Identifying similarities between text and descriptions in the Gene Ontology

Anonymous
Candidate number: 2401041

Submitted for the Degree of Master of Science in

Data Science and Analytics

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

June 16, 2014

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 8789

Student Name: Anonymous

Candidate Number: 2401041

Date of Submission: 28 August 2024

Signature:

Abstract

This dissertation explores the application of the machine learning models in the prediction of the Gene Ontology (GO) information. Gene Ontologies provide a structured vocabulary necessary for the categorisation of genes and their products across various organisms, which is facilitated by the efforts of the Gene Ontology Consortium. This project explores various techniques that can be used to annotate gene functions to allow the comparison across different species using the results.

This project involves analysis of Gene Ontology dataset and using machine learning models such as *K*-Nearest Neighbors, Support Vector Machines, and Random Forests. We conduct a series of experiments to predict the GO namespace and 'is a' relationship and optimise and analyse the model performance. This project delves into multi-class and multi-label classification and introduces the tool MultiGOClassifier which can be used to predict GO term information based on textual inputs. This project aims to enhance our understanding of machine learning applications in genomic research and provide a new tool which can be useful for the biomedical researchers. The structure of the code and usage has also been documented, which and be used and extended for future research.

Contents

1 Introduction

Gene Ontologies provide a structured vocabulary, for categorisation of genes and their products across various organisms. The Gene Ontology Consortium, a group of scientists in the disciplines of biology and computer science, has been pivotal in coordinating the efforts and ensuring a standardized approach to annotating gene products for multiple organisms. The Gene Ontology (GO) knowledgebase is a critical resource in supporting biomedical research, significantly contributing to tens of thousands of scientific studies [1]. GO allows researchers to create structured GO annotations for the GO annotation dataset, GO allows for comparisons of gene functions across different species and is commonly used for the analysis of genomic data [2]. The researchers often use tools which help to annotate GO terms [3].

In this project, we aim to predict GO term information by conducting several experiments exploring various machine learning models. This project will take us through various machine learning models and the performance metrics being used as well. We start by exploring the GO dataset and determining which information is predictable, proceeding to explore, clean and transform the data as needed. Further, we will be conducting experiments in order to predict this information to observe the performance of different models such as K-Nearest Neighbours, Support Vector Machines and Random Forest.

We explore multi-label and multi-class classification problems with different labels. Finally, we aim to implement our version of GOCat (Gene Ontology Categoriser) tool called MultiGOClassifier, which allows us to make GO predictions based on some input text. This project will not only provide insights into the applicability of different machine learning models in genomic research, but also familiarises us to the use of computational techniques analysing genomic information.

2 Conceptual Framework of Gene Ontologies

An Ontology is a formal representation of knowledge of a domain or an area of reality as a set of concepts and the relationships between them in a structured way. The Gene Ontology (GO) is an initiative by the Gene Ontology Consortium to develop and use a structured, standardized controlled vocabulary for gene and gene products for any organism [4]. It has become important to categorise these entities since there has been an exponential growth of biological information, which continues to grow, which makes the GO necessary to manage and interpret large amount of biological data [5]. This vocabulary, which describes a particular attribute of a gene, or a gene product is referred to as 'GO term' [6]. The Gene Ontology includes three ontologies that describe molecular function, biological processes and cellular component [5], [7]. In addition to providing the standardised vocabulary, GO knowledgebase comprises a database which provides access to annotations, query applications, the specialized datasets that have been generated as a result of employing these vocabularies to annotate gene and gene products, and GO Causal Activity models (GO-CAMs) [5], [8]. The representatives studying the genome of three model organisms using the databases - FlyBase (*Drosophila Melanogaster*), *Saccharomyces* Genome Databases (SGD) and Mouse Genome Databases (MGD), decided to work together on a common classification of gene function in 1998, leading to the beginning of The Gene Ontology Consortium [1], [7].

The Gene Ontology provides two key resources that are widely utilized in biomedical research:

- The GO ontology
- The corpus of GO Annotations

Together, the ontology and annotations provide a comprehensive model of biological systems [9].

2.1 The GO Ontology

The first resource in the Gene Ontology is the GO ontology itself, which is a comprehensive logical framework that describes the full complexity of biological processes. This ontology includes 'classes' or 'terms' that define various biological functions, the pathways that carry out different biological programs and the cellular locations where these occur; including the relationships that indicate how these terms are related to other terms [1].

2.1.1 The Three Ontologies

GO consists of three aspects or categories – the *biological process,* the *molecular function* and the *cellular component*.

Molecular Function - This ontology describes the activities of a gene product that take place at a molecular level such as catalytic or binding activities, they do not specify where, when or in what context the activity is being performed [10]. Examples of the broad molecular function terms are, 'enzyme', transporter' or 'ligand'; the examples of narrow molecular functions are 'adenylate cyclase' or 'Toll receptor ligand' [1], [4].

Biological Process - This ontology describes the biological goals that have been accomplished by the molecular functions; a process is accomplished thorough one or more ordered assemblies of molecular functions [4], [5]. Examples of the broad biological process terms are, 'cell growth and maintenance' or 'signal transduction'; the examples of specific biological terms are 'pyrimidine metabolism' or 'cAMP biosynthesis' [1], [4].

Cellular Component - This ontology describes the location where the gene product is active in the cell. The examples of cellular component terms are, 'ribosome' or 'proteasome' which specify where multiple gene products can be found, as well as terms like 'nuclear membrane' or 'Golgi apparatus' [4].

2.1.2 Structure of the Ontologies

The Gene Ontology structure is hierarchical, and can be described in terms of a graph, where each GO term is a node, and they are connected to each other in a Directed Acyclic Graph (DAG), and each "child" term can be associated with more than one "parent" term, or each "parent" can be associated with more than one "child" term [1], [5].

Relationships

The relationships between GO Terms can be described using below relations:

is a: This forms the most basic and common type of GO term relationship; suppose for the GO terms 'A' and 'B', if we say, "A *is a* B", it indicates that A is a subtype of the term B. Example, *mitotic cell cycle* is a *cell cycle* [1], [11].

part of: This relation is used to represent part-whole relationships from the perspective of the child, where a term is a part of another term, it is used only if between A and B, B is necessarily a part of A: presence of B implies the presences if A; however, presence of A does not imply presence of B [1], [11].

has part: This relation is used to represent part-whole relationships from the perspective of the parent, where a term always has another term as its part. Between terms A and B, the presence of A implies that B will always exist; however, the presence of B does not for certain indicate that term A exists or not [1].

regulates: When one process directly affects the manifestation of another process, this relation is used to mean necessarily-regulates, i.e. if A regulates B, then whenever A exists it always regulates B, but B may not always be regulated by A [1], [11].

2.2 GO Annotations

GO Annotations are the relationships between a gene or a gene product with a GO term, which is supported by evidence derived from an experimental study. GO Annotation is a statement based on evidence that relates a gene product to a GO term. The GO 'evidence code' is used to indicate the type of evidence for a given association between gene and a GO term [12]. A gene or gene product can be associated with more than one GO term, i.e. it can be linked to multiple biological processes, molecular functions and cellular components [5]. Annotations can be made using two methods – manually by trained curators which involves detailed reading of scientific literature and by automated methods which predict gene

functions [13], [14]. The development and the use of Gene Ontology terms for GO Annotations has increased extensively [15]. Biological validations are carried out to ensure accuracy and completeness which may reveal errors in either the ontology structure or annotations [11]. The GO Consortium has developed measures to access the annotations that have been made by the different groups [15], [16]. GO Annotations are regularly reviewed and can be modified or removed from the GO knowledgebase if they have been proven invalid by an experiment or if they are revised [8]. GO Annotations are used widely in the analysis of the genomic data such as gene expression studies [17], [18]. "Go annotations are meant to reflect the most up-to-date view of a gene products role in biology" [1].

2.3 Applications

The GO has been used for *Gene expression studies* which help interpret the results of experiments such as microarrays and RNA-seq data, in which the mapping between GO term and gene expressions can highlight which biological processes or molecular functions are altered [15], [19].

Gene function prediction is possible based on patterns of annotation, for example, if the annotations of two attributes frequently appear together in a database, then a gene possessing one attribute is likely to possess the other attribute as well [20]. Some of the other applications have been for improving disease gene prioritization using the semantic similarity of Gene Ontology terms [21], and to compare genes across different species to help understand functional evolution of genomes [22]. GO terms are utilized in text-mining systems like Textpresso which facilitates semantic searches and to categorize biological literature [23].

The Gene Ontology represents a significant step forward in genomics by offering a structured vocabulary for categorizing genes and gene products across all organisms. By differentiating molecular functions, biological processes and cellular components, GO enables an in-depth understanding on genomic data. GO has numerous practical applications, including gene expression studies, gene function prediction and comparative genomics making it important for advancement of biological research. As genomic data grows, the continued refining and implementation of Gene Ontology will be crucial for interpreting this complex information.

3 Datasets in Gene Ontology

3.1 The Gene Ontology Dataset

The Gene Ontology Dataset provides the vocabulary that define and describe the biological processes, molecular functions and the cellular components of the gene and gene products for different organisms. The Gene Ontology website offers a go-basic.obo file which contains only the essential data that is necessary for annotation tools, and the data is filtered such that the graph is acyclic and ensures that there are no recursive relationships and maintains simplicity [24]. It includes the relationships *is a, part of, regulates, negatively regulates* and *positively regulates;* and it does not include the relationships that cross the 3 main GO hierarchies (biological processes, molecular functions and cellular components) [24].

3.1.1 Format

The '.obo' in go-basic.obo file stands for Open Biomedical Ontologies. Biological ontologies such as the Gene Ontology are represented in this format [2].

3.1.2 Structure and Contents

An OBO document is structured in two main parts, which is the header frame, followed by zero or more entity frames [25].

3.1.2.1 Header Frame

The header frame is the first section if the OBO file and includes the meta data of the ontology. It provides various tags with information about the ontology [25]. Below are some of the tags which are included in the header [26]:

Required tag:

- *format-version:* specifies the OBO format version.

Optional tags:

- *data-version:* specifies the version of the current ontology.
- *subsetdef:* specifies the description of subsets within the ontology.
- *synonymtypedef:* specifies the user-defined synonym type.
- *default-namespace:* specifies the default namespace for terms in ontology.
- *ontology:* specifies the ID space of the current ontology. For GO, the value of this field will be "go".
- *remark:* General comments for this file

This metadata is crucial for understanding the context and structure of the ontology that follows.

3.1.2.2 Entity Frame

Following the header, the file contains zero or more entity frames, each describing a term, relationship type (typedefs) and instances. Each entity frame starts with a required 'id' tag that announces the object to which all the tags in the frame refer to i.e. the unique id of that entity [26].

3.1.2.2.1 Term Frame

The term frame defines the individual terms/nodes in the ontology. Some of the tags that are included in the term frames are [25], [26] :

Optional tags:

- *name:* specifies the name of the term.
- *namespace:* specifies which of the main categories the GO term belongs to i.e., biological process, molecular function or cellular component.
- *def:* Defines the current term
- *synonym:* specifies alternative name or description.
- *comment:* specifies a comment for the current term.
- *is_a:* Links to the parent term describing the relationship between one term and another.

3.1.2.2.2 Typedef Frame

The typedef frame defines the relationship type that that can be used in the term definitions to elaborate on the relationships between terms, and support almost all tags as term frame [26].

3.1.2.2.3 Instance Frame

Instance frames are used to represent the spatiotemporal particulars in an ontology and are not commonly used [26].

3.1.3 Challenges and Considerations

The ontology is regularly updated and expanded as biological knowledge accumulates, which is why the users must ensure that the latest version us being used to maintain compatibility an accuracy in its applications [1]. Since the Gene Ontology is comprehensive, understanding and using the relationships and information within GO can be complex and may require specialised knowledge.

3.2 The Annotations Dataset

The GO Annotation is a statement about the function of a particular gene which is given by linking the gene or gene product to a GO term [1].

3.2.1 Format

The Gene Ontology Annotations are stored in a GAF (GO Annotation File), which is a tab-delimited text file format that includes details about the gene or gene products and their associated GO terms [1].

3.2.2 Structure and Contents

The annotations are describes using 17 columns mentioned below [1]:

- *DB:* specifies the source of the database.
- *DB Object ID:* unique identifier of the item from the database.
- *DB Object Symbol:* specifies a unique symbol to which the DB object ID is matched.

- *Qualifier:* describes how a gene product relates to a GO term.
- *GO ID:* GO identifier of the term attributed to DB object ID.
- *DB:Reference*: reference supporting the annotation.
- *Evidence Cod*e: type of evidence for the annotation.
- *With (or) From*: additional identifiers.
- *Aspect:* ontology aspect (biological process, molecular function, cellular component).
- *DB Object Name:* name of gene product.
- *DB Object Synonym:* synonym for gene product.
- *DB Object Type:* description of gene product
- *Taxon:* Taxonomic identifier.
- *Date:* specifies the date of annotation
- *Assigned By:* specifies the database which made the annotation.
- *Annotation Extension:* Additional details for an annotation.
- *Gene Product Form ID:* specific form of gene product.

GOA dataset provides annotations for different species that are comprehensive. The Gene Ontology and Gene Ontology Annotation datasets offers combined strength to build a tool like GOCat.

4 GOCat – Gene Ontology Categorizer

4.1 Introduction to GOCat

GO Categorizer (GOCat) is a supervised learning Gene Ontology classifier. Gene Ontology Categorizer is a tool that utilises supervised machine learning and proposes relevant Gene Ontology terms for a biological abstract, which can be then used for the purposes of gene ontology annotation. The GOCat tool aims to provide a functional profile related to the GO terms as the output for a given abstract [27]. There are other GO classifiers with which GOCat has been compared with, dictionary-based classifiers called EAGL and GOPubMed classifier [28].

The increasing volume of biomedical literature makes GOCat necessary to help annotate this vast data, since the traditional method of manual curation can be time-consuming and fail to keep pace with this growing literature [3].

4.2 Methodology

The GOCat is based on the *k*-nearest neighbours' algorithm, where it assigns to an input abstract the GO terms which are present in majority in the *k* most similar instances in the knowledge base [27], [28]. This knowledge base is curated using the Gene Ontology Annotation database and MEDLINE, the National Library of Medicine's database. For an abstract, GOCat identifies the *k* most similar abstracts and assign's the most prevalent GO terms from these *k* abstracts [27]. Using this approach allows the GOCat to help in creating annotations even when accurate terms are not present in the text.

4.3 Comparison with other GO Classifiers

The effectiveness of the GOCat has been demonstrated in comparison with dictionary-based classifiers. A study shows that GOCat outperformed dictionary-based classifiers achieving 65% hierarchical recall for the top 20 outputted concepts [3]. It also shows that the quality of prediction of GO concepts by GOCat continues to improve over time, because of the growing high-quality GO concepts in the GOA, and that the performance of predicting for a new abstract has improved by 50% [3], [29], [30]. The study highlights that the machine learning approaches like GOCat, surpassed the dictionary-based approaches.

The lexical categorizers assign GO descriptors, which are rarely used to generate annotation, GOCat is dependent on [GO; PMID] pairs available in GO annotation dataset [2]. The only limitation of GOCat is its ability to detect only those GO terms that have already been annotated in GOA [28].

5 Background Research

5.1 K-Nearest Neighbours

The K-Nearest Neighbours (KNN) is a technique that is used for both Regression and Classification problems. It has been used for classification in this project. It is a non-parametric, supervised learning algorithm which makes prediction based on closest data points from the training set, without any assumption of the form of the distribution of the data [31]. KNN is an intuitive and non-linear algorithm. It is a part of instance-based learning, since learning in this algorithm consists of simply storing the training data and when a new instance needs to be classified, the stored instances are retrieved and used to classify the new instance [32].

A classification problem aims to predict a class label for an unlabelled input $\boldsymbol{x_0}$ for a dataset of N training records. Given a positive value for K, the KNN algorithm first identifies the K number of data points in the training dataset that are closest to the input $\boldsymbol{x_0}$ [31]. The "nearness" between the $\boldsymbol{x_0}$ and data points is measured using a dissimilarity metric, which influences how data points are grouped or separated. A most widely used dissimilarity (or distance) metric is the Euclidean distance, which is expressed as,

$$d(x, x') = \sqrt{\sum_{i=1}^{n} |x_i - {x'}_i|^2} \tag{1}$$

Euclidean distance is a special case of the Minkowski distance measure when p=2 [33],

$$d_{Mink}(x, x') = \sqrt[p]{\sum_{i=1}^{n} |x_i - {x'}_i|^p} \tag{2}$$

Once the K nearest points based on the distance have been identified, the most frequent label from these K points which are a part of N_0 will be considered as the predicted class for the input $\boldsymbol{x_0}$. The KNN classifier estimates the conditional probability for class j as the fraction of K whose response equals j,

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i=N_0} I(y_i = j) \tag{3}$$

After which, KNN classifies the unlabelled observation $\boldsymbol{x_0}$ to the class with highest probability from eq. (3) [31].

An optimal value of K which gives the best generalisation performance can be determined using cross-validation [34].

5.2 Support Vector Machines

The Support Vector Machine (SVM) is an extension of the Support Vector Classifier and a generalisation of the Maximal Margin Classifier. The fundamental idea of SVM is to find a hyperplane with p - 1 dimensions in p-dimensional space which separates the data points into two distinct classes [31]. The objective of SVM is to maximise the margin between the hyperplane and the nearest data points (of any class) from this hyperplane. The kernel trick can be employed further to solve the non-linear problems.

In two dimensions, the hyperplane is a one-dimensional line. In three dimensions, the hyperplane is a two-dimensional plane. For higher dimensions, the hyperplane is difficult to visualise but follows a p - 1 subspace [31]. p-dimensional hyperplane can be defined as,

$$w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_px_p + w_0 = 0 \tag{4}$$

This equation can also be expressed as below where it is applicable for any dimension,

$$\vec{W}^T x_i + b = 0 \tag{5}$$

Here, $\vec{W}^T$ represents the transpose of weight vector, whose components are coefficients and x_i represents the feature vector and b is the bias term. The points which lie exactly on the hyperplane satisfy the eq. (5).

For a data point, if $\vec{W}^T x_{ij} + b > 0$, then the data point lies on one side of the hyperplane. Whereas, if $\vec{W}^T x_{ij} + b < 0$, then it lies on another side of the hyperplane. Suppose the observations we have fall into two classes $y_i = \{-1, 1\}$.

The shortest distance from the separating hyperplane to its nearest positive and negative point is given as d_+ and d_-. The margin of the hyperplane is $d = d_+ + d_-$. The support vector algorithm looks for the hyperplane with the largest margin [35]. The negative and positive hyperplane passes through the nearest data points on the positive and negative side. If the training data satisfies the below constraints,

$$\vec{W}^T x_{ij} + b \geq +1 \; for \; y_i = +1 \tag{6}$$

$$\vec{W}^T x_{ij} + b \leq -1 \; for \; y_i = -1 \tag{7}$$

Then they can be combined and stated as,

$$y_i\left(\vec{W}^T x_{ij} + b\right) \geq 0 \tag{8}$$

for all $i = 1, \ldots, n$.

The hyperplanes in eq. (6) and (7) are parallel to the hyperplane as described in eq. (5). Suppose we have 2 data points x_1 and x_2, x_1 is a point on negative hyperplane $\vec{W}^T x_{ij} + b = -1$, which has perpendicular distance from the origin,

$$\frac{|-1 - b|}{\|W\|}$$

and x_2 is a point on positive hyperplane $\vec{W}^T x_{ij} + b = +1$, which has perpendicular distance from the origin,

$$\frac{|1-b|}{\|W\|}$$

Then,

$$d_+ = d_- = \frac{1}{\|W\|} \tag{9}$$

$$d = \frac{2}{\|W\|} \tag{10}$$

We can find the pair of hyperplanes which maximise the margin by minimising $\|\boldsymbol{W}\|^2$ [35].

The Observations which lie on these pair hyperplanes are called Support Vectors, and we only need these for prediction and the remaining data points can be ignored.

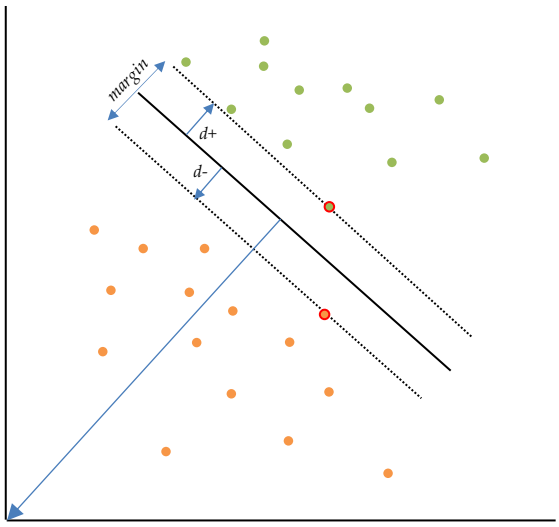

Figure 1. Hyperplane in SVM

The linear support vector classifier can be represented as,

$$f(x) = b + \sum_i^n \alpha_i \langle x, x_i \rangle \tag{11}$$

There are n parameters α_i, i= 1, …, n.

The feature space can be mapped to higher dimensional space by using kernels. The support vector machine has the form [31], [35],

$$f(x) = b + \sum_{i=1}^{N_S} \alpha_i y_i K(s_i, \boldsymbol{x}) \tag{12}$$

Where:

- b is the bias term
- α_i are the Langrage multipliers associated with support vectors
- K is a kernel function that quantifies the similarity of two observations
- y_i are the labels of support vectors
- s_i are the support vectors

The types of kernels that can be used [36]:

- Linear: $K(x, x') = \langle x, x' \rangle$
- Polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d$
- Radial Basis Function (RBF): $K(x, x') = e^{-\gamma \|x - x'\|^2}$

5.2.1 One-Vs-Rest (OVR)

The One-Vs-Rest approach that can be used to extend SVM for more than two classes. In the case of classes *K*, where *K* > 2, One-Vs-Rest method fits *K* number of SVMs. Means, it trains one classifier per class, and each time comparing one of these *K* classes to the remaining *K* − 1 classes [31]. Classification is performed by picking the class *j* that maximises the below function, before applying the sign function [37],

$$\arg \ max \ g^j(x)_{j=1,\dots,K} \ \ where \ g^j(x) = \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \tag{13}$$

If the parameters for a SVM classifier fitted for the *k*th class is given by $w_{0_k}, w_{1_k}, \dots, w_{p_k}$, and x^* is a test observation which is represented by its features $x_1^*, x_2^*, \dots, x_p^*$, the class for the observation is assigned for which $w_{0_k} + w_{1_k} x_1^* + \dots + w_{p_k} x_p^*$ is the largest. This indicates the high level of confidence that the observation belongs to the *k*th class [31]. This approach can also be used for multi-class classification,

5.2.2 One-Vs-One (OVO)

The One-Vs-One approach also known as all-pairs approach, constructs $\binom{K}{2}$ binary SVMs. Suppose we have a test observation, it is classified by each of $\binom{K}{2}$ classifiers, and the prediction for all classifier is tallied up for each class. The class which is chosen most frequently by the classifiers is the final classification.

5.3 Random Forests

Random Forests is an ensemble method that can be used for Regression as well as Classification. It is an improvement over the *bagging* technique, by building a large collection of *de-correlated* trees [31]. *Bagging* builds a number of decision trees on bootstrapped training models, to average noisy models and in turn reduce variance.

In Random Forests, when building these decision trees, at each split in each tree, only a random sample of *m* features are considered as split candidates from the full set of *p* features. The split is allowed to use only one of these *m* features [31]. The *m* variables are selected at random at each split.

For Classification, the value of *m* is typically $\sqrt{p}$, and for Regression it is *p*/3.

Random Forest for Classification [36]:

1. For $b = 1$ to B:
   - Create a bootstrapped sample from the training data.
   - Grow a tree T_b on the bootstrapped data, by following the below steps for each node,
     - Pick *m* features from the total *p* features
     - Identify the best split, i.e., the feature from *m* and a threshold for splitting the data, using the Gini Impurity criterion.
     - Split the node based on the best split
2. Out the ensemble $\{T_b\}^B$

To make prediction for a test observation *x*, the classification can be made by,

$$\hat{C}_{rf}^{B}(x) = majority\ vote\ \left\{\hat{C}_b(x)\right\}_1^B \tag{14}$$

Where $\hat{C}_b(x)$ is the class prediction of the *b*th random-forest tree.

5.4 Performance Metrics

5.4.1 Confusion Matrix

Confusion Matrix or a Contingency Table is used to describe the performance of a classification algorithm. We can extract the metrics *precision*, *recall*, and *f1-score* for each class from the confusion matrix [38].

The rows in a confusion matrix represent the actual classes and the columns represent the predicted classes. So, for a classifier with *c* labels, we get a $c \times c$ confusion matrix. The diagonal elements in the matrix from top left to bottom right represent the count where the actual labels match the predicted labels, i.e., where the labels were correctly predicted also called as True Positive (TP). For each class, the confusion matrix also shows the True Negatives (TN), which is the count of instances which were correctly identified as not belonging to that class; False Positives (FP) which is the count of instances which were incorrectly identified as

belonging to a class; and False Negatives (FN) which is the count of instances belonging to a class were predicted as belonging to another class.

A confusion matrix is often visualised in a. heatmap style using colours. Darker shades of colours usually represent higher values, and lighter shades represent lower values.

5.4.2 Accuracy Score, Precision, Recall and F1-Score

Accuracy represents the correctly classified data points by a classifiers prediction. It is a common metric used to evaluate the performance of a model [39]. It is given as,

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{15}$$

Precision is used to measure the positive predictions are correct from the total predicted instances in positive class,

$$precision = \frac{TP}{TP + FP} \tag{16}$$

Recall is used to measure the positive patterns that are correctly classified from positive predictions,

$$recall = \frac{TP}{TP + TN} \tag{17}$$

Individually, precision and recall are incomplete views of a classifiers information and are seldom informative [40]. *F1 measure* or *f1-score* is the harmonic mean of *precision* and *recall*,

$$f1\ measure = \frac{2 \times precision \times recall}{precision + recall} \tag{18}$$

For the overall analysis of a classification model, *micro*, *macro*, and *weighted* average of *f1-score* can be used.

6 Data Preprocessing

6.1 Bag-of-Words (BoW)

In text classification, the bag of words model is used to record the occurrences of each word, disregarding the grammar and order of the words. Based on text documents or sentences, each sentence is tokenized to produce a dictionary or a group of words. The number of occurrences of a word in a text is called as term frequency.

This approach is established upon the assumption, that the frequencies of words in a document or text can capture the differences or similarities between the documents or texts. These words are converted into feature vectors which capture the count of that word in that text. However, as the number of words increases, the number of features increases significantly, and the dimension can become extremely large [41]. Common stop words like 'the', 'is', 'a' etc are typically ignored while converting the words in the text into count feature vectors since they can be non-informative.

In this project, the 'definition' attribute for each GO term has been converted into feature vectors using bag-of-words method.

6.2 Label Encoding using one-hot encoding

In the case of multi-label classification, where we aim to classify an instance to one or more labels or categories from a set of possible labels. Using the 'MultiLabelBinarizer' from scikit-learn, the lists of multi-class labels 'is a' is transformed into binary format.

This means, each label is converted into a column and the presence of the label for a record is indicated by 1 and absence is indicated by 0. Suppose we have a given set of all possible labels $\{l_1, l_2, l_3, l_4\}$. For a sample that has labels $\{l_1, l_4\}$, the label encoding would be [1,0,0,1].

6.3 Normalisation

Normalisation is a technique used to scale the numerical features to a standard range. This was done only for experiments to observe the performance of the models with a normalised dataset.

The features were normalised by removing mean and scaling to unit variance. The standard score for a feature x was calculated as below,

$$z = \frac{(x - \mu)}{\sigma} \tag{19}$$

Where:

- x is the original value
- μ is the mean of the data
- σ is the standard deviation
- z is the normalised value

6.4 Feature Reduction

Performing bag-of-words on the 'definition' attribute during preprocessing resulted in a very high-dimensional space, consist of approximately 38,093 features. Such a large number of dimensions can lead to challenges, like overfitting and the curse of dimensionality, which may severely impact the performance of the models.

To address these issues, the experiments were conducted on two feature reduction scenarios for each model:

- Excluding the words which appear in less than **1%** of overall definitions
- Excluding the words which appear in less than **5%** of overall definitions

6.5 Down-sampling

This technique was used only for the multi-class classification experiments for the optimization process, for the experiments involving Support Vector Machines and Random Forests.

Performing an exhaustive grid search for various possible parameter values takes a lot of time and is computationally intensive. By using down sampled dataset for optimization process we sought to reduce time and optimize computational efficiency as well. Equal number of records for each label were resampled without replacement. Another idea behind the down-sampling was to ensure that each class was equally represented to mitigate any potential bias due to class imbalance.

Down sampling was done by first dividing the dataset into three subsets corresponding to the three namespace categories 'Biological Process', 'Molecular Function' and 'Cellular Component'. Each subset was then down sampled to a predetermined sample size of 2500. This was performed without replacement to avoid duplicating instances. After down sampling, these subsets were combined into a single dataset.

7 Experiments

7.1 Multi-class classification

The multi-class classification experiments were conducted using four distinct scenarios across each model to assess the impact of dataset normalisation and feature reduction on model performance. The experiments were structured as follows,

Experiment Scenarios:

1. **Non-normalised Dataset:**
   - **Scenario 1**: Non-normalised dataset with words which appear in less than **1%** of overall definitions being excluded.
   - **Scenario 2**: Non-normalised dataset with words which appear in less than **5%** of overall definitions being excluded.
2. **Normalised Dataset:**
   - **Scenario 3**: Normalised dataset with words which appear in less than **1%** of overall definitions being excluded.
   - **Scenario 4**: Normalised dataset with words which appear in less than **5%** of overall definitions being excluded.

In these experiments, we try to classify the GO terms into 3 different namespace categories, 'Biological Process', 'Molecular Function' and 'Cellular Component' based on their definitions. In the GO ontology dataset, there is a significant class imbalance in the data set, which can be observed in Figure 2.

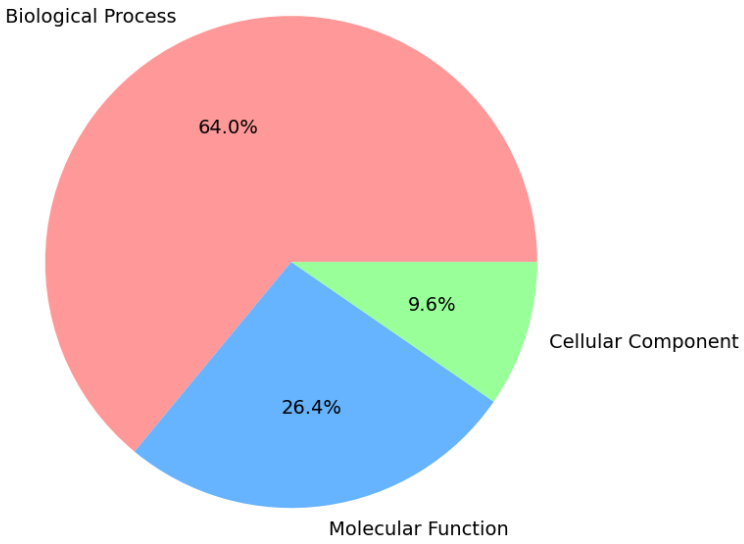

Figure 2. GO Terms Distribution by Namespace

7.1.1 K-Nearest Neighbours

The K-Nearest Neighbours (KNN) algorithm to classify the namespace categories from the Gene Ontology dataset based on the definitions, using the datasets processed through feature reduction. The analysis covered four experimental setups, each varying by the degree of feature exclusion and normalisation, to observe and understand the algorithms performance across different data conditions.

The first phase of experiments utilized non-normalised datasets. In the first experiment, the words that appeared in less than 1% of definitions were excluded (Scenario 1). This set up aimed to observe the behaviour of KNN without the influence of feature scaling. The model was trained on 80% of the dataset, with remaining 20% held for testing. The models performance was evaluated for different values of *k*, ranging from 1 to 20. The aim of this evaluation was to find an optimal value of *k* that yields the highest accuracy. The results showed that the accuracy was highest at $k = 1$ with a score of 0.9535, and it decreased as *k* increased. The second experiment used a dataset which excluded the words found in less than 5% of definitions (Scenario 2). Unlike the first, this dataset showed an increase in accuracy as *k* increased, peaking at $k = 6$ with an accuracy of 0.9261.

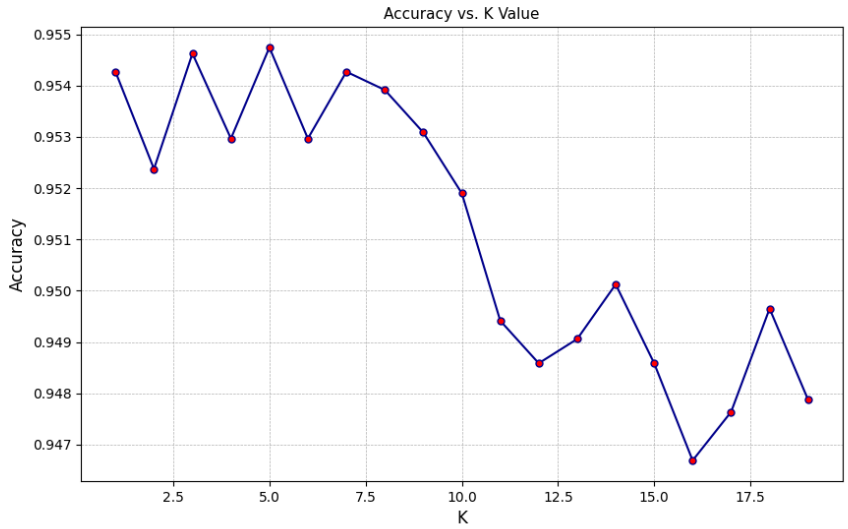

Figure 3. Evaluation of accuracy for different values of *k* for Scenario 1

The second phase of experiments focused on normalised datasets, where the feature vectors were scaled to a uniform range to potentially enhance the model's performance. The third experiment involved data that excluded words appearing in less than 1% of definitions, while experiment four involved data that excluded words appearing in less than 5% of the definitions. Both datasets followed the same training and test split. In the third experiment, the optimal accuracy was achieved at $k = 1$ with a score of 0.9436. For the fourth experiment, the highest accuracy was 0.9245 at $k = 16$.

7.1.2 Support Vector Machines

This section discusses the application of Support Vector Machines (SVM) algorithm for classification of Gene Ontology namespace based on GO definitions. The experiment investigated the performance of SVM across both non-normalised and normalised datasets incorporating different feature exclusion criteria, and also involved optimising SVM parameter settings through grid search and cross-validation.

For this study, down-sampled datasets were created, containing 2,500 records for each of the three categories (Biological Process, Molecular Function, Cellular Component). These down-sampled datasets were used to run a comprehensive parameter tuning using GridSearchCV[42] to enhance SVM's performance. Tuning hyperparameters over a large dataset can be computationally intensive and takes a lot of time which is why we used a down-sampled dataset for the same. The down sampling involved randomly selecting a subset of the data, ensuring all classes were equally represented to maintain balance necessary for training and evaluating an unbiased model.

The GridSearchCV explores a range of values for the parameters like *C* which is a regularization parameter, *kernel* and *gamma*. The grid search was performed using a 5-fold cross-validation to ensure the effectiveness of the model across different subsets of the data. After the optimization process, the SVM was assessed on the full datasets for both non-normalised and normalised datasets, which was split into training and test subsets, to validate the selected parameters.

The Table. 1 summarises the optimized parameters for each experiment, highlighting the differences in the settings for based on the distinct scenarios.

Experiment	1	2	3	4
Data Description	Non-normalised, <1% exclusion	Non-normalised, <5% exclusion	Normalised, <1% exclusion	Normalised, <5% exclusion
C	10	10	1	1
Kernel	RBF	RBF	RBF	Poly
Gamma	0.1	0.1	0.01	0.1
Accuracy	0.9676	0.9246	0.9572	0.9238

Table 1. Optimized parameters for multi-class SVM

The non-normalised dataset, which excluded words which appeared in less than 1% of definitions, achieved the highest test accuracy of 0.9676.

The confusion matrix depicted in Figure 5. illustrates the classification performance on the test set and provides a detailed breakdown of the models' predictive accuracy across the namespace categories.

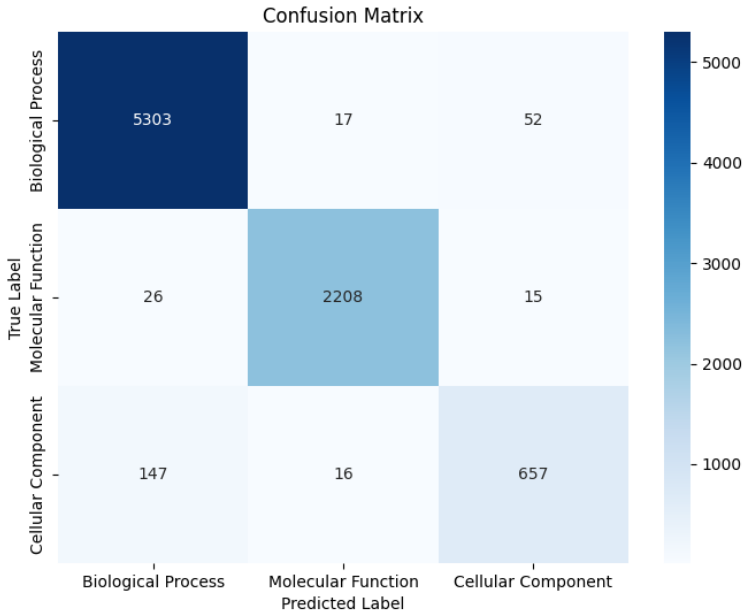

Figure 5. Test set prediction confusion matrix for multi-class SVM

7.1.3 Random Forests

This section investigates the application of Random Forest classifiers to classify the GO namespace. Similar to the previous models, the experiments are conducted on the normalised and non-normalised datasets, with words excluded based on their appearance in less than 1% and 5% of definitions, respectively.

The parameter optimization for Random Forest is performed using GridSearchCV as well. The datasets were down-sampled and used for hyperparameter tuning since it involves training numerous models to evaluate the performance across a range of parameter values, which can be computationally intensive and time-consuming. The hyperparameters to be tuned were *n_estimators*, *max_depth*, *min_sample_split*, *min_samples_leaf* and *bootstrap*.

Following the optimisation of the hyperparameters, the models were then trained on full dataset using the 80-20 split for the training and testing purposes. The optimised parameters were used for all four scenarios. The optimized parameter values which resulted in maximum accuracy of 0.9705 were *n_estimators* = 30, *min_samples_split* = 5, *min_samples_leaf* = 1, *bootstrap* = True and *max_depth*= None.

The normalised dataset also achieved an accuracy close to this at 0.9681 with its optimized parameter values. Although, the dataset which excludes the words appearing in less than 5% of definitions for both normalised and non-normalised datasets achieve the same lower accuracy even after optimization, which is between 0.9246. The Table 2. Summarises the optimised parameters for all the scenarios/experiments and their corresponding accuracies.

Experiment	1	2	3	4
Data Description	Non-normalised, <1% exclusion	Non-normalised, <5% exclusion	Normalised, <1% exclusion	Normalised, <5% exclusion
n_estimators	30	30	60	30
max_depth	None	50	None	None
min_sample_split	5	10	5	10
min_samples_leaf	1	4	2	4
bootstrap	True	False	True	False
Accuracy	0.9702	0.9246	0.9681	0.9246

Table 2. Optimized parameters for multi-class Random Forest

7.2 Multi-label classification

The multi label classification was done to predict the 'is a' relationship based on the definitions. The 'is a' attribute has more than 15,000 unique values and presents a considerable challenge due to its extensive variability. In this project, we have focused on the top 10 and top 20 frequent values, treating them as distinct labels. A GO term can have multiple 'is a' values, hence this falls under the multi-label classification problem.

Top 20 labels:

GO:0110165	GO:0016616	GO:0032991	GO:0016709	GO:0016758
GO:0048856	GO:0098797	GO:0140513	GO:0016747	GO:0003006
GO:1901700	GO:0016836	GO:0008757	GO:0016811	GO:0051241
GO:0016773	GO:0051240	GO:0008168	GO:0014070	GO:1901701

The experiments were exclusively conducted on a non-normalised dataset. The experiments conducted will be using SVM and Random Forest and each model was tested with top 10 labels and top 20 labels. Feature exclusion criteria included excluding the words that appear in less than 1% of the definitions.

7.2.1 Support Vector Machines

Support Vector Machine (SVM) was employed within a One-Vs-Rest approach to solve each label as a separate binary classification problem, and to predict the 'is a' relationship in the multi-label classification by using the 'definitions' attribute that was transformed into feature vectors.

For the first experiment, the top 10 frequent values for 'is a' were used as labels. A grid search for various values for the hyperparameters was done to perform an exhaustive search over specified parameter values. The parameters are optimised by 5-fold cross-validation grid-search over a parameter grid. The

GridSearchCV was used for the same and was optimised based on the f1-score. These optimised parameters were then used to train the model on the full dataset which was split into 80% for training and 20% for testing purposes. The result with the optimised parameters was micro average f1-score of 0.8512.

The second experiment utilised top 20 frequent values for 'is a' attribute as labels. This experiment achieved a micro average f1-score of 0.8118 with the optimised parameters.

Experiment	1	2
Data Description	With top 10 labels	With top 20 labels
C	50	5
Kernel	RBF	RBF
Gamma	0.01	0.1
Accuracy	0.8512	0.8118

Table 3. Optimized parameters for multi-class SVM

7.2.2 Random Forests

The Random Forest classifier was also used to address the multi-label classification task for 'is a' labels. Hyperparameter tuning was performed to maximize the f1-score similar to the previous experiments.

For the first experiment with 10 labels, the model trained on optimised parameter values resulted in the f1-score of 0.8676. Whereas, for the second experiment, the model trained on optimised parameter values resulted in the f1-score of 0.8002.

Experiment	1	2
Data Description	With 10 labels	With 20 labels
n_estimators	170	270
max_depth	42	33
min_sample_split	2	2
min_samples_leaf	1	1
bootstrap	True	True
Accuracy	0.8676	0.8002

Table 2. Optimized parameters for multi-class Random Forest

8 MultiGOClassifier Implementation

The MultiGOClassifier tool is a Gene Ontology Classification tool that we have implemented has two main components: one classifier to predict the 'namespace' and another to predict the 'is a' relationship based on the input text provided. The classifier takes a number of inputs before predicting the result. The inputs required vary depending on the type of classifier that is being used. The step-by-step process the tool undergoes can be seen in the Figure 6.

Before making predictions, the classifier requires several inputs:

- **Type of classifier**: for which the options are 'Namespace' classifier and 'is a' classifier.
- **Type of model to use:**
  Options for 'Namespace' classifier are:
  - K Nearest neighbours (KNN)
  - Support Vector Machines (SVM)
  - Random Forest (RF)

  Options for 'is a' classifier are:
  - Support Vector Machines (SVM)
  - Random Forest (RF)
- **The number of labels to use:** This option is only for the multi-class 'is a' classifier. The top *n* frequent values for 'is a' are used as labels. The user has the option to change the number of labels (*n*). The default value for this parameter is 10.
- **Use custom dataset:** The Gene Ontology dataset keeps updating regularly. The user has an option to use a different (updated) dataset.
- **Change words exclusion %:** The tool by default excludes the words that appear in less than 1% of definitions. This threshold can be adjusted by the user.
- **Choose to optimise the model parameters:** The user has an option to run optimisation for the model parameters based on the other selected inputs.
- **Input text:** The input for this tool is a text description that the user wants to predict the Namespace or 'is a' relationship for.

The tool uses the optimised parameter values from the previous experiments as the default model parameter values.

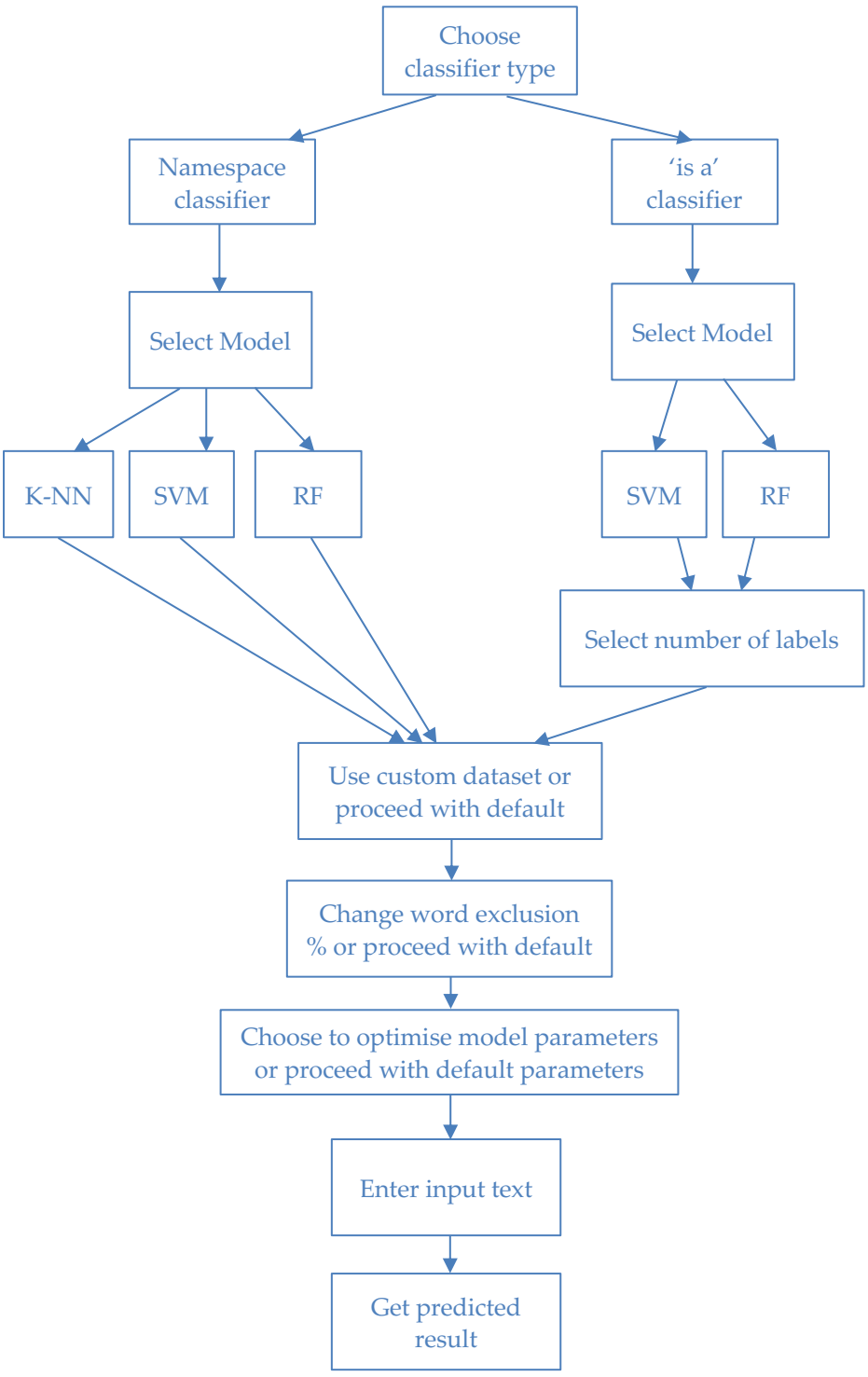

Figure 6. MultiGOClassifier workflow

9 Code Structure and Usage

The code for this project was written in *Python*. The experiments were conducted in *jupyter notebooks* and the MultiGOClassifier tool was implemented using *.py* files.

The structure of the code is shown below,

```
.
├── Readme.md
├── dataset
│   └── go-basic.obo
├── experiments
│   ├── multigoclassfier_multiclass.ipynb
│   ├── multigoclassfier_multilabel_10.ipynb
│   └── multigoclassfier_multilabel_20.ipynb
├── multigoclassfier_tool
│   ├── is_a_classifier
│   │   ├── __init__.py
│   │   ├── is_a_classifier.py
│   │   └── models
│   │       ├── __init__.py
│   │       ├── random_forest.py
│   │       └── svm.py
│   └── namespace_classifier
│       ├── __init__.py
│       ├── models
│       │   ├── __init__.py
│       │   ├── namespace_knn.py
│       │   ├── namespace_random_forest.py
│       │   └── namespace_svm.py
│       └── namespace_classifier.py
├── images
│   ├── KNN1.png
│   ├── SVM1.png
│   ├── example.png
│   └── piechart.png
├── requirements.txt
└── run.py
```

Figure 7. Hierarchy of the code

The experiment notebooks are located in *experiments* folder.

The *multigoclassifier_tool* consists of the MultiGOClassifier implementation files. Which include the functions to take required inputs, parse the dataset, clean

and transform the dataset for training, optimisation functions and training the models. The *run.py* file in the root folder is the file used to run the classifier and generate predications. All the dependencies are listed in the *requirements.txt* file.

The instructions to set up the project are provided in the *readme.md* file of the project.

Once the dependencies have been installed, the tool can be used by running the *run.py* using the below command,

python3 run.py

The tool will prompt the user for the inputs and then provide predications based on the same. Below is an example for running the tool:

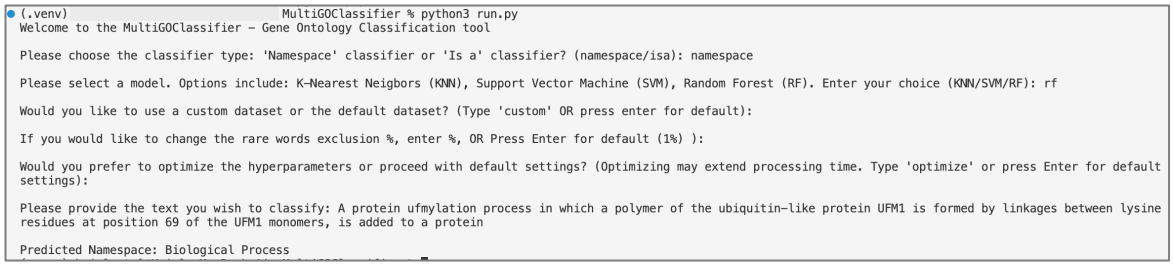

Figure 8. Example of running the MultiGOClassifier Namespace classifier

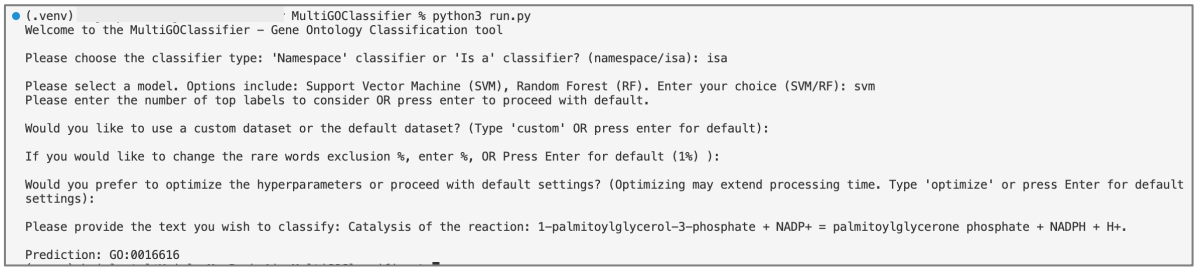

Figure 9. Example of running the MultiGOClassifier 'is a' classifier

10 Future Directions, Conclusion

The exploration of machine learning models for predicting Gene Ontology information points to several directions for future research. One of them is the integration advanced machine learning algorithm like deep learning, which may enhance the accuracy and scalability of the predictions. Since user can use custom dataset, this makes it important that the format and structure of the custom dataset are compatible with those of the Gene Ontology dataset. Further studies may focus on integrating gene annotation dataset as well to predict different information.

The optimisation process runs a grid search for the model parameter evaluating different values. Currently the tool uses the values that were used in the experiment to run the grid search. But if the dataset changes, there may be a possibility that we may need to explore some other values which may not be a part of the optimisation function code right now. This is a potential issue that can be considered in further studies.

An additional interesting direction would be to collaborate with genomic platforms to facilitate real-time data integration, allowing the information to be predicted and models to be optimised continuously as new genomic data becomes available instead of making the user select a custom (updated) dataset. As we refine these models and expand their capabilities, machine learning will lead to more innovative tools useful for the biomedical researchers.

This project demonstrated the potential of machine learning models to predict gene ontology information. We explored K-Nearest Neighbours, Support Vector Machines and Random Forests in multi-label and multi-class classification scenarios. We optimised the hyperparameters for them and observed performance of these models under different scenarios and criteria. We were able to achieve highest accuracy with Random Forest amongst all. We developed a categorizer tool based on these experiments.

References

[1] Gene Ontology Consortium, "The Gene Ontology Resource." [Online]. Available: https://geneontology.org/

[2] C. Dessimoz and N. Škunca, Eds., *The Gene Ontology Handbook*, vol. 1446. in Methods in Molecular Biology, vol. 1446. New York, NY: Springer New York, 2017. doi: 10.1007/978-1-4939-3743-1.

[3] J. Gobeill, E. Pasche, D. Vishnyakova, and P. Ruch, "Closing the loop: from paper to protein annotation using supervised Gene Ontology classification," *Database (Oxford)*, vol. 2014, 2014, doi: 10.1093/database/bau088.

[4] M. Ashburner *et al.*, "Gene ontology: Tool for the unification of biology," May 2000. doi: 10.1038/75556.

[5] M. Ashburner *et al.*, "Creating the Gene Ontology resource: Design and implementation," *Genome Res*, vol. 11, no. 8, pp. 1425–1433, 2001, doi: 10.1101/gr.180801.

[6] J. Z. Wang, Z. Du, R. Payattakool, P. S. Yu, and C. F. Chen, "A new method to measure the semantic similarity of GO terms," *Bioinformatics*, vol. 23, no. 10, pp. 1274–1281, May 2007, doi: 10.1093/bioinformatics/btm087.

[7] B. Smith, J. Williams, and S. Schulze -Kremer, "The Ontology of the Gene Ontology."

[8] S. A. Aleksander *et al.*, "The Gene Ontology knowledgebase in 2023," *Genetics*, vol. 224, no. 1, May 2023, doi: 10.1093/genetics/iyad031.

[9] Gene Ontology Consortium, "The Gene Ontology Resource." [Online]. Available: http://www.geneontology.org

[10] M. A. Harris *et al.*, "The Gene Oncology (GO) database and informatics resource," *Nucleic Acids Res*, vol. 32, no. DATABASE ISS., Jan. 2004, doi: 10.1093/nar/gkh036.

[11] T. Z. Berardini *et al.*, "The Gene Ontology in 2010: Extensions and refinements," *Nucleic Acids Res*, vol. 38, no. SUPPL.1, Jan. 2010, doi: 10.1093/nar/gkp1018.

[12] S. Carbon *et al.*, "Expansion of the gene ontology knowledgebase and resources: The gene ontology consortium," *Nucleic Acids Res*, vol. 45, no. D1, pp. D331–D338, Jan. 2017, doi: 10.1093/nar/gkw1108.

[13] J. A. Blake *et al.*, "Gene ontology annotations and resources," *Nucleic Acids Res*, vol. 41, no. D1, Jan. 2013, doi: 10.1093/nar/gks1050.

[14] J. A. Blake *et al.*, "Gene ontology consortium: Going forward," *Nucleic Acids Res*, vol. 43, no. D1, pp. D1049–D1056, Jan. 2015, doi: 10.1093/nar/gku1179.

[15] M. A. Harris *et al.*, "The gene ontology (GO) project in 2006," *Nucleic Acids Res*, vol. 34, pp. D322–D326, 2006, doi: 10.1093/nar/gkj021.

[16] E. B. Camon *et al.*, "An evaluation of GO annotation retrieval for BioCreAtIvE and GOA," *BMC Bioinformatics*, vol. 6, no. S1, p. S17, May 2005, doi: 10.1186/1471-2105-6-S1-S17.

[17] D. Barrell, E. Dimmer, R. P. Huntley, D. Binns, C. O'Donovan, and R. Apweiler, "The GOA database in 2009 - An integrated Gene Ontology Annotation resource," *Nucleic Acids Res*, vol. 37, no. SUPPL. 1, 2009, doi: 10.1093/nar/gkn803.

[18] P. D. Thomas, H. Mi, and S. Lewis, "Ontology annotation: mapping genomic regions to biological function," Feb. 2007. doi: 10.1016/j.cbpa.2006.11.039.

[19] P. Khatri, S. Draghici, G. C. Ostermeier, and S. A. Krawetz, "Profiling gene expression using Onto-Express," *Genomics*, vol. 79, no. 2, pp. 266–270, Feb. 2002, doi: 10.1006/geno.2002.6698.

[20] O. D. King, R. E. Foulger, S. S. Dwight, J. V. White, and F. P. Roth, "Predicting gene function from patterns of annotation," *Genome Res*, vol. 13, no. 5, pp. 896–904, May 2003, doi: 10.1101/gr.440803.

[21] A. Schlicker, T. Lengauer, and M. Albrecht, "Improving disease gene prioritization using the semantic similarity of Gene Ontology terms," in *Bioinformatics*, Oxford University Press, 2011, pp. i561–i567. doi: 10.1093/bioinformatics/btq384.

[22] C. R. Primmer, S. Papakostas, E. H. Leder, M. J. Davis, and M. A. Ragan, "Annotated genes and nonannotated genomes: Cross-species use of Gene Ontology in ecology and evolution research," Jun. 2013. doi: 10.1111/mec.12309.

[23] H. M. Müller, E. E. Kenny, and P. W. Sternberg, "Textpresso: An ontology-based information retrieval and extraction system for biological literature," *PLoS Biol*, vol. 2, no. 11, Nov. 2004, doi: 10.1371/journal.pbio.0020309.

[24] Gene Ontology Consortium, "Download the ontology." [Online]. Available: https://geneontology.org/docs/download-ontology/

[25] "https://owlcollab.github.io/oboformat/doc/obo-syntax.html."

[26] Mungall Chris and Ireland Amelia, "The OBO Flat File Format Guide, version 1.4." [Online]. Available: https://owlcollab.github.io/oboformat/doc/GO.format.obo-1_4.html#top

[27] J. Gobeill, E. Pasche, D. Vishnyakova, and P. Ruch, "Managing the data deluge: Data-driven GO category assignment improves while complexity of functional annotation increases," *Database*, vol. 2013, 2013, doi: 10.1093/database/bat041.

[28] J. Gobeill *et al.*, "Deep question answering for protein annotation," *Database*, vol. 2015, 2015, doi: 10.1093/database/bav081.

[29] Y. Mao *et al.*, "Overview of the gene ontology task at BioCreative IV," 2014, *Oxford University Press*. doi: 10.1093/database/bau086.

[30] G. Julien, P. Emilie, V. Dina, and R. Patrick, "BiTeM/SIBtex group proceedings for BioCreative IV, Track 4: Gene Ontology curation." [Online]. Available: http://www.ncbi.nlm.nih.gov/gene/about-generif

[31] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, "An Introduction to Statistical Learning with Applications in Python."

[32] T. M. . Mitchell, *Machine learning*. McGraw-Hill, 1997.

[33] V. B. S. Prasath *et al.*, "Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier -- A Review," Aug. 2017, doi: 10.1089/big.2018.0175.

[34] D. Barber, "Bayesian Reasoning and Machine Learning," 2007.

[35] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," 1998.

[36] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning."

[37] Bernhard. Scho ̈lkopf and A. J. . Smola, *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[38] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022, doi: 10.1109/ACCESS.2022.3151048.

[39] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[40] Gavin Hackeling, *Mastering Machine Learning with scikit-learn*.

[41] Qader Wisam A., Ameen Musa M., and Ahmed Bilal I., *An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges*. IEEE, 2019.

[42] scikit-learn, "GridSearchCV." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html