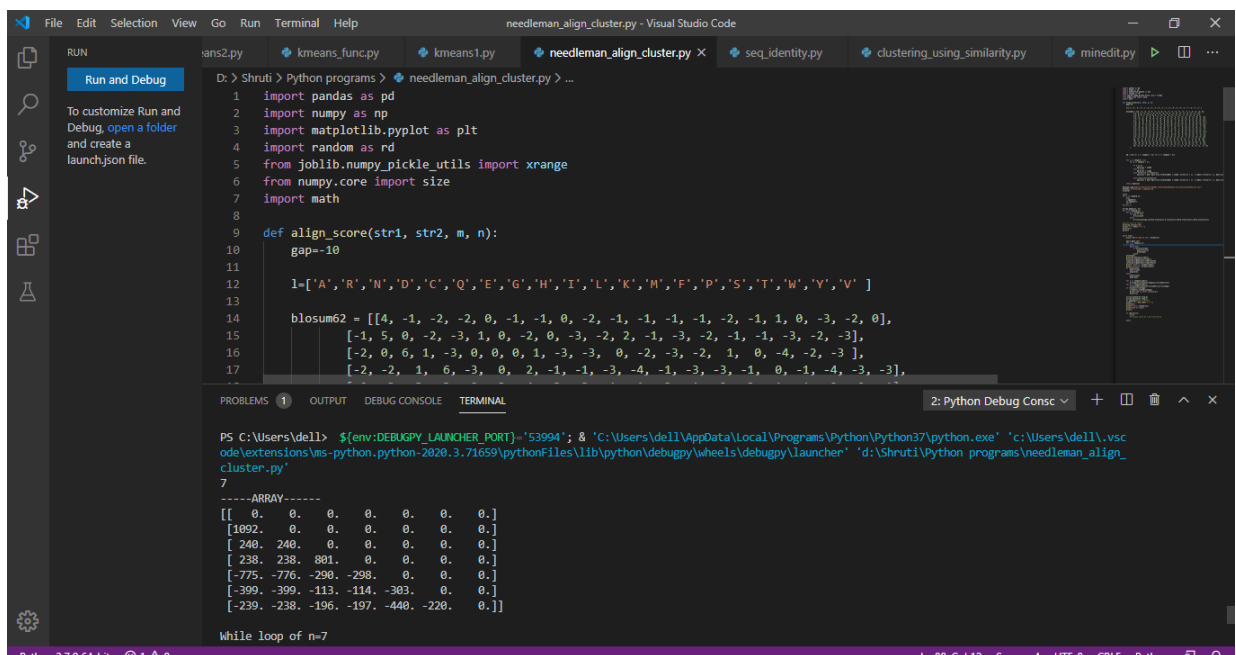


Figure 5. 7 Cluster label for sample data

Clustering of proteins:-

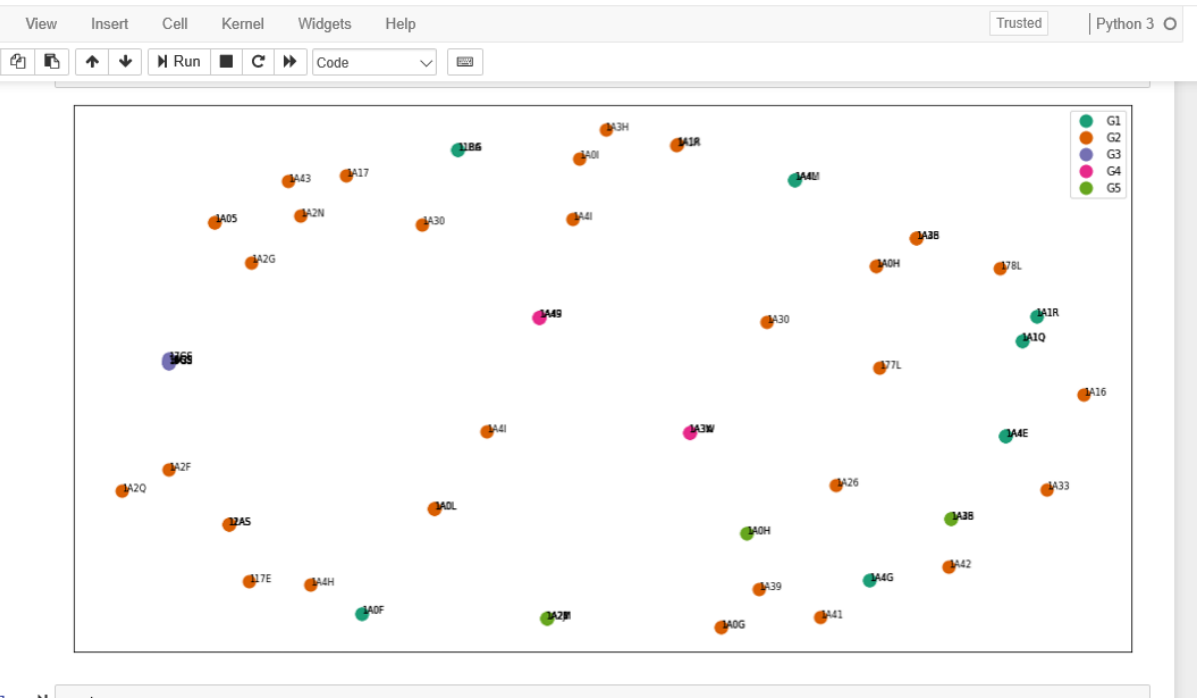


Figure 5. 8 Cluster of 100 protein sequences using Kmeans (Dimension reduced)

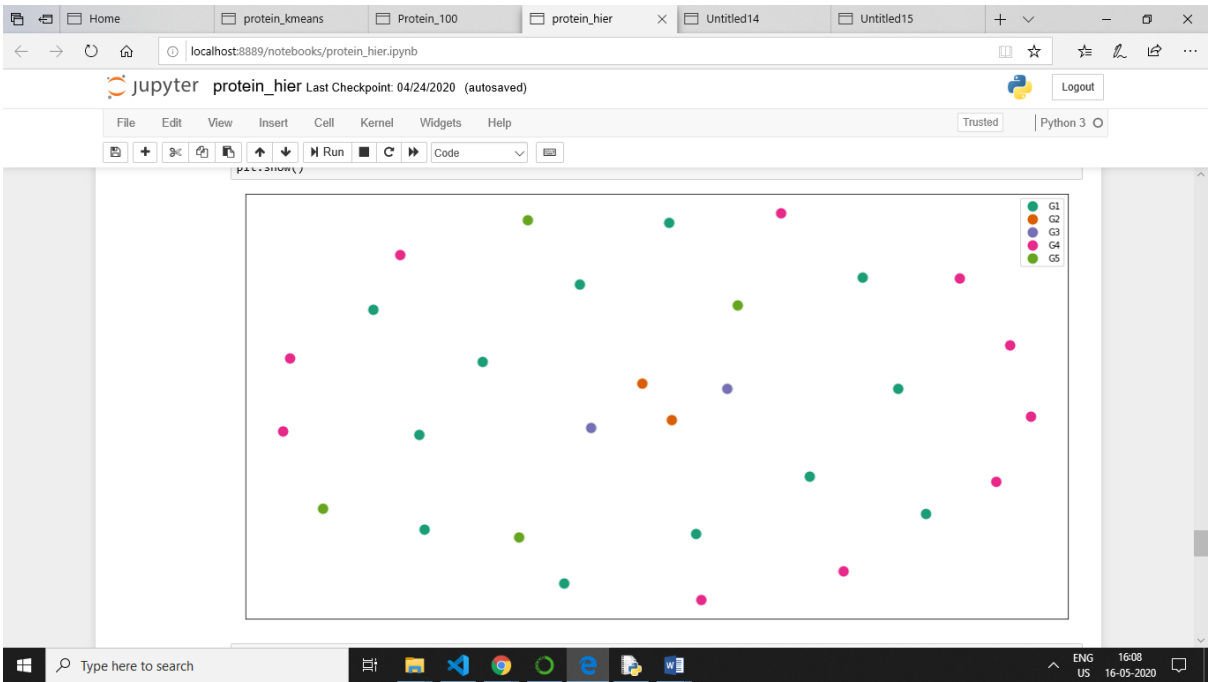


Figure 5. 9 Spectral Clustering

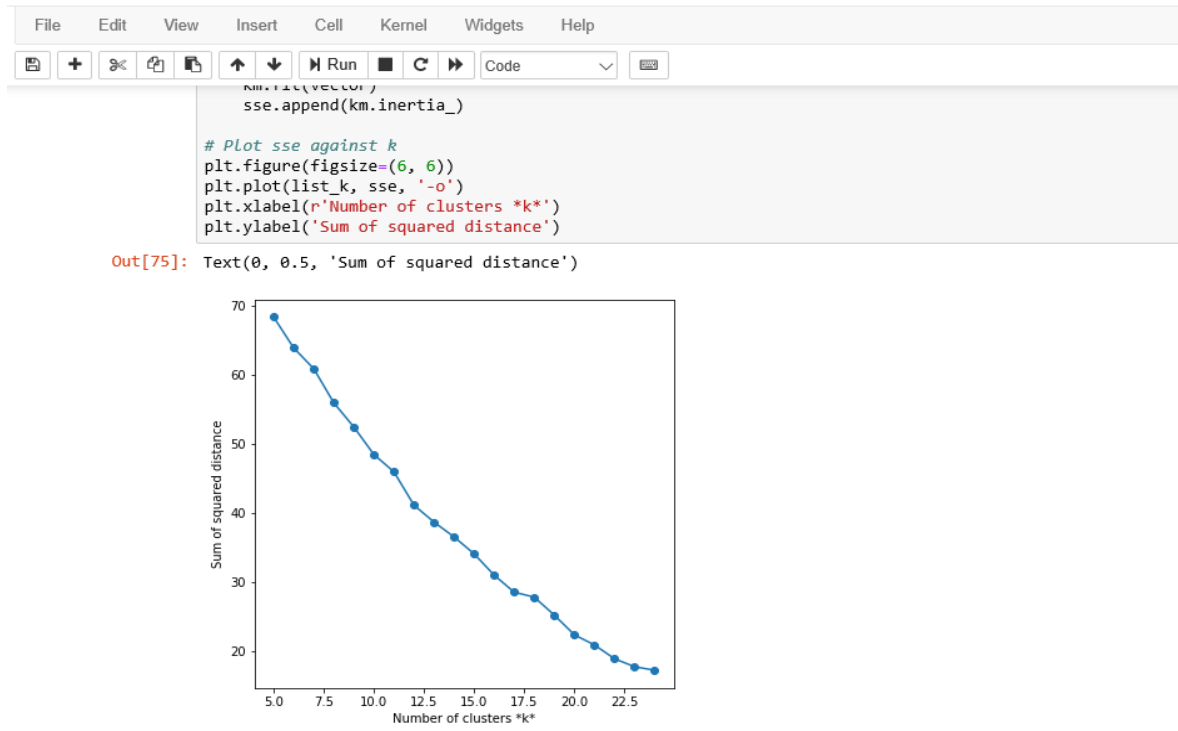


Figure 5.90 Decreasing SSE

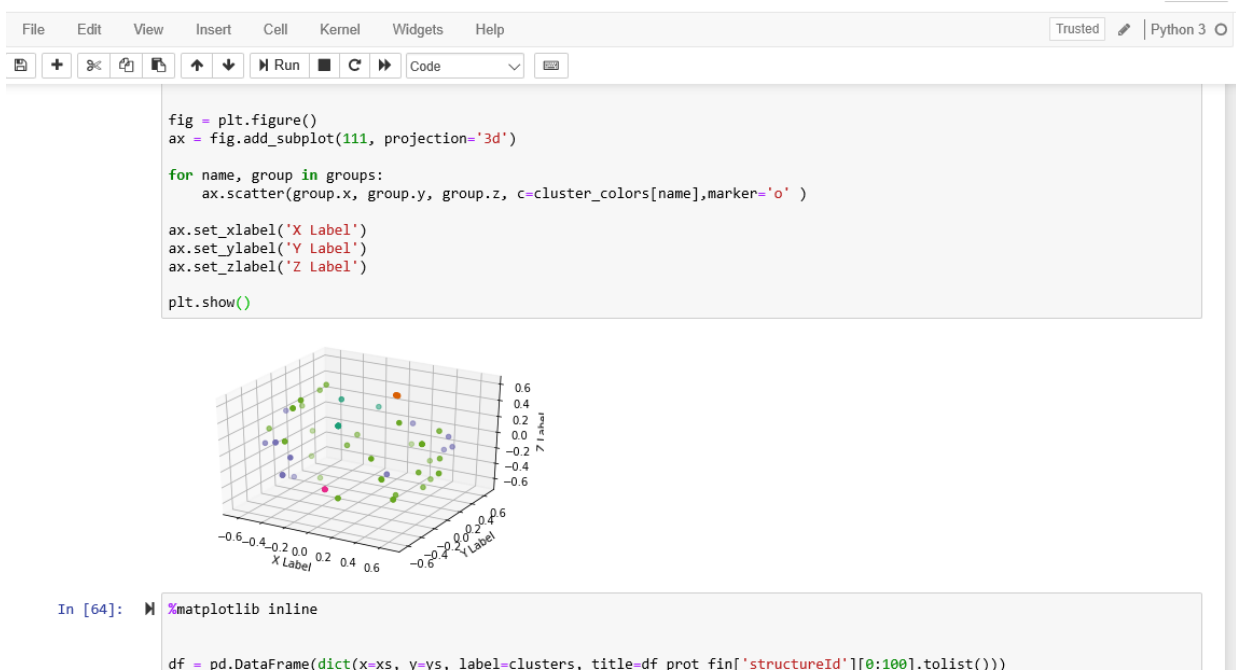


Figure 5.11 3d plot of kmeans with 100 proteins

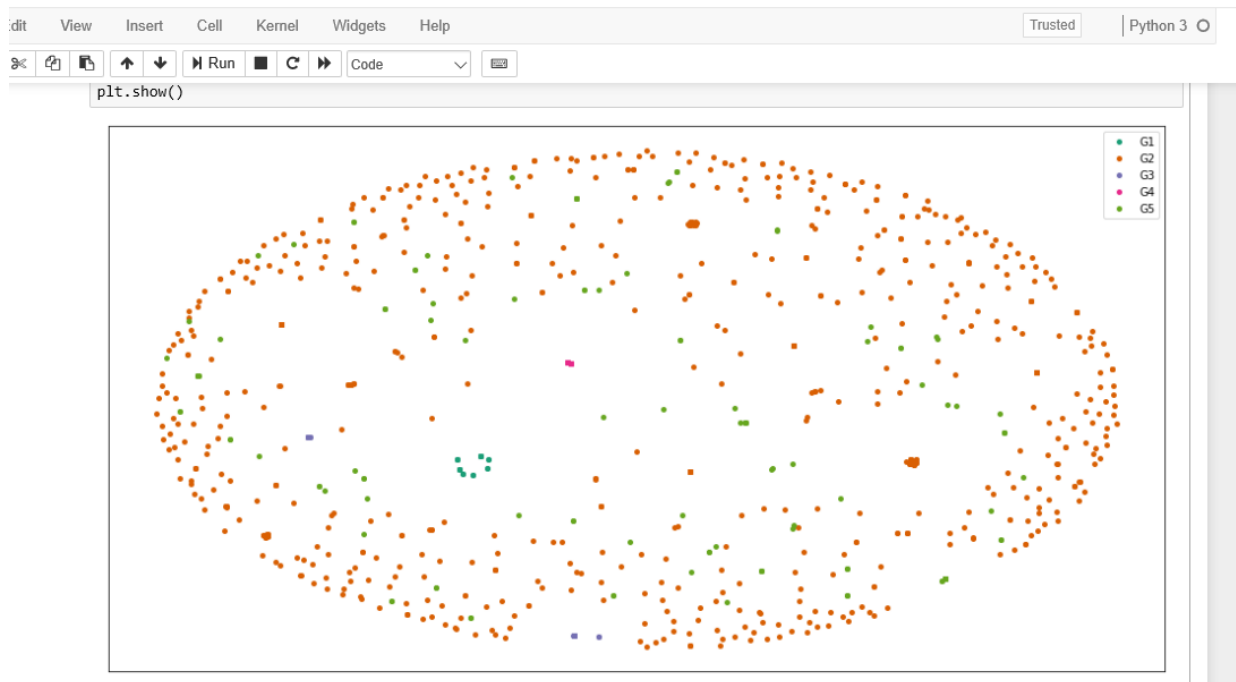


Figure 5. 12 Kmeans for 1000 proteins

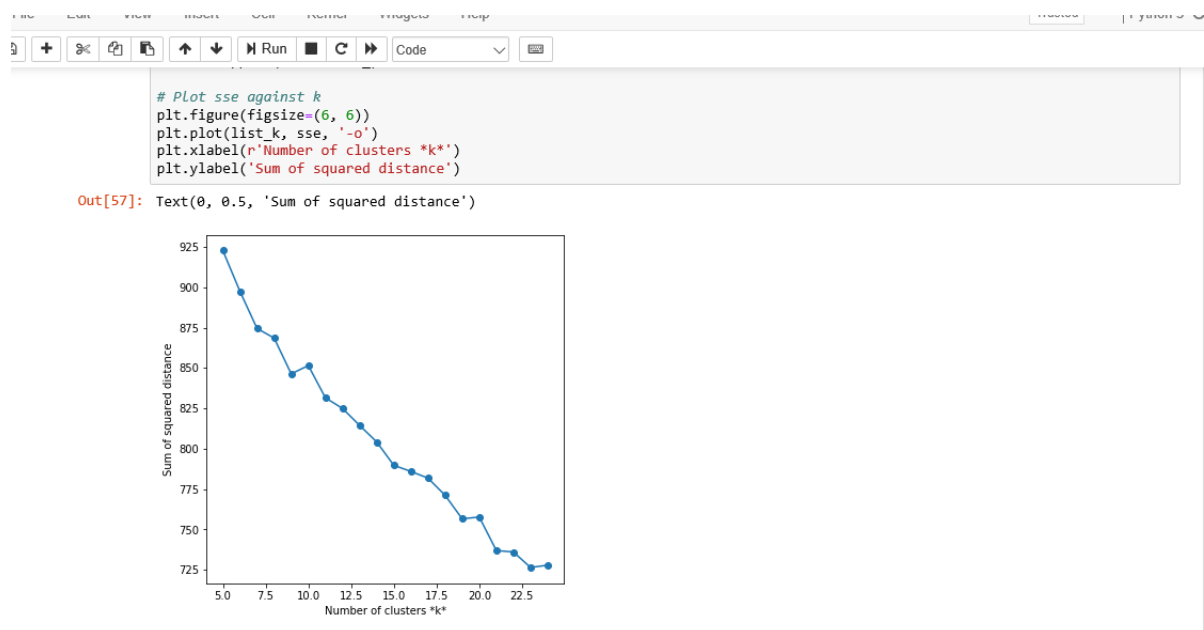


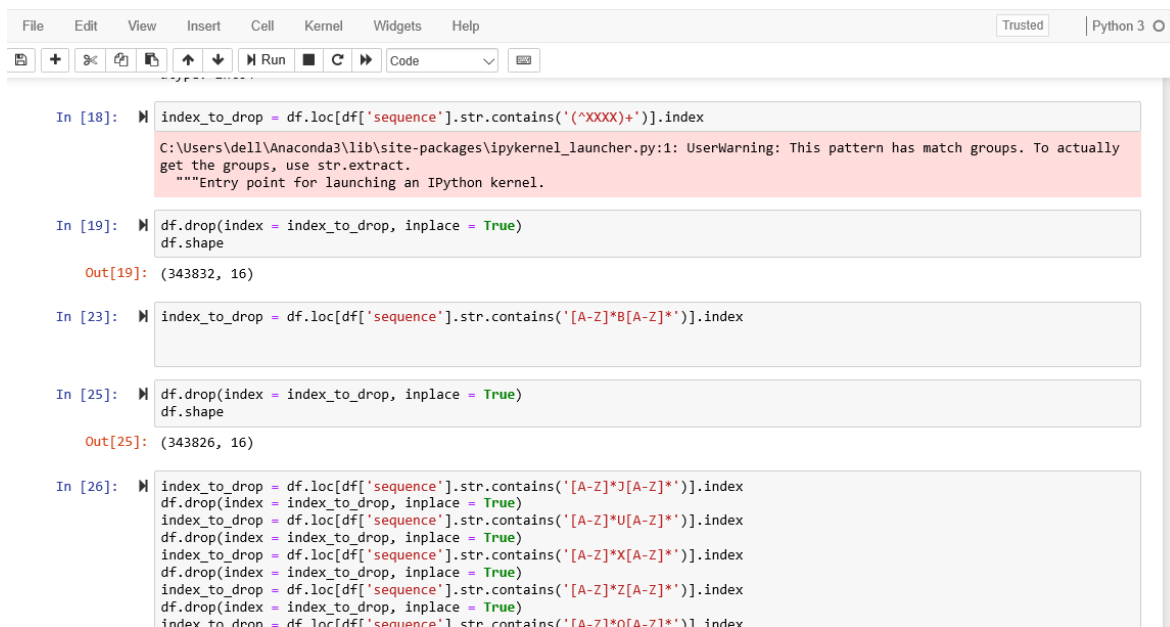
Figure 5. 13 Decreasing SSE for 1000 entries

Chapter - 6

Results and Discussion

Various Graphs were plotted using Matplotlib package offered by python to visualize the data. Since the features obtained are in multi-dimensional, to represent them scaling was performed.

On analyzing the data set some outlier entries were obtained which did not contain the letters in the sequence belonging to the set of 20 amino acids. These sequences were searched for and removed in the initial stage of pre-processing.



```
In [18]: index_to_drop = df.loc[df['sequence'].str.contains('[^A-Z]*').index
C:\Users\dell\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: This pattern has match groups. To actually
get the groups, use str.extract.
"""Entry point for launching an IPython kernel.

In [19]: df.drop(index = index_to_drop, inplace = True)
df.shape
Out[19]: (343832, 16)

In [23]: index_to_drop = df.loc[df['sequence'].str.contains('[A-Z]*B[A-Z]*').index

In [25]: df.drop(index = index_to_drop, inplace = True)
df.shape
Out[25]: (343826, 16)

In [26]: index_to_drop = df.loc[df['sequence'].str.contains('[A-Z]*J[A-Z]*').index
df.drop(index = index_to_drop, inplace = True)
index_to_drop = df.loc[df['sequence'].str.contains('[A-Z]*U[A-Z]*').index
df.drop(index = index_to_drop, inplace = True)
index_to_drop = df.loc[df['sequence'].str.contains('[A-Z]*X[A-Z]*').index
df.drop(index = index_to_drop, inplace = True)
index_to_drop = df.loc[df['sequence'].str.contains('[A-Z]*Z[A-Z]*').index
df.drop(index = index_to_drop, inplace = True)
index_to_drop = df.loc[df['sequence'].str.contains('[A-Z]*O[A-Z]*').index
df.drop(index = index_to_drop, inplace = True)
```

Figure 6. 1 Dropping the malicious entries

On Applying Keyword based similarity using Tf_Idf Vectorizer a vector is obtained which has 9236 feature based on n-grams (trigrams here) :-

```
In [81]: vector.shape
Out[81]: (99, 9236)

In [82]: type(vector)
```

Figure 6. 2 Shape of vector

On using Tf-Idf vectorizer and K-means algorithms for clustering protein sequences the following cluster labels were obtained:-

```
In [68]: from sklearn.cluster import KMeans
num_clusters = 5
km = KMeans(n_clusters=num_clusters)
%time km.fit(vector)
clusters = km.labels_.tolist()

Wall time: 2.37 s

In [69]: print(clusters)

[1, 1, 1, 0, 0, 0, 0, 2, 2, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 1, 0, 0, 1, 1, 4, 1, 4, 1, 1, 1, 1, 1, 1, 1, 0,
0, 0, 0, 0, 1, 1, 4, 1, 1, 1, 4, 4, 4, 4, 1, 1, 1, 1, 1, 1, 4, 1, 4, 1, 3, 3, 3, 3, 1, 1, 1, 1, 4, 3, 3, 3, 3,
3, 3, 3, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0]
```

Figure 6. 3 Cluster labels

The clusters obtained are represented using MDS scaling to represent it in 2 dimensions. The following plot was obtained for a number of 5 clusters.

It can be seen that the clusters are scattered according to the measure obtained by MDS :-

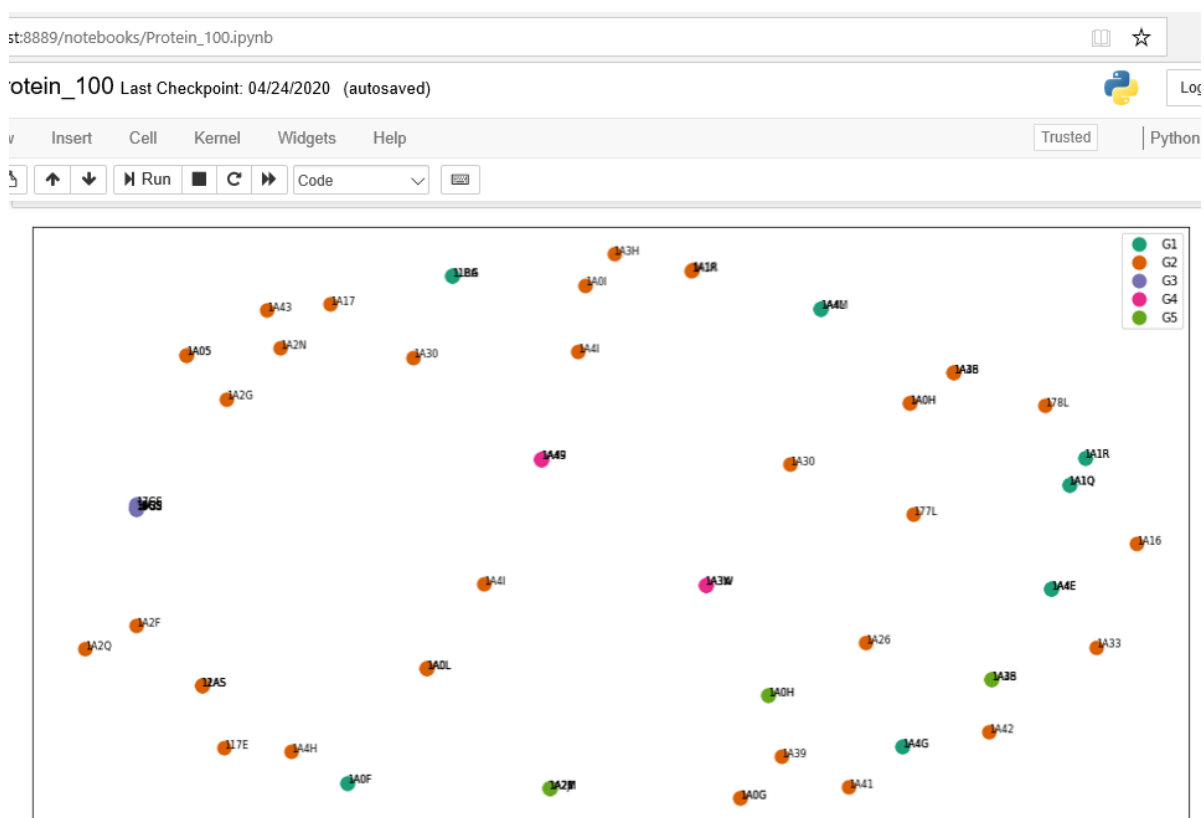


Figure 6. 4 Clusters for kmeans

The silhouette scores obtained are obtained using metrics from sklearn package in python:-

```

for n_clusters in no_of_clusters:

    cluster = KMeans(n_clusters = n_clusters)
    cluster_labels = cluster.fit_predict(vector)

    silhouette_avg = silhouette_score(vector, cluster_labels)

    print("For no of clusters =", n_clusters,
          " The average silhouette_score is :", silhouette_avg)

For no of clusters = 5 The average silhouette_score is : 0.27104083299899573
For no of clusters = 6 The average silhouette_score is : 0.3003884265961843
For no of clusters = 7 The average silhouette_score is : 0.33621843601462953
For no of clusters = 8 The average silhouette_score is : 0.380231372225539
For no of clusters = 9 The average silhouette_score is : 0.4263522092288743
For no of clusters = 10 The average silhouette_score is : 0.43964492510229386
For no of clusters = 11 The average silhouette_score is : 0.46412632352174554
For no of clusters = 12 The average silhouette_score is : 0.5034651558084069
For no of clusters = 13 The average silhouette_score is : 0.5344361905273897
For no of clusters = 14 The average silhouette_score is : 0.5654674208818248
For no of clusters = 15 The average silhouette_score is : 0.5973906578190491
For no of clusters = 16 The average silhouette_score is : 0.6077226637284644
For no of clusters = 17 The average silhouette_score is : 0.616652257435756
For no of clusters = 18 The average silhouette_score is : 0.6677010712660661
For no of clusters = 19 The average silhouette_score is : 0.6754475853050576
For no of clusters = 20 The average silhouette_score is : 0.6985342303231604
For no of clusters = 21 The average silhouette_score is : 0.7164719287120186
For no of clusters = 22 The average silhouette_score is : 0.7285208092819123
For no of clusters = 23 The average silhouette_score is : 0.7468955393675443
For no of clusters = 24 The average silhouette_score is : 0.7323052909978273

```

Figure 6. 5 Silhouette score for kmeans

For next clustering Hierarchical clustering was used and both feature vector (using keyword similarity) and then alignment based methods can be used. For alignment based methods a similarity matrix is used. The dendrograms obtained are as follows:-

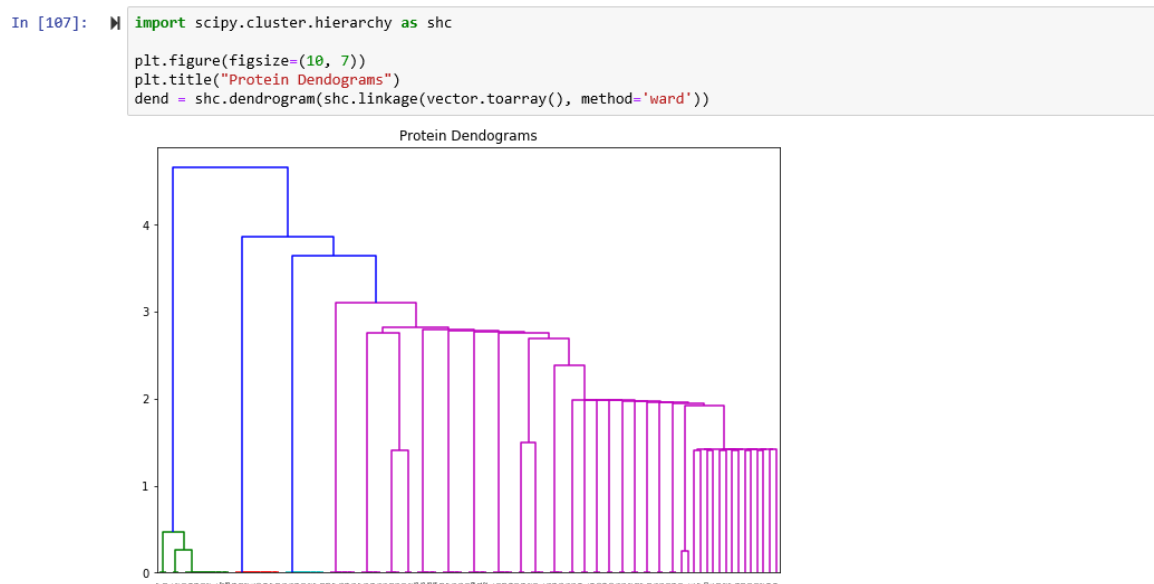


Figure 6. 6 Dendrogram for keywords based clusters

```
In [169]: import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Protein Dendograms")
dend = shc.dendrogram(shc.linkage(arr_for_dendrogram, method='ward'))
```

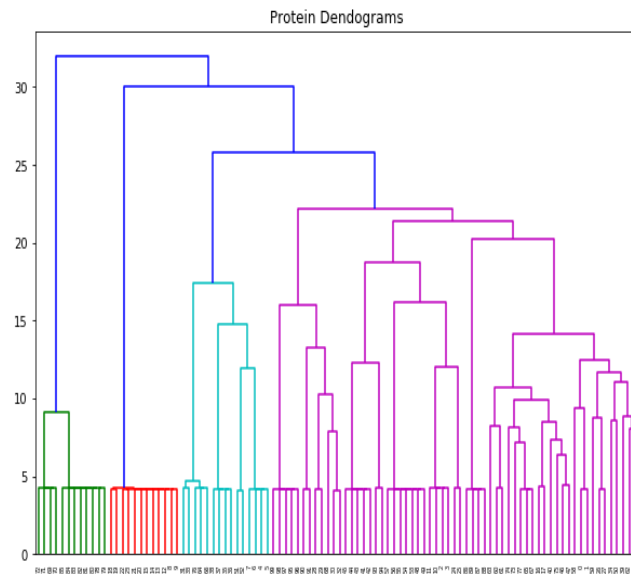


Figure 6. 7 Dendrogram for similarity matrix

For evaluating the clusters formed Silhouette Score is used :-

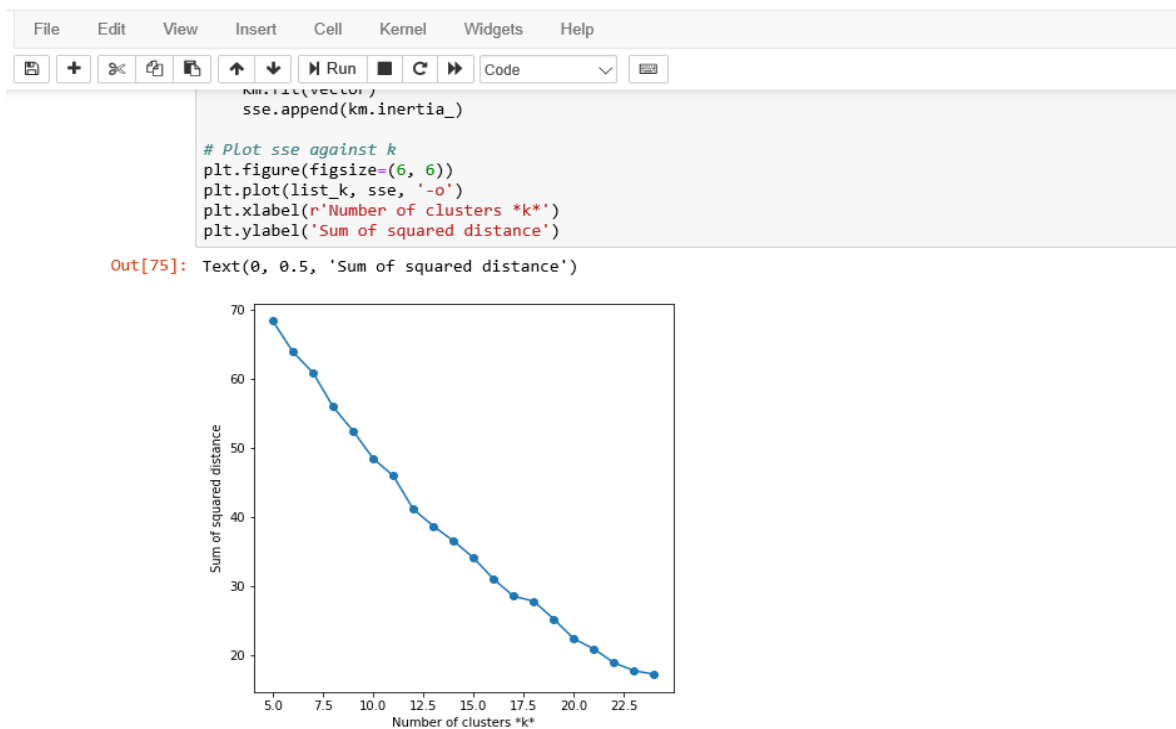


Figure 6. 8 Silhouette score plot

Chapter - 7

Summary and Conclusion

Various Clustering algorithm were implemented along with alignment algorithm. For alignment of sequences dynamic programming was used by implementing Needleman Wunsch Algorithm and Smith Waterman Algorithm. The scoring matrices that were used are BLOSUM-62 and PAM-250. Initially seven sequences were used, which were myoglobins in FASTA format. They were clustered on the basis of similarities. Then Insulin sequences were added to the dataset and results changed in accordance to that. Insulins were grouped together and myoglobins together. for this hierarchical and spectral clustering from python libraries were implemented by taking pre-defined similarity matrix as the measure. Data Stream clustering algorithms are yet to be implemented on protein sequences. Stream algorithm given by Guha generates interesting results. Also K means when implemented gave different clusters for different values of K. K means was first tried on mathematical data.

CluStream Algorithm uses Cluster feature trees to cluster stream data. It becomes tedious task generating CF trees for protein sequences. Also Model based pre-processing on sequences is more complicated in calculating than the alignment based dynamic programming.

I have also studied various Research papers on Data stream Clustering and studies methods and tools for sequence alignment. Also I read materials on sequence clustering algorithms. The data stream clustering problem requires a process capable of partitioning observations continuously while taking into account restrictions of memory and time. In the literature of data stream clustering methods, a large number of algorithms use a two-phase scheme which consists of an online component that processes data stream points and produces summary statistics, and an offline component that uses the summary data to generate the clusters. An alternative class is capable of generating the final clusters without the need of an offline phase.

Among the most useful computer-based tools in modern biology are those that involve sequence alignments of proteins, since these alignments often provide important insights into gene and protein function.

Chapter - 8

Future Scope

There are other methods that can be used to calculate the similarity of protein sequences which are based on keywords and suffix trees and probabilistic models. These tools in combination with stream data clustering can produce different results. Also if the data is considered to be time series data in nature, yet another class of clustering concepts come into picture which can produce interesting and useful results.

Multiple Sequence alignment concept can be used to calculate similarity and speed up the clustering process. It is yet left to compute the efficiency of different algorithms and increase the size of database which is required.

Appendices

A. Proteins and their representation

A1. Amino Acids

Amino acid	Abbreviation	Single letter abbreviation
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamine	Gln	Q
Glutamic acid	Glu	E
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V
Pyrrolysine	Pyl	O
Selenocysteine	Sec	U
Glutamic acid or Glutamine	Glx	Z

A2. BLOSUM 62 Matrix

[illegible]

A variety of BLOSUM (BLOcks SUBstitution Matrix) matrices are available, whose utility depends on whether the user is comparing more highly divergent or less divergent sequences. The BLOSUM62 matrix was developed by analyzing the frequencies of amino acid substitutions in clusters of related proteins. Within each cluster, or block, the amino acid sequences were at least 62% identical when two proteins were aligned. Investigators computationally determined the frequencies of all amino acid substitutions that had occurred in these conserved blocks of proteins. They then used this data to construct the BLOSUM62 scoring matrix for amino acid substitutions. The BLOSUM62 score for a particular substitution is a log-odds score that provides a measure of the biological probability of a substitution relative to the chance probability of the substitution.[14]

A3. PAM 250 Matrix

A	2																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

PAM matrices or the Percent Accepted Mutation (or Point Accepted Mutation, or Probability of Accepted Mutation...) are based on sound evolutionary principles, but original matrices were calculated in 1978. It used reconstruction of ancestral states of 34 superfamilies on 71 groups of sequences that were at least 85% similar. Based on a total of 1572 changes. These are very small numbers by modern standards

It is based on a Markov model in which each change is assumed to be independent of previous states. For each protein, trace changes on a phylogenetic tree, it helps to reconstruct ancestral states. A "PAM 250" matrix corresponds to 250% substitution. These sequences should still have about 20% identity. It is usually presented as a log-odds table: Ratio of likelihood of alignment in related sequences divided by likelihood of alignment in unrelated sequences. And then this is converted to a base two logarithm.

A4. FASTA file format

In bioinformatics and biochemistry, the FASTA format is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes. The format also allows for sequence names and comments to precede the sequences.

A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line (define) is distinguished from the sequence data by a greater-than (">") symbol at the beginning. It is recommended that all lines of text be shorter than 80 characters in length. An example sequence in FASTA format is:

```
>P01013 GENE X PROTEIN (OVALBUMIN-RELATED)
      QIKDLLVSSSTDLDLTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLP
      KMKILELPPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTWETNPNTMEKRRVKVYLPQMKIEEKYNL
      TSVLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESQFRAD
      HPLFLIKHNPTNTIVYFGRYWSP
```

Blank lines are not allowed in the middle of FASTA input.

Sequences are expected to be represented in the standard IUB/IUPAC amino acid and codes, with these exceptions: lower-case letters are accepted and are mapped into upper-case; U and * are acceptable letters (see below). Before submitting a request, any numerical digits in the query sequence should either be removed or replaced by appropriate letter codes (e.g., X for unknown amino acid residue).

The IUB/IUPAC amino acid codes are:

A alanine	P proline
B aspartate/asparagine	Q glutamine
C cystine	R arginine
D aspartate	S serine
E glutamate	T threonine
F phenylalanine	U selenocysteine
G glycine	V valine
H histidine	W tryptophan
I isoleucine	Y tyrosine
K lysine	Z glutamate/glutamine
L leucine	X any
M methionine	* translation stop
N asparagine	- gap of indeterminate length

NOTE:

1. U in protein sequences is replaced by X first before the search since it is not specified in any scoring matrices.
2. PolyPhen will not accept "-" in the query. To represent gaps, use a string of N or X instead.

Bibliography and References

- [1] Adi L. Tarca, Vincent J. Carey, Xue-wen Chen, Roberto Romero, Sorin Draghici, “Machine Learning and Its Applications to Biology”, A tutorial in PLoS Computational Biology, vol.3, June 2007.
- [2] Davide Chicco, “Ten quick tips for machine learning in computational biology”, BioData Mining, 2017.
- [3] Pavlopoulos GA, “How to cluster protein sequences: tools, tips and commands”, MOJ Proteomics Bioinform. 2017;5(5):158–160. DOI: 10.15406/mojpb.2017.05.00174.
- [4] Antje Krause, Jens Stoye, Martin Vingron, “Large Scale Hierarchical clustering of protein sequences”, BMC Bioinformatics 2005,6:15. DOI 10.1186/1471-2105-6-15.
- [5] Charu Aggarwal, Chandan Reddy, *Data Clustering*, CRC Press, Taylor and Francis Group, 2014.
- [6] A. Sharaf Eldin, S. AbdelGaber, T. Soliman, S. Kassim, and A. Abdo, “A Comparative Study of Protein Sequence Clustering Algorithms”, DOI: 10.1007/978-481-9112-3_63.
- [7] Scikit Learn Modules website [Online] Available: <https://www.scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster> [Accessed 12 January, 2020]
- [8] pandas documentation website [Online] Available: <https://www.numpy.org/> [Accessed 2 February, 2020]
- [9] numpy documentation website [Online] Available: <https://www.pandas.pydata.org/> [Accessed 2 February, 2020]
- [10] matplotlib documentation website [Online] Available: <https://www.matplotlib.org/> [Accessed 2 February, 2020]
- [11] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan. Clustering data streams. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science. IEEE Computer Society, 2000.
- [12] Dan Gusfield, *Algorithms on Strings, Trees and Sequences*, 1997

- [13] A. A. Puntambekar, *Software Engineering and Project Management*, Technical Publications, February 2016

- [14] BLOSUM62 scoring matrix for amino acid substitutions [Online]
Available: [https://bio.libretexts.org/Bookshelves/Cell and Molecular Biology/](https://bio.libretexts.org/Bookshelves/Cell_and_Molecular_Biology/)
[Accessed 2 March 2020]