# INDCAL
# INDICES CALCULATOR

## 1. ABSTRACT:

This project focuses on identifying the physicochemical and biological properties of chemical compounds with the help of molecular descriptors. The challenge is to predict the properties of various unknown chemical compounds in the development of quantitative structure-activity relationships in which the biochemical activity or other properties of molecules can be correlated with their chemical structure. This is being applied in many disciplines such as risk assessment, toxicity prediction, and regulatory decisions in addition to drug discovery and lead optimization.

## KEYWORDS:

Cheminformatics, Properties, Chemical Compounds, Molecular Descriptors, Indices, Machine Learning

## 2. INTRODUCTION:

This project belongs to the domain 'Cheminformatics'. Cheminformatics is a field that applies computational techniques and tools to the analysis, management, and manipulationof chemical data. It is an interdisciplinary domain that combines chemistry, computer science, mathematics, and statistics to solve problems related to chemical information.

Cheminformatics involves the use of various software tools, databases, and algorithms to handle and analyze large amounts of chemical data. This includes molecular modeling and simulation, chemical structure searching and retrieval, drug design and discovery, prediction of chemical properties and toxicity, and analysis of chemical reaction networks.

One of the major applications is being implemented in the project:

Till date, for identification and study the properties of a chemical compound, microscopicresearch has to be done. Scientists and researchers should have a basic knowledge about the properties and uses of the elements present in it. They should also have a prior knowledge on how the reactions take place when the

elements are put together. It involvesa lot of study and requires time to complete the task. It also requires a combined work of many people. Even if done in a systematic manner, the accuracy of the results will not be as expected. If there's a mistake, it's tedious to identify why, what, where and how it happened. It's also difficult to opt for an alternative and rectify it within a specific durationof time. Also, man-made research tends to mis happenings. It's dangerous for the humans and the researchers to deal with hazardous chemical reactions in order to find their properties and usages.

Nowadays, advances in computer processing power and algorithms have made it possibleto simulate and predict the behavior of molecules and chemical reactions. This allows researchers to explore new ideas and test hypotheses before performing expensive and harmful experiments. When machines are trained to find the properties of the compounds, the task is completed within the given duration and with a greater accuracy of results as expected. When the molecular descriptors of a particular chemical compound are identified with respect to their structure, it's easy for the scientists and the researchers to predict the physical, chemical and biological properties of the compound along with its uses with high accuracy. It also reduces the time spent and workload. This has many important applications in various fields of chemistry, including drug discovery, materials science, environmental science, and industrial chemistry. Some specific applications of predictingproperties of chemical compounds include:

**Drug discovery:** Prediction of the properties of potential drug candidates, such as their solubility, bioavailability, and toxicity, can help in the design and optimization of new drugs. This can lead to more effective and safer drugs being developed in a shorter time.

**Materials science:** Prediction of properties such as mechanical strength, thermal conductivity, and electrical conductivity can help in the design and development of new materials with specific properties for various applications, such as construction, electronics, and aerospace.

**Environmental science:** Prediction of properties such as biodegradability, persistence, and toxicity can help in the assessment and management of chemical pollutants in the environment, including soil, water, and air.

**Industrial chemistry:** Prediction of properties such as boiling point, viscosity, and reactivity can help in the design and optimization of industrial processes for the production of chemicals, pharmaceuticals, and materials.

**Toxicology:** Prediction of properties such as carcinogenicity, mutagenicity, and reproductive toxicity can help in the assessment and management of the safety of chemicals, drugs, and other substances.

Overall, the prediction of properties of chemical compounds plays a vital role in the development of new materials and drugs, the management of environmental pollutants, and the safety assessment of chemicals and drugs.

In this work, the SMILES notation is given as input to the indices calculator. SMILES in chemistry is abbreviated as SIMPLIFIED MOLECULAR INPUT LINE ENTRY SYSTEM. It is used to translate a chemical's three-dimensional structure into a string of symbols that is easily understood by computer software. Once it's given as input, molecular descriptors such as topological indices and topochemical indices of the chemical compounds are calculated. These indices include generalized randic index, augmented zagreb index, geometric arithmetic index, harmonic index, product connectivity index, general sum connectivity index, atom-bond connectivity index, molecular connectivity index, etc. Molecular descriptors are the mathematical representations of molecular properties that are generated by algorithms. The numerical values of molecular descriptorsare used to quantitatively describe the physical and chemical information of the molecules. Once the molecular descriptors are found, the properties of given chemical compounds can be identified. When a machine learning model is trained with these datasets which include the descriptors and its related properties of various known compounds, the properties of new and unknown chemical compounds can be predicted.

## 2.1 PURPOSE:

The purpose is to describe all the requirements for the implementation of the indices calculator. The following are some of the stakeholders:

1. User
2. Developer

## 2.2 SCOPE:

- The scope of the project is prediction of physicochemical and biological properties of an unknown chemical compound provided its chemical structure isgiven as input in the form of SMILES.
- Additionally, the purpose is to find the applications of the chemical compound with the help of molecular descriptors.
- The input will be given via a database and then it is further validated for processing.
- Helps to improve the accuracy of results when compared to man made researchwithin the given duration of time.

The steps implemented in the indices calculator are given below:

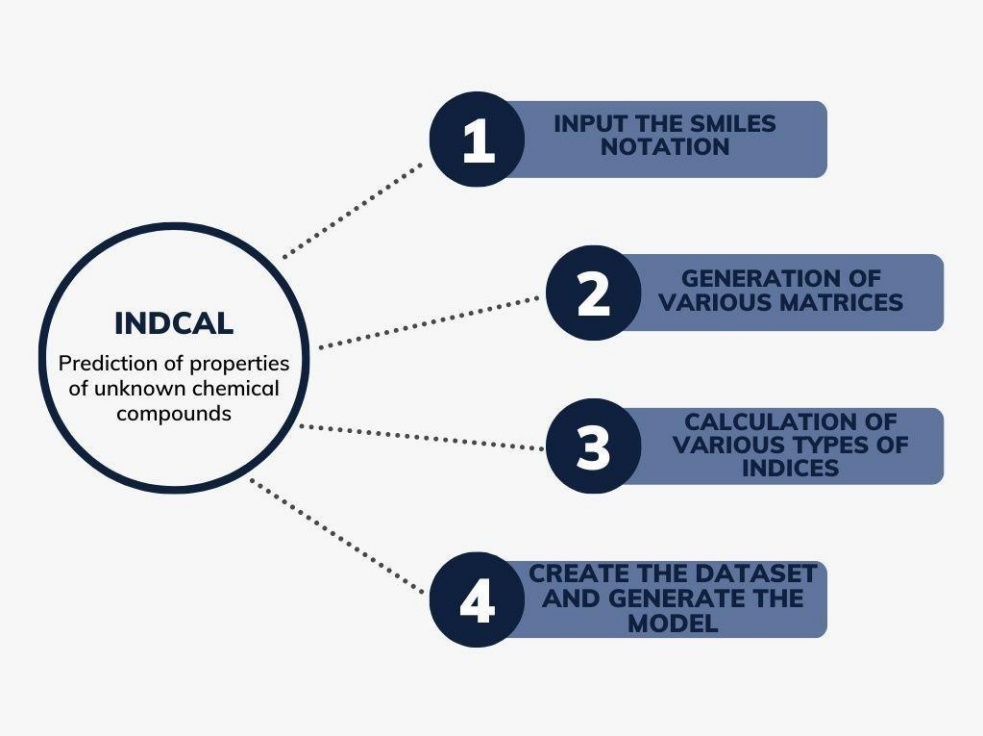



**Fig. 2.1** Steps performed in INDCAL

In Fig. 2.1, to predict the properties of unknown chemical compounds, the work has beendivided into several categories.

## 3. PROPOSED SYSTEM:

Our approach to solve the problem of identifying the physicochemical and biologicalproperties of unknown chemical compound is broadly divided into four blocks:

1. Validate the given input
2. Generation of adjacency matrix
3. Generation of distance matrix
4. Calculation of molecular descriptors
5. Generation of train and test data set
6. Implementation of prediction model

**A.** The first process is to validate the notation and identify whether it's correct or not. A SMILES notation is considered correct when

- An atom is represented using its respective atomic symbol. Upper case letters refer to non-aromatic atoms; lower case letters refer to aromatic atoms. If the atomic symbol has more than one letter the second letter must be lower case.

- Single bonds are the default and therefore need not be entered. Double bonds arerepresented by '=' and Triple bonds by '#'.

- The structures that are entered using SMILES are hydrogen suppressed. The SMILES software understands the number of possible connections that an atom can have. If enough bonds aren't identified by the user through SMILES notation, the system will automatically assume that the other connections are satisfied by hydrogen bonds.

- A branch from a chain is specified by placing the SMILES symbol(s) for the branch between parenthesis. The string in parentheses is placed directly after thesymbol for the atom to which it is connected. If it is connected by a double or triple bond, the bond symbol immediately follows the left parenthesis.

- SMILES allows a user to identify ring structures by using numbers to identify the opening and closing ring atom. Chemicals that have multiple rings may be identified by using different numbers for each ring. If a double, triple, or aromatic bond is used for the ring closure, the bond symbol is placed before thering closure number.

If these basic rules are satisfied by the input provided, then it moves for the next step.Otherwise, the software pops out an error.

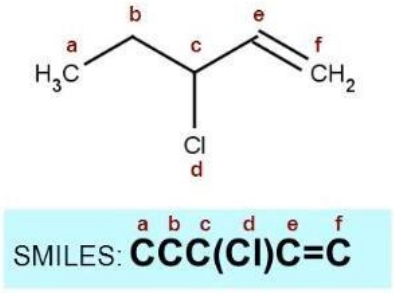



**Fig. 3.1** SMILES of an aliphatic compound

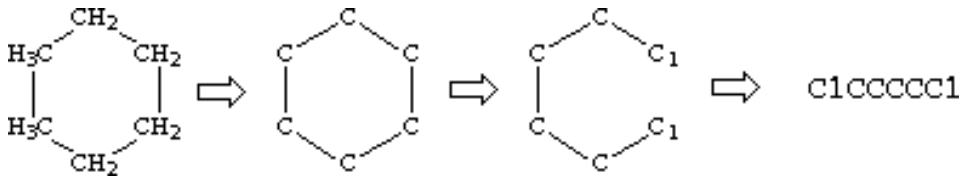



**Fig. 3.2** SMILES of an aromatic compound

| No. | Compound | SMILES |
|---|---|---|
| 1 | Ethanol | OCC |
| 2 | Pentanal | OCCCCC |
| 3 | Decane | C(CCCCCCCC)C |
| 4 | Hexanal | O=CCCCCC |
| 5 | 2-Butylfuran | o1c(ccc1)CCCC |
| 6 | 2-Heptanone | O=C(C)CCCCC |
| 7 | Heptanal | O=CCCCCCC |
| 8 | (*E*)-2-Hexenal | O=C/C=C/CCC |
| 9 | 2-Pentylfuran | o1c(ccc1)CCCCC |
| 10 | 1-Pentanol | OCCCCC |
| 11 | 2-Octanone | O=C(C)CCCCCC |
| 12 | Octanal | O=CCCCCCCC |
| 13 | Tridecane | C(CCCCCCCC)CCCC |
| 14 | (*E*)-2-Heptenal | O=C/C=C/CCCC |
| 15 | 2-Methyl-3-octanone | O=C(CCCCC)C(C)C |
| 16 | 2-Hexylfuran | o1c(ccc1)CCCCCC |
| 17 | 6-Methyl-5-hepten-2-one | O=C(C)CC\C=C(/C)C |
| 18 | 1-Hexanol | OCCCCCC |
| 19 | 2-Nonanone | O=C(CCCCCCC)C |
| 20 | Nonanal | O=CCCCCCCCC |
| 21 | 7-Methyl-2-decene | C(=C/CCCC(CCC)C)\C |
| 22 | (*E*)-2-Octenal | O=C/C=C/CCCCC |
| 23 | 2-Heptylfuran | o1c(ccc1)CCCCCCC |
| 24 | (*E*)-4-Nonenal | O=CCC/C=C/CCCC |
| 25 | (*E*)-2-Octen-1-ol | OC/C=C/CCCCC |
| 26 | 1-Octen-3-ol | OC(\C=C)CCCCC |
| 27 | 1-Heptanol | OCCCCCCC |
| 28 | (*E*,*E*)-2,4-Heptadienal | O=C\C=C\C=C\CC |
| 29 | 2-Decanone | O=C(CCCCCCCC)C |
| 30 | Decanal | O=CCCCCCCCCC |

**Fig. 3.3** SMILES notation of different chemical compounds

In fig. 3.1 and 3.2 SMILES of the given chemical compound is mentioned along with thestructure. In fig. 3.3 SMILES of all types of chemical compounds are listed.

**B.** The second step is to generate an adjacency matrix. Adjacency matrix helps to specifywhether there is a bond between any two atoms present in the given carbon compound.

- If there is a bond between two elements of the given carbon compound, it'srepresented in the adjacency matrix by value '1'. Otherwise '0'.

- It's a N X N square symmetric matrix. The diagonal entries of the adjacencymatrix are all 0's as the compound doesn't bond with itself.
- In the adjacency matrix, the upper and the lower diagonal elements are equal.

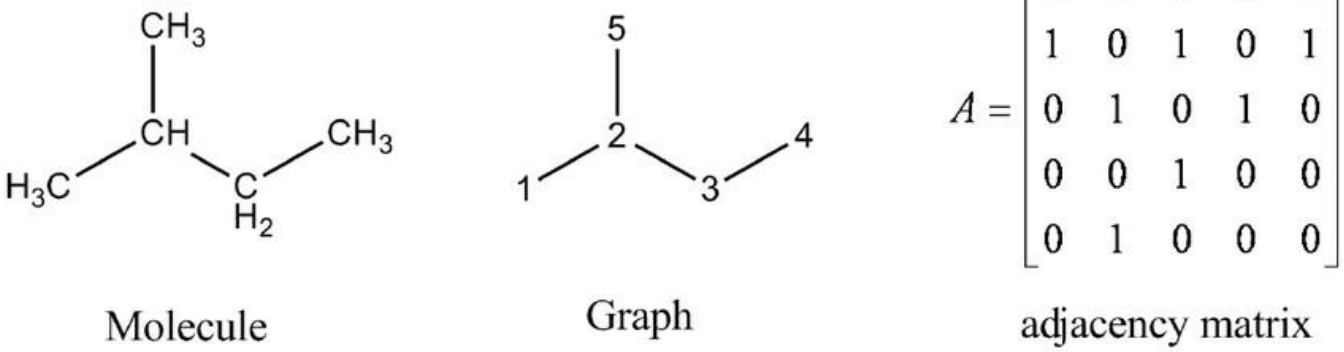
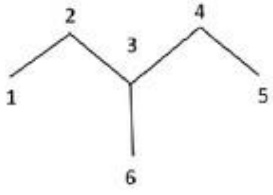



**Fig. 3.4** Adjacency Matrix of the chemical compound 'CC(C)CC'

In the fig. 3.4 adjacency matrix of the chemical compound 'CC(C)CC is generated.

**C.** The third step involves generation of the distance matrix. The distance matrix D(G) of a molecular graph G is a real N X N square symmetric matrix, where N is the number of atoms (vertices) in a graph. The distance matrix contains elements Dij representing the number of edges (length) between the vertices i and j along with the shortest path betweenthem. For all the diagonal elements, the value is '0'as there doesn't exist any self-loop.




| $C_i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 3 | 4 | 3 |
| 2 | 1 | 0 | 1 | 2 | 3 | 2 |
| 3 | 2 | 1 | 0 | 1 | 2 | 1 |
| 4 | 3 | 2 | 1 | 0 | 1 | 2 |
| 5 | 4 | 3 | 2 | 1 | 0 | 3 |
| 6 | 3 | 2 | 1 | 2 | 3 | 0 |

**Fig. 3.5** Distance Matrix of the chemical compound 'CCC(C)CC'

In the fig. 3.5 distance matrix of the chemical compound 'CCC(C)CC' is generated.

**D.** The fourth step is to calculate the molecular descriptors. It can be further divided astopological and topochemical indices.

A topological index is a mathematical descriptor of the structure of a molecule that is based on its topology or connectivity. Topological indices are used to characterize various molecular properties, such as boiling point, solubility, toxicity, and biological activity. They include the Wiener Index, Zagreb Index and the Molecular Connectivity Index.

## 1. ZAGREB INDEX:

The Zagreb index is a graph-theoretical index used in chemistry to describe the molecularstructure of a compound. It is calculated based on the number of vertices (atoms) and edges (bonds) present in the molecule.

- The first Zagreb index, denoted by M1, is defined as the sum of the squares of the degrees of all vertices in a molecule. The degree of a vertex is the number ofedges connected to that vertex.
- The second Zagreb index, denoted by M2, is defined as the sum of the product of the degrees of pairs of adjacent vertices in a molecule.

The Zagreb indices can be used to predict various properties of molecules, such as boiling point, viscosity, and biological activity. M1 and M2 have been found to be particularly useful in predicting the octanol/water partition coefficient, which is a measure of a compound's hydrophobicity and is important for predicting its absorption and distribution in the body. The Zagreb indices have also been used to develop quantitative structure- activity relationship (QSAR) models, which are used to predict the activity or toxicity of compounds based on their molecular structure.

## 2. MOLECULAR CONNECTIVITY INDEX:

The molecular connectivity index is a topological index used in chemistry to describe the structure of a molecule. The molecular connectivity index is calculated by summing the contributions of individual atoms in a molecule based on their connectivity to other atoms.This provides information

about the molecular structure and properties of a compound. A higher molecular connectivity index value indicates a more complex structure, while a lower molecular connectivity index value suggests a simpler structure. Also, this is used to predict certain physical properties of molecules, such as boiling point, and solubility andbiological activity.

| Atom | $(Z^{v} - h) / (Z - Z^{v} - 1)$ | $\delta^{v}$ |
|---|---|---|
| $-CH_3$ | (4-3)/(6-4-1) | 1 |
| $-CH_2-$ | (4-2)/(6-4-1) | 2 |
| $=CH_2$ | (4-1)/(6-4-1) | 2 |
| $CR_4$ | (4-0)/(6-4-1) | 4 |
| $-NH_2$ | (5-2)/(7-5-1) | 3 |
| -NH- | (5-1)/(7-5-1) | 4 |
| $NR_3$ | (5-0)/(7-5-1) | 5 |
| -OH | (6-1)/(8-6-1) | 5 |
| -O- | (6-0)/(8-6-1) | 6 |
| =O | (6-0)/(8-6-1) | 6 |
| -F | (7-0)/(9-7-1) | 7 |
| -Cl | (7-0)/(17-7-1) | 0.78 |
| -Br | (7-0)/(35-7-1) | 0.26 |
| -I | (7-0)/(53-7-1) | 0.16 |
| -SH | (6-1)/(16-6-1) | 0.56 |
| -S- | (6-0)/(16-6-1) | 0.67 |

**Fig. 3.6** MCI for various elements

In the fig. 3.6 the MCI for various elements is displayed.

## 3. WIENER INDEX:

The Wiener index, also known as the Wiener number, is a topological index used in graph theory and chemistry to describe the molecular structure of a compound. It is named after mathematician Harry Wiener, who introduced it in 1947.The Wiener index is calculated by summing the lengths of the shortest paths between all pairs of vertices (atoms) in a molecule. In other words, it represents the sum of the distances between all pairs of atomsin a molecule. The Wiener index provides information about the overall size and branching of a molecule. It is related to various molecular properties, such as boiling.

point, melting point, and reactivity. For example, there is a correlation between the Wiener index and the boiling point of hydrocarbons, where larger Wiener index values are associated with higher boiling points. The Wiener index has applications in various areas, including quantitative structure-property relationship (QSPR) studies, drug design, and chemical graph theory. It can be used to analyze and compare the structural complexity of different molecules, as well as to develop predictive models for various physical and chemical properties of compounds.

| Sl. No. | Topological Index | Notation | Formula of Topological Indices |
|---|---|---|---|
| 1. | First Zagreb Index [26] | $M_1(G)$ | $M_1(G) = \sum_{uv \in E(G)} (d(u) + d(v))$ |
| 2. | Second Zagreb Index [26] | $M_2(G)$ | $M_2(G) = \sum_{uv \in E(G)} (d(u)d(v))$ |
| 3. | Modified Second Zagreb Index [27] | ${}^{m}M_2(G)$ | ${}^{m}M_2(G) = \sum_{uv \in E(G)} \frac{1}{d(u)d(v)}$ |
| 4. | General Randić Index [29] | $R_\alpha(G)$ | $R_\alpha(G) = \sum_{uv \in E(G)} (d(u)d(v))^\alpha$ |
| 5. | Inverse Randić Index [30] | $RR_\alpha(G)$ | $RR_\alpha(G) = \sum_{uv \in E(G)} \frac{1}{(d(u)d(v))^\alpha}$ |
| 6. | Symmetric Division (Deg) Index [31] | $SDD(G)$ | $SDD(G) = \sum_{uv \in E(G)} \left\{ \frac{\min(d(u),d(v))}{\max(d(u),d(v))} + \frac{\max(d(u),d(v))}{\min(d(u),d(v))} \right\}$ |
| 7. | Harmonic Index [37] | $H(G)$ | $H(G) = \sum_{uv \in E(G)} \frac{2}{d(u)+d(v)}$ |
| 8. | Inverse Sum (Indeg) Index [31] | $ISI(G)$ | $ISI(G) = \sum_{uv \in E(G)} \frac{d(u)d(v)}{d(u)+d(v)}$ |
| 9. | Augmented Zagreb Index [33] | $AZ(G)$ | $AZ(G) = \sum_{uv \in E(G)} \left\{ \frac{d(u)d(v)}{d(u)+d(v)-2} \right\}^3$ |

**Fig. 3.7** List of various topological indices with the formulae

In the fig. 3.7 various topological indices are displayed along with their formulae.

A topochemical index is a class of topological indices that are used to describe the electronic and magnetic properties of organic molecules. Unlike other topological indices which focus on atom distances and bond connectivity, topochemical indices take into account the spatial distribution of electrons in a molecule. Its types are Aromaticity Index, Huckel's Rule Index, Randic Connectivity

Index, Balaban Index, Electronic-Topological State Index, and Molecular Orbital Topology Index.

- **AROMATICITY INDEX:**

  The aromaticity index is a topochemical index used to quantify the degree of aromaticity in a molecule. It is based on the concept of π-electron delocalizationin conjugated systems. The aromaticity index provides a numerical measure of the aromatic character of a molecule, which is useful in studying aromatic compounds, understanding their stability, and predicting their reactivity.

- **HUCKEL'S RULE INDEX:**

  Huckel's rule index, also known as the Huckel molecular orbital index, is a topochemical index derived from Huckel's rule in quantum chemistry. It is used to assess the aromatic character of cyclic conjugated systems. The index is calculated based on the number of π-electrons in the system and can predict the aromaticity or antiaromaticity of molecules. It finds applications in the study ofaromaticity, organic chemistry, and theoretical calculations.

- **RANDIC CONNECTIVITY INDEX:**

  The Randic connectivity index is a topochemical index that characterizes the connectivity patterns in a molecular graph. It is calculated based on the number of connected atom pairs and their bond orders. The Randic connectivity index provides information about the branching and connectivity of a molecule and is useful in quantitative structure-activity relationship (QSAR) studies, molecularproperty prediction, and chemical graph theory.

- **BALABAN INDEX:**

  The Balaban index, also known as the Balaban distance connectivity index, is atopochemical index that describes the shape and symmetry of a molecule. It is calculated based on the distances between pairs of atoms in the molecular graph. The Balaban index is used to assess the molecular complexity, ring strain, and aromaticity of compounds. It finds applications in chemical graph theory, molecular structure analysis, and drug design.

- **ELECTRONIC-TOPOLOGICAL STATE INDEX:**

  The Electronic-Topological State Index (ETSI) is a topochemical index that combines information about the electronic structure and topology of a molecule.It is calculated based on the number of atoms, bonds, and valence electrons ina molecule. ETSI provides insights into the electronic and topological properties of compounds, aiding in the prediction of various molecular properties, including reactivity, stability, and bioactivity.

- **MOLECULAR ORBITAL TOPOLOGY INDEX:**

  The Molecular Orbital Topology Index (MOTI) is a topochemical index that describes the distribution and topology of molecular orbitals in a molecule. It captures the information about bonding interactions and electron density distribution. MOTI is useful in studying molecular electronic structure, predicting reactivity, and understanding the electronic properties of compounds.

These topochemical indices have various applications in molecular property prediction, drug design, chemical reactivity analysis, and theoretical calculations. They provide quantitative measures and insights into the structural, electronic, and aromatic properties of molecules, aiding in the understanding and prediction of their behavior.

List of molecular descriptors computed to assign weights (δ) for the substituents

| | |
|---|---|
| **Graph theoretical descriptors** | |
| $I^{W}{}_{D}$ | Information index for the magnitudes of distances between all possible pairs of vertices of a graph |
| $I_h{}^{W}{}_{D}$ | Mean information index for the magnitude of distance |
| **W** | Wiener index = half-sum of the off-diagonal elements of the distance matrix of a graph |
| **H** | Harary index = half-sum of reciprocal square of the elements of the distance matrix of a graph |
| **J** | Balaban index = sum of the reciprocal square root of degree of all adjacent pairs |
| $Ic^{D}$ | Information content of the distance matrix |
| **IC** | Information content of the distance matrix partitioned by frequency of occurrences of distance |
| **O** | Order of neighbourhoods when ICr reaches its maximum value for the hydrogen-filled graph |
| **M1** | Zagreb group parameter = sum of square of degree over all vertices |
| **M2** | Zagreb group parameter = sum of cross-product of degrees over all neighbouring (connected) vertices M2 |
| **Topo-structural descriptors** | |
| $^{h}\chi$ | Path connectivity index of order $h$ = 0-6 |
| $^{h}\chi_{c}$ | Cluster connectivity index of order $h$ ) 3-6 |
| $^{h}\chi_{ch}$ | Chain connectivity index of order $h$ ) 3-6 |
| $^{h}\chi_{c}$ | Path-cluster connectivity index of order $h$ ) 4-6 |
| $P_h$ | Number of paths of length $h$ = 0-10 |
| **Topo-chemical descriptors (bond-order connectivity)** | |
| $^{h}\chi^{b}$ | Bond path connectivity index of order $h$ = 0-9 |
| $^{h}\chi_{c}{}^{b}$ | Bond cluster connectivity index of order $h$ = 3-6 |
| $^{h}\chi_{ch}{}^{b}$ | Bond chain connectivity index of order $h$ = 3-6 |
| $^{h}\chi_{Pc}{}^{b}$ | Bond path-cluster connectivity index of order $h$ = 4-6 |
| **Topo-chemical descriptors (valence connectivity)** | |
| $^{h}\chi^{v}$ | Valence path connectivity index of order $h$ = 0-9 |
| $^{h}\chi_{c}{}^{v}$ | Valence cluster connectivity index of order $h$ = 3-6 |
| $^{h}\chi_{ch}{}^{v}$ | Valence chain connectivity index of order $h$ = 3-6 |
| $^{h}\chi_{Pc}{}^{v}$ | Valence path-cluster connectivity index of order $h$ = 4-6 |

**Fig. 3.8** Various Graph Theoretical Indices

| | |
|---|---|
| **Information theoretic descriptors** | |
| **$I_{ORB}$** | Information content or complexity of the hydrogen suppressed graph at its maximum neighbourhood of vertices |
| **$O_{ORB}$** | Order of neighbourhoods when ICr reaches its maximum value for the hydrogen suppressed graph |
| **$IC_r$** | Mean information content or complexity of a graph based on the $r^{th}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-filled graph |
| **$SIC_r$** | Structural information content for $r^{th}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-filled graph |
| **$CIC_r$** | Complementary information content for $rt^{h}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-filled graph |
| **$TIC_r$** | Total information content for $r^{th}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-filled graph |
| **$BIC_r$** | Bond information content for $r^{th}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-filled graph |
| **$H_SIC_r$** | Mean information content or complexity of a graph based on the $r$th ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-suppressed graph |
| **$H_SSIC_r$** | Structural information content for $r^{th}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen-suppressed graph |
| **$H_SCIC_r$** | Complementary information content for $r^{th}$ ($r$ = 0-6) order neighbourhood of vertices in a hydrogen suppressed graph |
| **$H_STIC_r$** | Total information content for $r^{th}$ ($r$ = 0-6) order neighborhood of vertices in a hydrogen-suppressed graph |
| **$H_SBIC_r$** | Bond information content for $r^{th}$ ($r$ = 0-6) order neighborhood of vertices in a hydrogen-suppressed graph |
| **Overall connectivity indices** | |
| **SumR** | Sum of connectivity indices of all orders ($\Sigma\,^{h}\chi$) |
| **VSumR** | Sum of valence connectivity indices of all orders ($\Sigma\,^{h}\chi^{v}$) |
| **BSumR** | Sum of bond order connectivity indices of all orders ($\Sigma\,^{h}\chi^{b}$) |
| **OPM** | Overall path multiplicity $\Sigma\,^{h}\chi \times P_h$ |
| **VOPM** | Overall path multiplicity of valence connectivity $\Sigma\,^{h}\chi^{v} \times P_h$ |
| **BOPM** | Overall path multiplicity of bond order connectivity $\Sigma\,^{h}\chi^{b} \times P_h$ |
| **MPC** | Mean path connectivity index $\Sigma\,^{h}\chi \div P_h$ |
| **VMPC** | Mean path connectivity index based on valence connectivity $\Sigma\,^{h}\chi^{v} \div P_h$ |
| **BMPC** | Mean path connectivity index based on bond connectivity $\Sigma\,^{h}\chi^{b} \div P_h$ |
| **SumK** | Sum of all paths of length 1 to h $\Sigma P_h$ |

**Fig. 3.9** Various Information Theoretic Indices

In the fig. 3.8 several types of graph theoretical indices are mentioned. In the fig. 3.9various types of information theoretic indices are specified.

**E.** After the calculation of these indices, the results are displayed on the screen. This is then used as a training set for the machine learning model. The model is trained with the available data and tested with a new set of data. According to the accuracy, the results arepredicted. This is the final step.

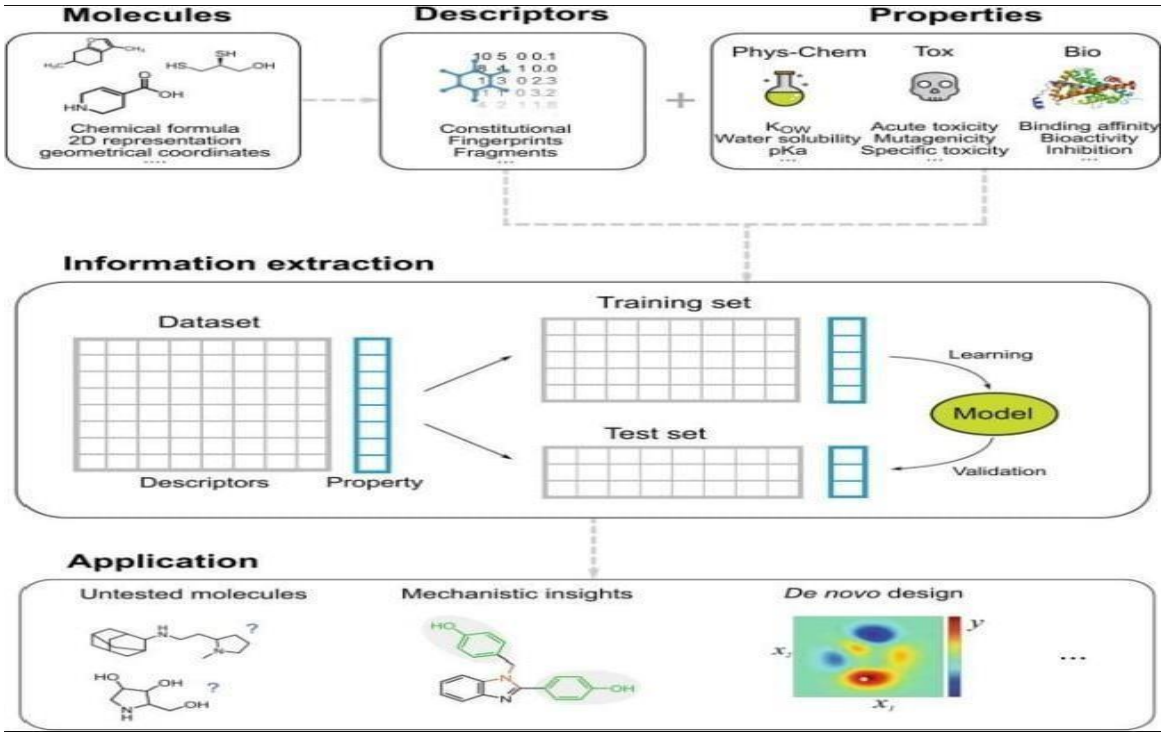



**Fig. 3.10** Architecture of the proposed model

In the fig. 3.10 architecture of the system is displayed.

## 4. SYSTEM ARCHITECTURE:

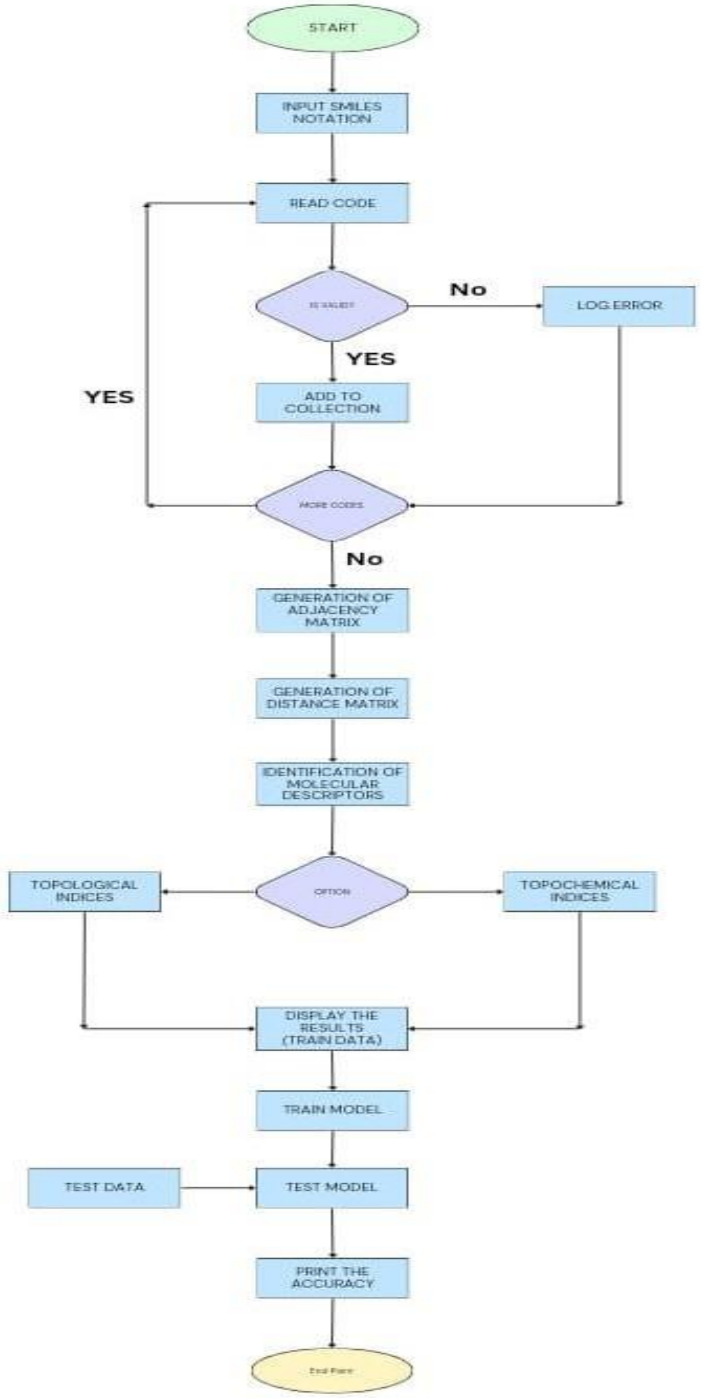



**Fig. 4.1** Workflow of the system

In the fig. 4.1 workflow of the system is shown.

## 4.1 FLOW:

The flow of the fig. 4.1 is explained below.

The implementation of indices calculator consists of the following steps:

1. The SMILES notation is given as input from the user.
2. The SMILES notation is validated to identify whether the given input satisfiesthe rules mentioned. If yes, it is stored in the database otherwise it pops out error.
3. The adjacency matrix is generated for each chemical compound.
4. The distance matrix is generated for each input.
5. The molecular descriptors are calculated. According to the choice of the user, either topological indices or topochemical indices are calculated. The user can also opt for both the choices.
6. The output from the previous step is collected and store in the database. This isused as the train and test data for the machine learning model.
7. The machine learning model is built and trained with the available data.
8. The model is tested and accuracy is analyzed.
9. This further predicts the properties of the chemical compounds that is given as input. With this, properties of unknown chemical compounds can be predicted.

## 5. USE CASE DIAGRAM:

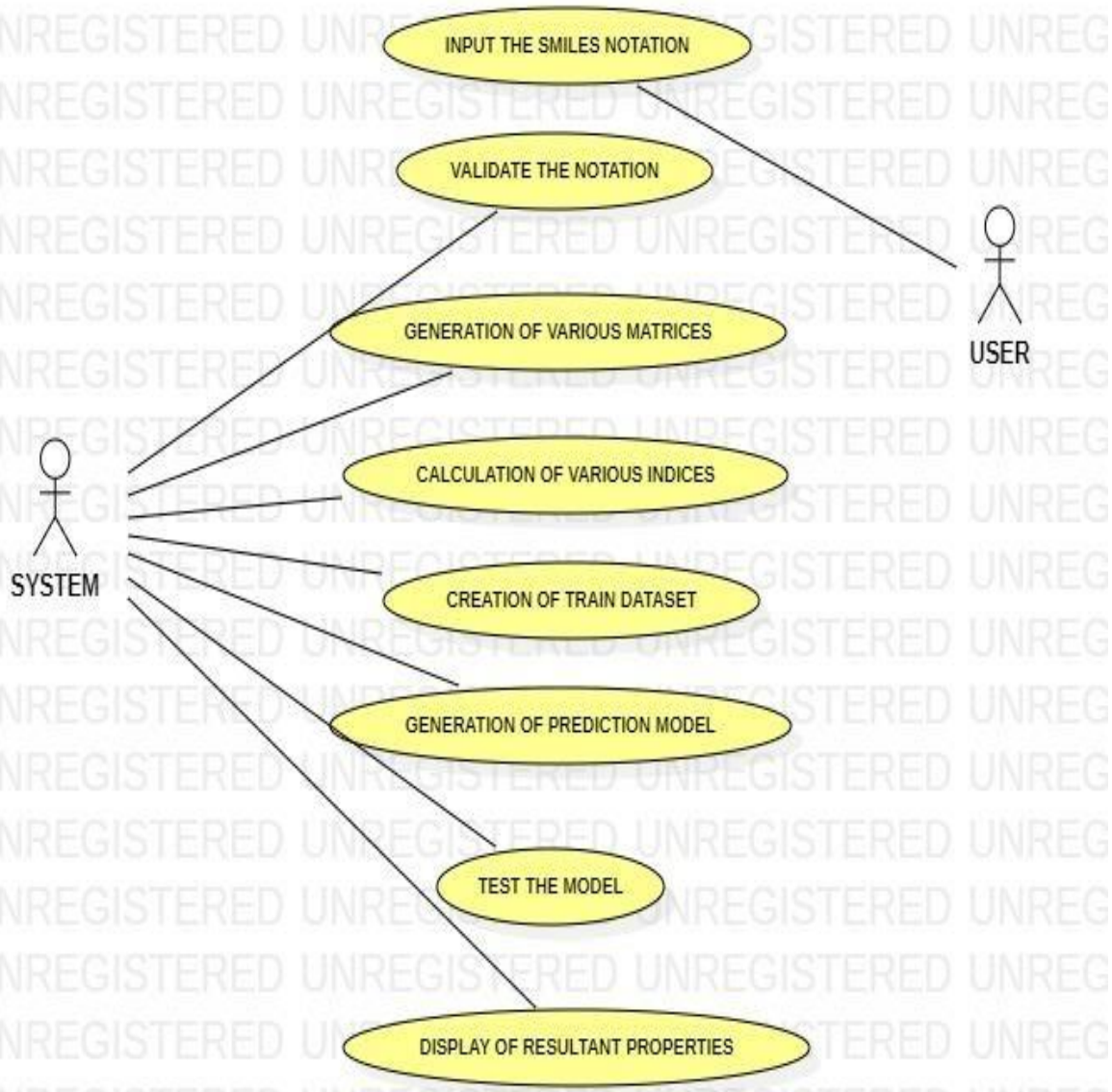



**Fig. 5.1** Use case diagram of the application

Fig. 5.1 depicts the use case diagram for the indices calculator. Since it is a sequentialprocess, each segment depends on the other i.e., each must include the previous one.

# 6. REQUIREMENTS:

## 6.1. FUNCTIONAL REQUIREMENTS:

- The system must get the input from the user and validate it.
- System should generate the adjacency matrix and distance matrix.
- The system should calculate various types of indices.
- System should generate the train and test data set.
- The system should predict the physicochemical and biological properties ofunknown chemical compounds.

## 6.2. SOFTWARE REQUIREMENTS:

- Language : C#
- Tool : Visual Studio Code
- OS : Windows

## 6.3. NON-FUNCTIONAL REQUIREMENTS:

**6.3.1 Performance:** The properties of the chemical compounds will be recognized with anaccuracy of about 95.

**6.3.2 Functionality:** This software will deliver on the functional requirements.

**6.3.3 Availability:** This system will predict the properties of chemical compounds only ifthe SMILES is correctly given as the input.

**6.3.4 Flexibility:** It provides the users to load the input easily.

**6.3.5 Learnability:** The software is very easy to use and reduces the learning work.

## 6.4 PERFORMANCE REQUIREMENTS:

**6.4.1 Response Time:** The system shall take some time for processing but not more than30 seconds.

**6.4.2 Capacity:** The system must accept and process only one input at a time.

## 7. MODULE EXPLANATION:

It has four main modules:

1. Validate the input
2. Generate various matrices
3. Calculation of molecular descriptors
4. Building the machine learning model

## 7.1 VALIDATE THE INPUT:

The user gives structure of the chemical compound as the input in the form of SMILES. SMILES (Simplified Molecular Input Line Entry System) is a string-based notation system used to represent the structure of chemical molecules. It is a compact and human-readable format widely used in cheminformatics and computational chemistry. The validation of a SMILES string involves verifying its syntax and ensuring that it conformsto the defined rules and standards of the SMILES notation. Validating SMILES strings iscrucial for ensuring accurate molecular representation and reliable data analysis in various scientific and industrial applications.

## 7.2 GENERATE VARIOUS MATRICES:

The matrices generated are adjacency matrix and distance matrix. The adjacency matrix is constructed by assigning a value of 1 to each pair of connected atoms in the graph and 0 to non-connected pairs. The distance matrix is derived from the adjacency matrix by computing the shortest path distances between every pair of atoms in the molecule. Thesematrices provide valuable information about the connectivity

and spatial relationships within the molecule, facilitating various analyses and computations in cheminformatics and molecular modeling.

## 7.3 MOLECULAR DESCRIPTORS:

Molecular descriptors are numerical representations derived from molecular structures and properties. From the adjacency matrix and distance matrix, various molecular descriptors can be calculated. These include topological and topochemical descriptors that capture information about connectivity patterns, such as atom and bond counts, graph indices, and molecular weight. Additionally, 3D descriptors can be computed fromthe distance matrix, such as molecular volume, surface area, and shape descriptors. These descriptors provide quantitative information about the molecular structure, aiding in taskslike chemical similarity analysis, property prediction, and drug design.

## 7.4 BUILDING THE MACHINE LEARNING MODEL;

First, a dataset of chemical compounds with known properties and their corresponding molecular descriptors is prepared. Next, the dataset is divided into training and testing sets.Then, an ML algorithm, such as random forest, support vector machine, or neural network,is chosen and trained using the training data. The trained model is then evaluated using thetesting data, and its performance is assessed using appropriate metrics. Finally, the model can be used to predict the properties of unknown compounds based on their molecular descriptors, enabling efficient virtual screening and property prediction in drug discoveryand material science applications.

## 8. TEST CASES:

**Table 8.1** Test cases

| TEST ID | TEST SCENARIO | TEST STEPS | TEST DATA | EXPECTED RESULTS | ACTUAL RESULTS | PASS/ FAIL | SIGN |
|---|---|---|---|---|---|---|---|
| 1. | Predict the properties of compound 'CCCC' | Open the application and give 'CCCC' as input | Given input: 'CCCC' | Zagreb Index is 1.9 and MCI is 1.9. | Zagreb Index is 1.9 and MCI is 1.9. | **PASS** | |
| 2. | Predict the properties of compound 'CC(C)C' | Open the application and give 'CC(C)C' as input | Given input: 'CC(C)C' | Zagreb Index is 1.7 and MCI is 1.7 | Zagreb Index is 1.7 and MCI is 1.7 | **PASS** | |
| 3. | Predict the properties of compound 'C(O)C' | Open the application and give 'C(O)C' as input | Given input: 'C(O)C' | Zagreb Index is 1.4 and MCI is 1.4. | Zagreb Index is 1.4 and MCI is 1.4. | **PASS** | |
| 4. | Predict the properties of compound 'CCCCl' | Open the application and give 'CCCCl' as input | Given input: 'CCCCl' | Zagreb Index is 1.9 and MCI is 1.9. | Zagreb Index is 1.9 and MCI is 1.9. | **PASS** | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5. | Predict the properties of compound 'C1CCCCC1' | Open the application and give 'C1CCCCC1' as input | Given input: 'C1CCCCC1' | Zagreb Index is 3 and MCI is 3. | Zagreb Index is 3 and MCI is 3. | **PASS** | |

## 9. CONCLUSION:

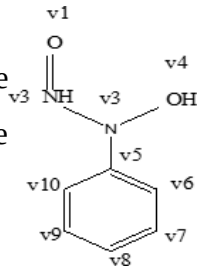
In this study, we aimed to utilize molecular descriptors to elucidate the physical, chemical, and biological properties of a specific chemical compound. Our objective was to streamline the work of scientists and researchers by employing cutting-edge technologies to identify the characteristics and applications of unknown chemical compounds. By leveraging molecular descriptors, we sought to predict the properties and usages of these compounds without the need for extensive research. This approach has the potential to significantly reduce the time and effort invested in traditional experimental methods. Ultimately, our work aimed to enhance efficiency and enable rapid predictions of compound properties, thereby facilitating advancements in various scientific and industrial fields.

## 10. REFERENCES:


- Machine Learning Prediction of Nine Molecular Properties Based on theSMILES Representation of the QM9 Quantum-Chemistry Dataset.
- SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular PropertyPrediction.
- A Novel Molecular Representation Learning for Molecular Property Predictionwith a Multiple SMILES-Based Augmentation.
- Machine learning prediction of empirical polarity using SMILES encoding oforganic solvents.
- Identifying Structure–Property Relationships through SMILES Syntax Analysiswith Self-Attention Mechanism.

## 11. SCREENSHOTS:

Let the sample carbon compound be CUPFERRON (NH4[C6H5N(O)NO]). The structure of the chemical compound is displayed. The SMILES notation for the compound will be O=N-N(O)C1CCCCC1.



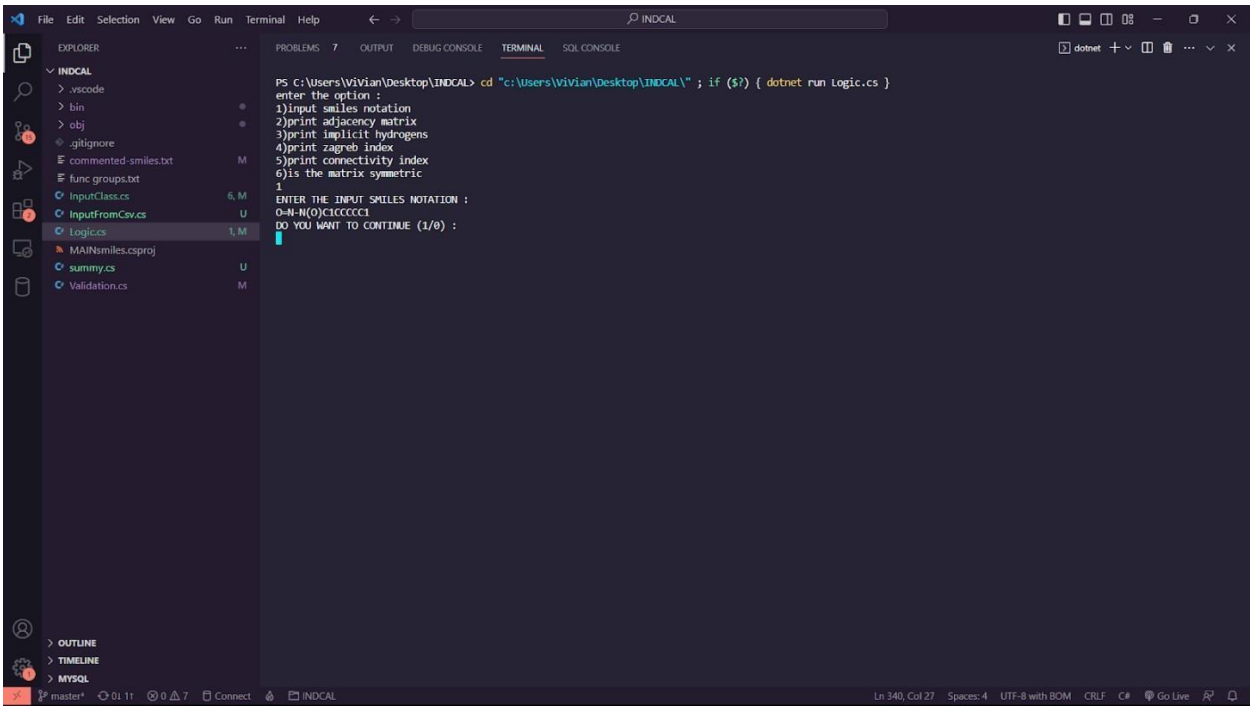




**Fig. 11.1** Option menu with available operations.

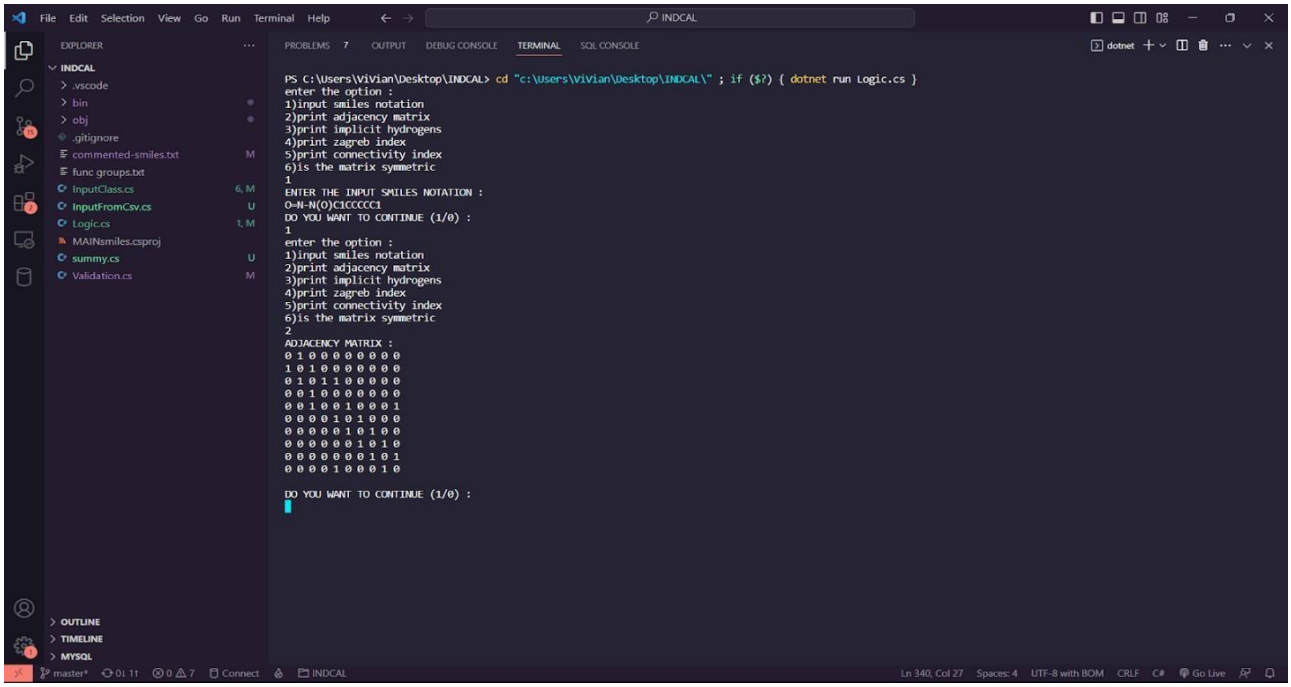



**Fig. 11.2** Adjacency matrix of Cupferron.

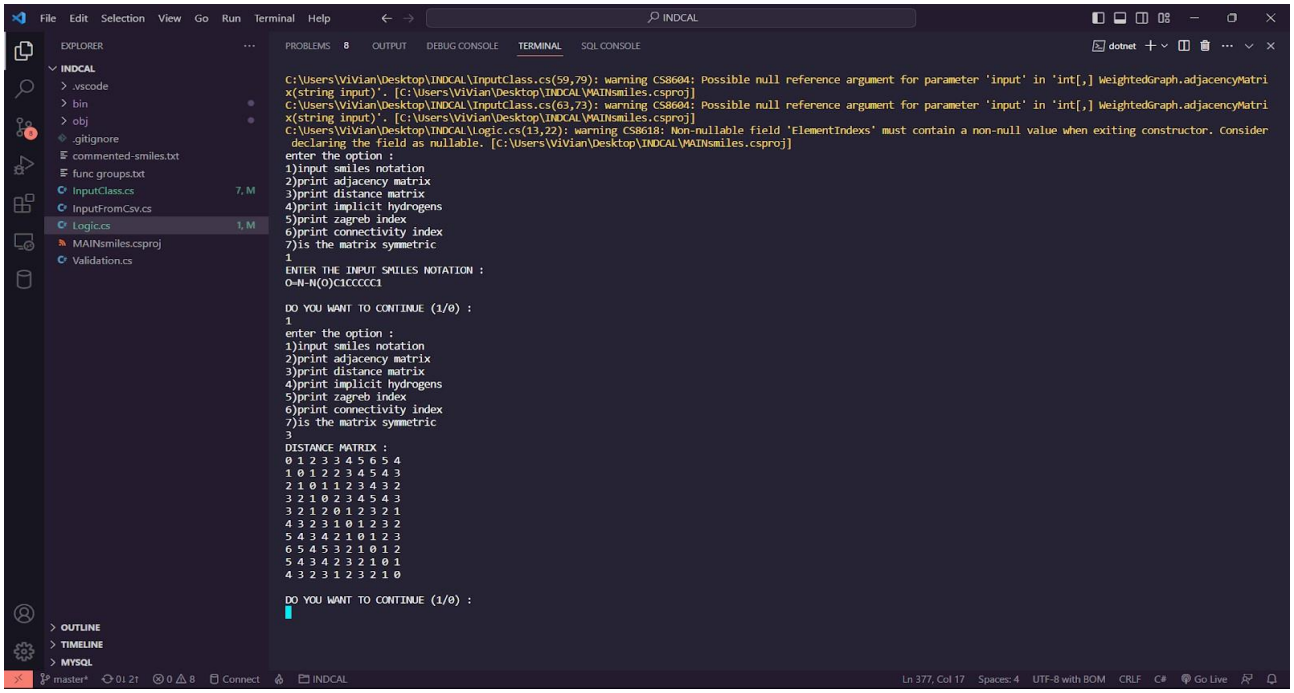



**Fig.11.3** Distance Matrix of Cupferron.

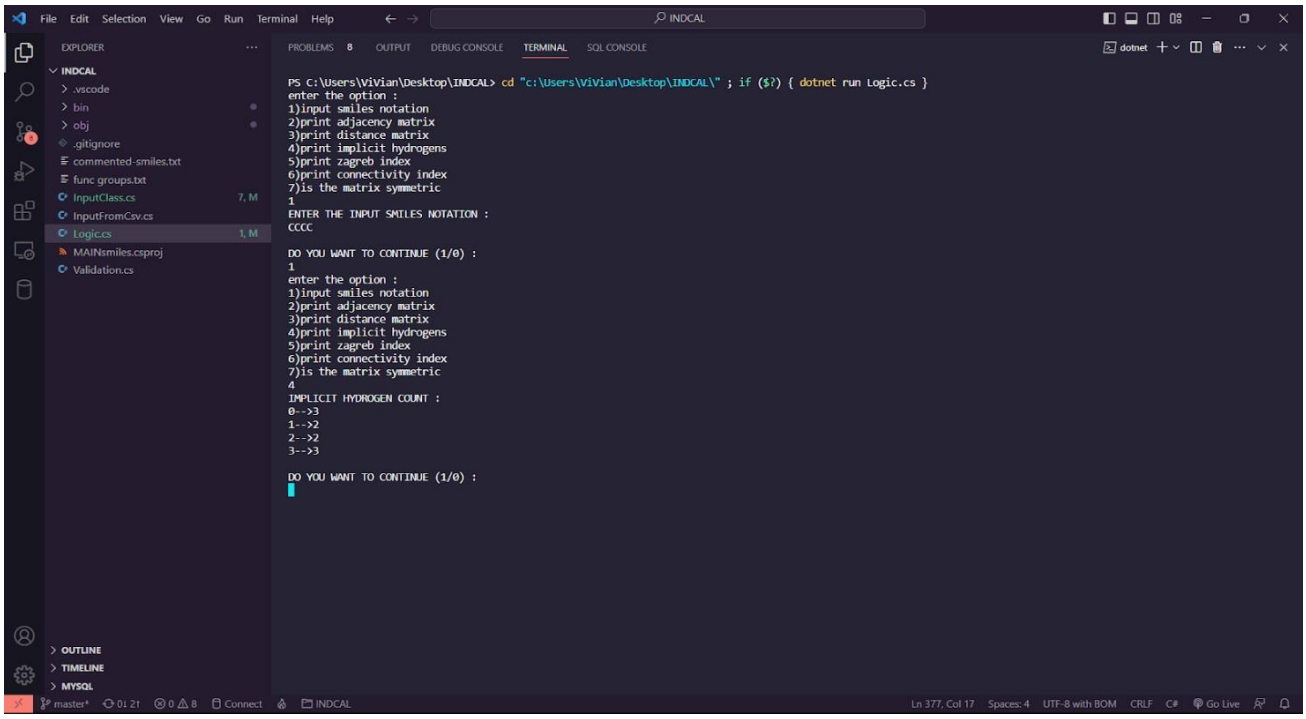



**Fig. 11.4** Implicit Hydrogen Count of Cupferron.

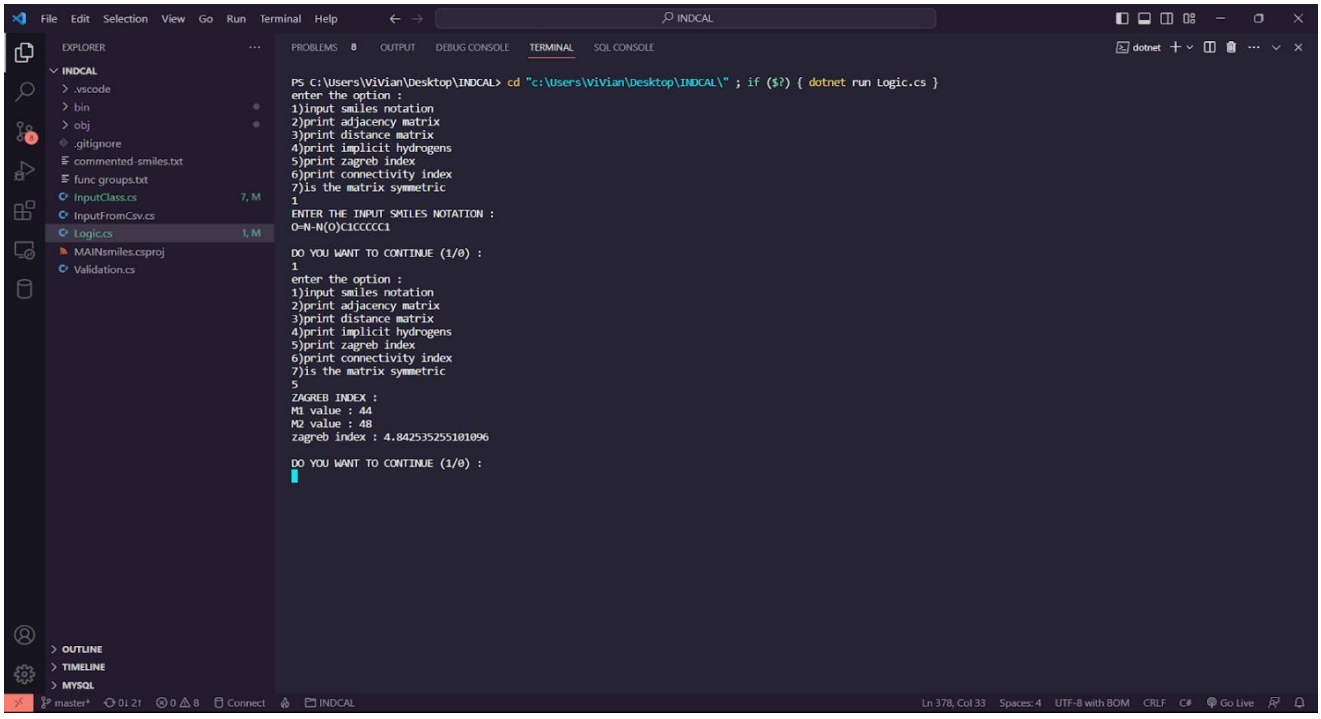



**Fig. 11.5** Zagreb Index of Cupferron.

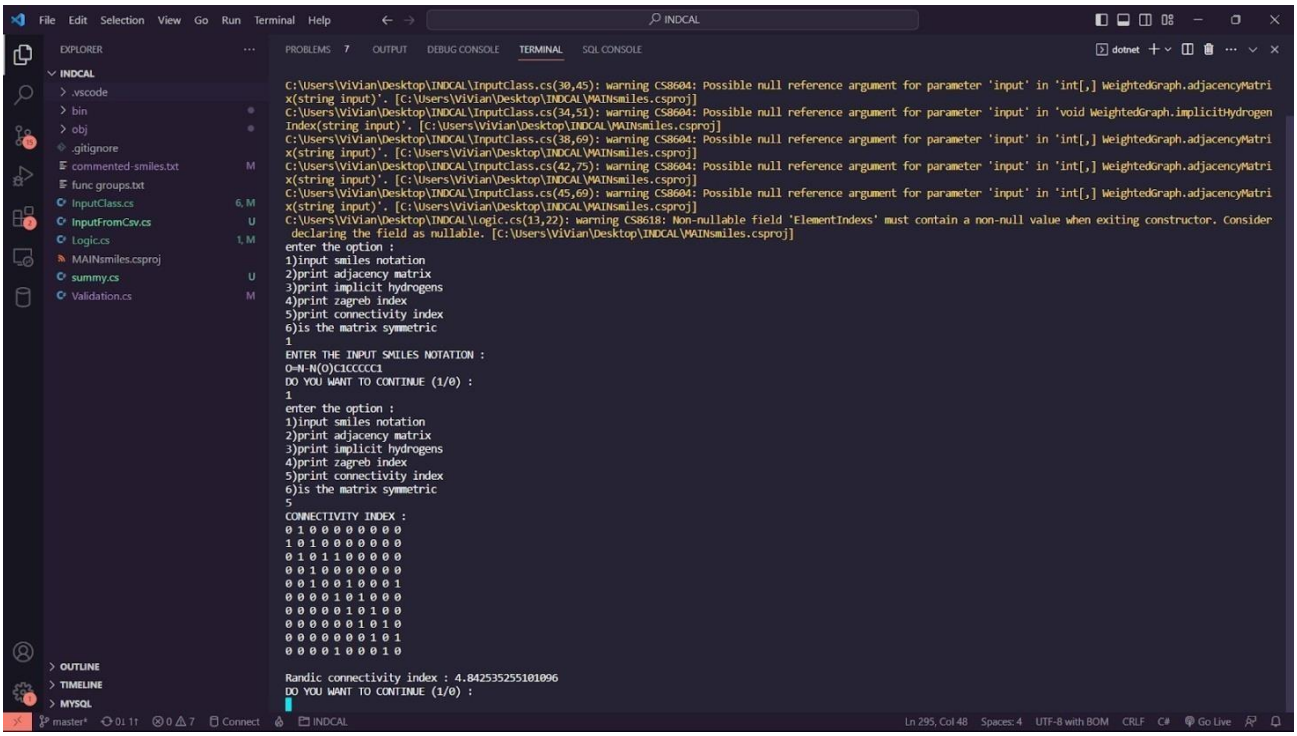



**Fig. 11.6** Molecular Connectivity Index

From the fig. 11.1 to fig. 11.6 the following works are displayed:

- Option menu with available operations.
- Adjacency matrix.
- Distance matrix.
- Implicit Hydrogen Count.
- Zagreb Index.
- Molecular Connectivity Index.

## 12. CODE:

## Input from CSV file:

```
using System;
using System.Collections.Generic;using
System.Linq;
using System.Text;
using System.Threading.Tasks;using
System.IO;
using Logic;


namespace ReadingDataFromCSV
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string filePath = @"C:\Users\ViVian\Downloads\smilessss1 (1).csv";StreamReader reader =
            null;
            int j=-1;
            List<WeightedGraph> inputobj = new List<WeightedGraph>();if
            (File.Exists(filePath)){
                reader = new StreamReader(File.OpenRead(filePath));List<string> listA
                = new List<string>();
                while (!reader.EndOfStream){ var line
                    = reader.ReadLine(); var values =
                    line.Split(',');
                    Console.WriteLine(values[0]);
```

```
                j+=1;
                Console.WriteLine(j+"th time"); inputobj.Add(new
                WeightedGraph());
                inputobj[j].initiateVertex(values[0]);
                inputobj[j].readSmiles(values[0]);
                inputobj[j].adjacencyMatrix(values[0]);
                // foreach (var item in values){
                // listA.Add(item);
                // }
                // foreach (var coloumn1 in listA){
                // Console.Write(coloumn1);
                // Console.WriteLine();
                // }
            }
        } else {
            Console.WriteLine("File doesn't exist");
        }
        Console.ReadLine();
    }
  }
}
```

## Input class:

```
using Logic; class
inputClass
{
    static void Main(string[] args)
    {
        List<WeightedGraph> inputobj = new List<WeightedGraph>();int option ;
        int j=-1;
        string inputSmiles = "";do{
          Console.WriteLine("enter the option : \n1)input smiles notation\n2)print adjacency
          matrix\n3)print distance matrix\n4)print implicit hydrogens\n5)print zagreb index\n6)print
          connectivity index\n7)is the matrix symmetric");
         option =Convert.ToInt32(Console.ReadLine());switch
        (option)
        {
            case 1:
                Console.WriteLine("ENTER THE INPUT SMILES NOTATION : ");
                inputSmiles = Console.ReadLine();
                Validation validate = new Validation();
                if(!string.IsNullOrEmpty(inputSmiles) && validate.isValid(inputSmiles))
                {
                    j+=1;
                    inputobj.Add(new WeightedGraph());
                    inputobj[j].initiateVertex(inputSmiles);
                    inputobj[j].readSmiles(inputSmiles);
                }
                else Console.WriteLine("ENTER A PROPER INPUT");
```

```
    Console.WriteLine();
    break;
case 2:
    Console.WriteLine("ADJACENCY MATRIX : ");
    int[,] adjacencyMatrix = inputobj[j].adjacencyMatrix(inputSmiles);for(int
    i=0;i<adjacencyMatrix.GetLength(0);i++)
    {
        for(int k=0;k<adjacencyMatrix.GetLength(0);k++)
        {
            Console.Write(adjacencyMatrix[i,k]+" ");
        }

        Console.WriteLine();
    }
    Console.WriteLine();
    break;
case 3:
    Console.WriteLine("DISTANCE MATRIX : ");

   inputobj[j].GenerateDistanceMatrix(inputobj[j].adjacencyMatrix(inputSmiles));
    Console.WriteLine();
    break;
case 4:
    Console.WriteLine("IMPLICIT HYDROGEN COUNT : ");
    inputobj[j].implicitHydrogenIndex(inputSmiles);
    Console.WriteLine();
    break;
case 5:
    Console.WriteLine("ZAGREB INDEX : ");
```

```
            inputobj[j].zagrebIndex(inputobj[j].adjacencyMatrix(inputSmiles)); Console.WriteLine();
            break;
        case 6:
            Console.WriteLine("CONNECTIVITY INDEX : ");
            inputobj[j].connectivityIndex(inputobj[j].adjacencyMatrix(inputSmiles)); Console.WriteLine();
            break;
        case 7:
            inputobj[j].isSymmetric(inputobj[j].adjacencyMatrix(inputSmiles)); Console.WriteLine();
            break;
        default:
            Console.WriteLine("INVALID OPTION");
           Console.WriteLine();
            break;
    }
    Console.WriteLine("DO YOU WANT TO CONTINUE (1/0) :");
    option = Convert.ToInt32(Console.ReadLine());
    }while(option>0);
  }
}
```

## Validation Class:

```
class Validation
{
    public static readonly Dictionary<string,int> Elements = new Dictionary<string, int>()
    {
        {"H",1},{"He",2},{"Li",1},{"Be",2},{"B",3},{"C",4},{"N",5},
        {"O",6},{"F",7},{"Ne",8},{"Na",1},{"Mg",2},{"Al",3},{"Si",5},
        {"P",5},{"S",6},{"Cl",7},{"Ar",8},{"K",1},{"Ca",2},{"Sc",3},
        {"Ti",4},{"V",5},{"Cr",6},{"Mn",7},{"Fe",8},{"Co",9},{"Ni",10},
        {"Cu",11},{"Zn",12},{"Ga",3},{"Ge",3},{"As",5},{"Se",6},{"Br",7},
        {"Kr",8},{"Rb",1},{"Sr",2},{"Y",3},{"Zr",4},{"Nb",5},{"Mo",6},
        {"Tc",7},{"Ru",8},{"Rh",9},{"Pd",10},{"Ag",11},{"Cd",12},{"In",3},
        {"Sn",4},{"Sb",5},{"Te",6},{"I",10},{"Xe",8},{"Cs",1},{"Ba",2},
        {"La",3},{"Ce",4},{"Pr",5},{"Nd",6},{"Pm",7},{"Sm",8},{"Eu",9},
        {"Gd",10},{"Tb",11},{"Dy",12},{"Ho",13},{"Er",14},{"Tm",15},{"Yb",16},
        {"Lu",3},{"Hf",4},{"Ta",5},{"W",6},{"Re",7},{"Os",8},{"Ir",9},
        {"Pt",10},{"Au",11},{"Hg",12},{"Pb",4},{"Bi",5},{"Po",6},
        {"At",7},{"Rn",8},{"Fr",1},{"Ra",2},{"Ac",3},{"Th",4},{"Pa",5},
        {"U",6},{"Np",7},{"Pu",8},{"Am",9},{"Cm",10},{"Bk",11},{"Cf",12},
        {"Es",13},{"Fm",14},{"Md",15},{"No",16},{"Lr",3},{"Rf",4},{"Db",5},
        {"Sg",6},{"Bh",7},{"Hs",8},{"Mt",9},{"Ds",10},{"Rg",11},{"Cn",12},
        {"Nh",3},{"Fl",4},{"Mc",5},{"Lv",6},{"Ts",7},{"Og",8}
    };
```

```
public static readonly Dictionary<char,int> bonds = new Dictionary<char, int>()
{
    {'-',1},{'=',2},{'#',3},{'$', 4},{'.', 0},{':',1}
};

public bool isValid(string input)
{
    char[] res = input.ToCharArray();
    bool isRing=false,isBranch=false,ValidRing=true,ValidBranch=true; for(int
    i=0;i<res.Length;i++)
    {
        if(char.IsDigit(res[i])) isRing=true;
        if(res[i]=='(' || res[i]==')') isBranch=true;
    }

    if(isRing) ValidRing = checkRingValidity(res); if(isBranch)
    ValidBranch = checkBranchValidity(res);

    return (ValidRing && ValidBranch && (checkElementValidity(res)));
}

public bool checkRingValidity(char[] res)
{
    int count=0;
    for(int i=0;i<res.Length;i++) if(char.IsDigit(res[i])) count++;return
    count%2==0;
}
```

```
    public bool checkBranchValidity(char[] res)
    {
        int count=0;
        for(int i=0;i<res.Length;i++)
        {
            if(res[i]=='(') count++;
            if(res[i]==')') count--;
        }
        return count==0;
    }

    public bool checkElementValidity(char[] res)
    {
        for(int i=0;i<res.Length-1;i++)
        {
            if(Elements.ContainsKey(char.ToString(res[i])) ||
            Elements.ContainsKey(char.ToString(res[i])+char.ToString(res[i+1])) || bonds.ContainsKey(res[i]))
            continue;
            else if(char.IsLower(res[i]) || res[i] =='(' || res[i]==')' || char.IsDigit(res[i]))continue;
            else return false;
        }
        if(char.IsLower(res[res.Length-1]) || res[res.Length-1] =='(' || res[res.Length-1]==')'
|| char.IsDigit(res[res.Length-1]) || Elements.ContainsKey(char.ToString(res[res.Length-1]))) return
true;
        else return false;
    }}
```

## Logic Class:

```
public static readonly Dictionary<char,int> bonds = new Dictionary<char, int>()
    {
        {'-',1},{'=',2},{'#',3},{'$', 4},{'.', 0},{':',1}
    };
    //CONSTRUCTOR
    public WeightedGraph()
    {
        //INITIALIZING adjacencyList AND HydrogenCount AS EMPTY LISTSthis.adjacencyList =
        new Dictionary<int , Dictionary<int, int>>(){}; this.HydrogenCount = new Dictionary<int,
        int>(){};
    }
    //VERTEX CREATION-->Add new vertexpublic
    void addVertex(int vertex)
    {
        if(!this.adjacencyList.ContainsKey(vertex))
        {
            this.adjacencyList.Add(vertex, new Dictionary<int, int>());
        }
        else
        {
            Console.WriteLine("this vertex is in use");
        }
    //CREATION OF EDGES BETWEEN NODES--> New edge between 2 vertices
    }
```

```
  public void addEdge(int v1, int v2, int weight)
{
    if (this.adjacencyList.ContainsKey(v1) && this.adjacencyList.ContainsKey(v2))
    {
        this.adjacencyList[v1].Add(v2, weight);
        this.adjacencyList[v2].Add(v1, weight);
    }
    else
    {
        Console.WriteLine(v1);
        Console.WriteLine(v2);
        Console.WriteLine("Error: Vertex does not exist");
    }
}

//nearestIndex-> when given a index it finds the nearest element index on the leftand right side the given index
public int nearestIndex(int idx,int side)
{
    if(side==1)
    {
        for(int i=idx-1;i>=0;i--)
        {
            if(ElementIndexs[i]==1)
            {
                idx=i;
```

```
                break;
            }
        }
    }
    if(side==0)
    {
        for(int i=idx+1;i<ElementIndexs.Length;i++)
        {
            if(ElementIndexs[i]==1)
            {
                idx=i;
                break;
            }
        }
    }
    return idx;
}
//implicitHydrogenIndex--> CALCULATES THE IMPLICIT HYDROGENCOUNT OF EACH ELEMENT AND ADDS IT TO THE HydrogenCount DICTIONARY
public void implicitHydrogenIndex(string input)
{
    char[] res = input.ToCharArray();int
    valency=0;
    int bondsCount=0;int
    element= 0;
    string firstLetter = ""; string
    secondLetter = "";bool isit
```

```
=false;
foreach (var x in this.adjacencyList)
{
    valency=0;
    bondsCount=0;
    firstLetter = char.ToString(res[x.Key]);
    secondLetter = "";isit
    = false;
    if(x.Key<res.Length-1)
    {
        secondLetter = char.ToString(res[x.Key+1]);
    }
    string newelement = string.Join("",firstLetter,secondLetter);
    if(Elements.ContainsKey(newelement))
    {
        isit=true;
    }
    if(isit==false)
    {
        element=Elements[char.ToString(res[x.Key])];
    }
    else
    {
        element=Elements[newelement];
    }
    foreach(KeyValuePair<int, int> y in x.Value)
    {
```

```
            bondsCount+=y.Value;
        }
        valency=element-bondsCount;
        this.HydrogenCount.Add(x.Key,valency);
    }
    foreach(KeyValuePair<int, int> ele in this.HydrogenCount)
    {
        Console.WriteLine(ele.Key +"-->"+ ele.Value);
    }
}

// adjacencyMatrix --> GENERATES THE ADJACENCY MATRIXpublic int[,]
adjacencyMatrix(string input)
{
    //size --> NO OF ELEMENTS IN THE GIVEN INPUT SMILES NOTATION
    //Matrix_size --> LENGTH OF THE GIVEN INPUT SMILES NOTATION
    char[] res = input.ToCharArray();
    int sum=0,n=0,m=0,size = elementCount,Matrix_size = res.Length;int[,] array =
    new int[Matrix_size,Matrix_size];
    int[] check = new int[Matrix_size];
    int[,] adjacencyMatrix = new int[size,size];

    //GENERATE THE BIG MATRIX
    foreach(var x in this.adjacencyList)
    {
        foreach(KeyValuePair<int, int> y in x.Value)
```

```
    {
        array[x.Key,y.Key] = 1;
    }
}

//LOGIC TO REDUCE THE BIG MATRIX TO SMALLER ONE
for(int i=0;i<Matrix_size;i++)
{
    sum=0;
    for(int j=0;j<Matrix_size;j++)
    {
        sum+=array[i,j];
    }
```