DSA PART I

END SEM PROJECT

HEADING

Predicting Inhibition Values of Anti-Biofilm Agents Using Regression Models and One-Hot Encodings: A Data-Driven Approach

PROBLEM STATEMENT

The process of identifying and characterizing potential anti-biofilm agents for combating pathogenic infections is laborious and resource-intensive. This research aims to alleviate these challenges by proposing a novel methodology that utilizes one-hot encodings of the Simplified Molecular Input Line Entry System (SMILES) notation to predict inhibition values of anti-biofilm agents. By employing advanced regression models such as linear regression, random forest, and decision trees, the project seeks to develop a web application that automates the conversion of molecular formulas to one-hot encodings. This innovative solution will significantly streamline the drug discovery process, reducing the burden and time required for biotechnologists to evaluate potential drug candidates. Accurate prediction of inhibition values using this approach has the potential to expedite the discovery of novel and efficacious drugs, offering invaluable assistance to biologists and researchers in their endeavors to combat biofilm-related infections. By leveraging these advanced computational techniques, this project aims to revolutionize the field of drug discovery and contribute to the development of new therapeutic options.

METHODOLOGY

In this project, various data structures and libraries were employed to handle and analyze the data effectively. Arrays (lists) were used to store the float inhibition values of anti-biofilm agents, providing a convenient structure for numerical data storage. Strings were utilized for storing names, SMILES, and other relevant information. Hashtables (dictionaries) were used to keep track of SMILES and their corresponding encoded values efficiently.

The project made use of several libraries to facilitate different tasks. Matplotlib was employed for data visualization and generating plots. Scikit-learn enabled the implementation of regression models such as linear regression, random forest, and decision trees for predicting inhibition values. Seaborn complemented Matplotlib by enhancing the visual presentation of data. The regex library facilitated web scraping and extraction of necessary data. CSV library aided in file manipulation, allowing seamless import and export of data in CSV format.

By leveraging these data structures and libraries, this project achieved efficient data handling, analysis, visualization, and prediction, contributing to the advancement of anti-biofilm agent discovery and drug development.

**Advantages:**

Arrays (lists) provide a straightforward and flexible structure for storing and manipulating float inhibition values, enabling efficient data management.

Strings allow for easy storage and retrieval of names, SMILES, and other textual information, ensuring convenient data handling.

Hashtables (dictionaries) provide fast access to encoded values based on SMILES, ensuring efficient tracking and retrieval of data.

Utilizing these data structures promotes organized and structured data representation, facilitating seamless analysis and interpretation of the information.

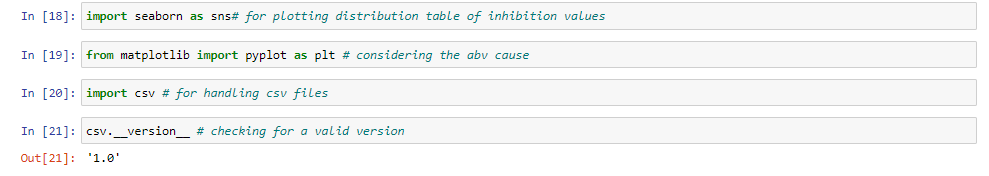
**Disadvantages:**

Arrays have a fixed size and may require resizing if the data exceeds the initial capacity, potentially leading to memory inefficiencies.

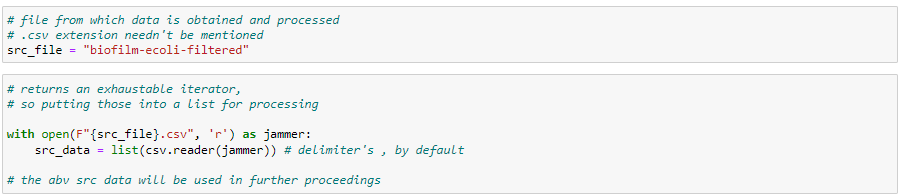
String operations can be computationally expensive, especially when dealing with large datasets, impacting overall performance.

Hashtables may consume additional memory due to the overhead associated with maintaining key-value pairs, potentially increasing memory usage.

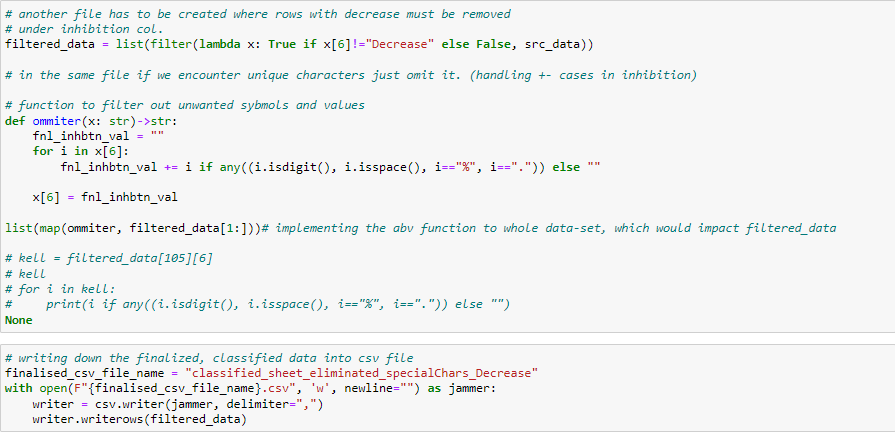
Incorrect usage or improper implementation of these data structures may result in data inconsistency or errors in the analysis process.

RESULT

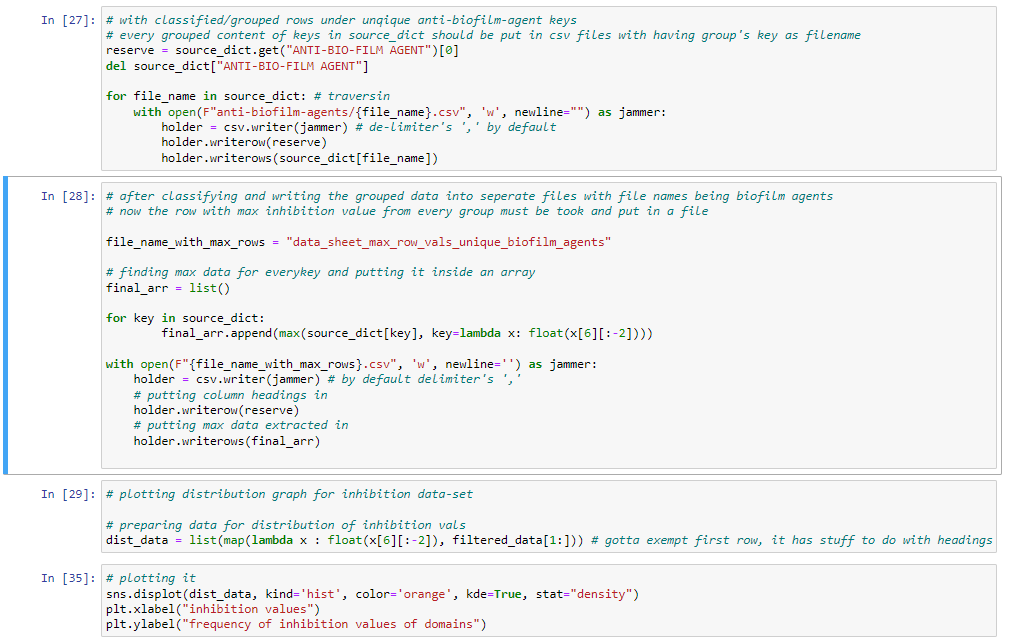
1. Imports necessary for processing obtained data, consisting of smiles corresponding to their compound names and inhibition values.

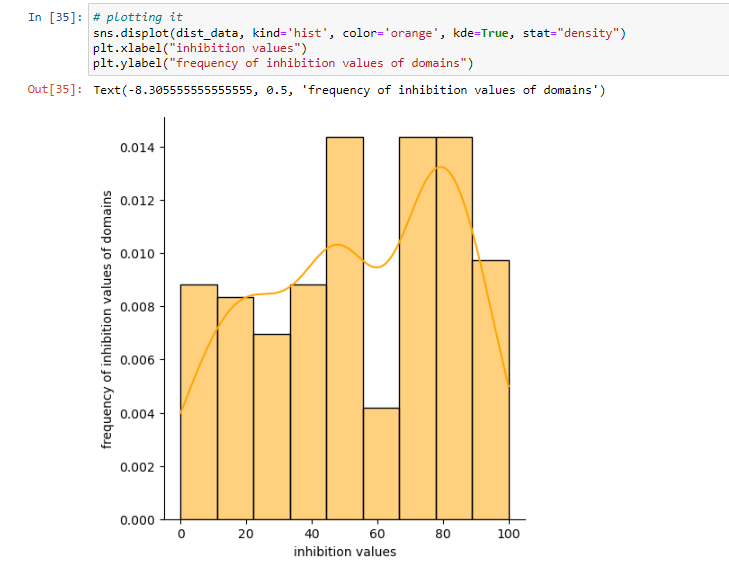


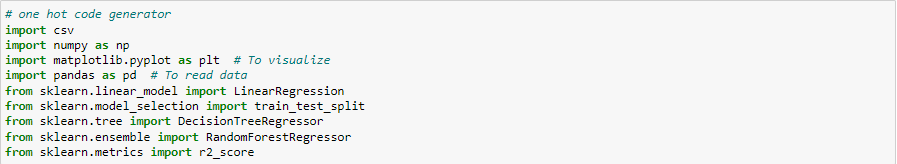
1. Scr\_file variable would hold file to be read, the library csv’s used to read data and retrieve a list of rows from csv file specified.



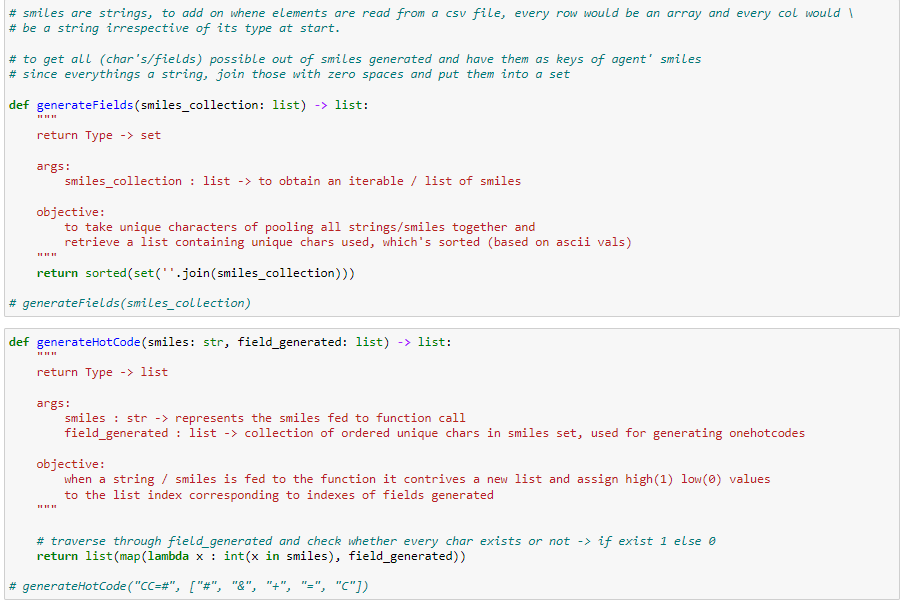
1. Data read must be filtered, talking inhibition values… chars like (%, +- and .) are evicted.



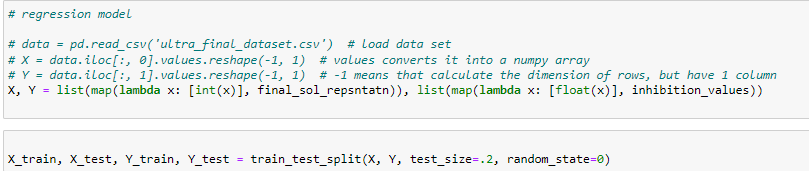
1. processed data’s then stored in multiple csv files beside their corresponding inhibition value. The one with redundancies would be housed under the same file.
2. Inhibition values collected are passed through seaborn method for a distribution plot.



1. In another kernel import necessary libraries for passing data through regression models.



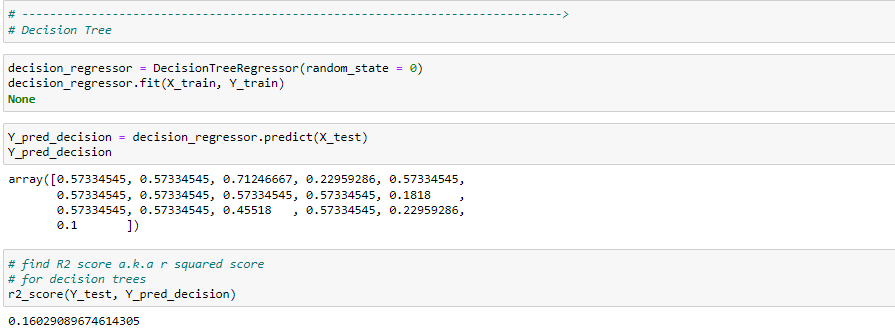
1. Functions listed above are used to generate fields of the given dataset, fields indicate the number of bits to be reserved for every smiles. GenerateCode would take in field as parameter and would convert smiles fed to its encoding format.



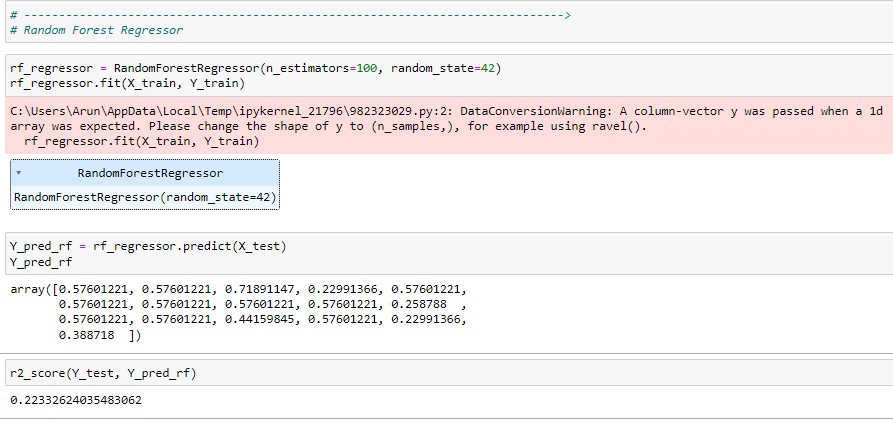
1. Data read from finalized csv file is chunked into X and Y. Where X is the independent variable and Y is the dependent variable. X and Y are grouped 80-20, 80 for training and 20 for testing.



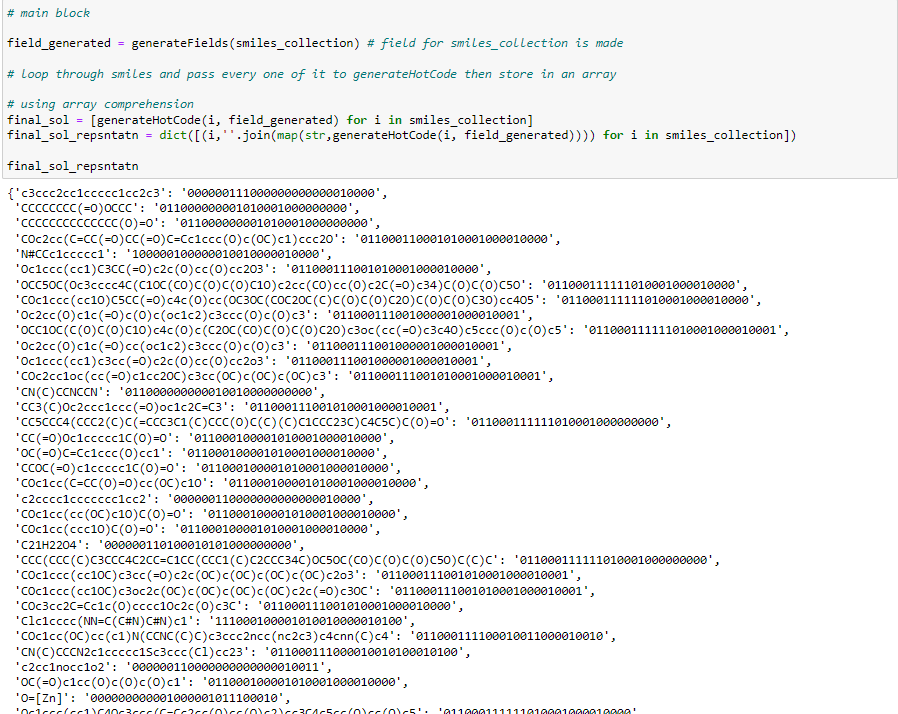
1. X and Y datasets which were chunked for training and testing were passed to **LinearRegression** model under SCIKIT LEARN. **R2** score was **0.102** or **10.2%** of accurate prediction.



1. X and Y datasets which were chunked for training and testing were passed to **DecisionTreeRegressor** model under SCIKIT LEARN. **R2** score was **0.160** or **16.02%** of accurate prediction. Slightly better than the previous model.



1. X and Y datasets which were chunked for training and testing were passed to **RandomForestRegressor** model under SCIKIT LEARN. **R2** score was **0.223** or **22.33%** of accurate prediction. Way better than couple of previous models.



1. Smiles along with One-Hot-Encodings generated are put in a Hashtable a.k.a Dictionaries in python. Where key’s smiles and value’s encoding generated.

Analysis and Discussion

1. Time Complexity :

1. Writing csv files and reading csv files – O(N) time complexity
2. Map function in python – O(N) time complexity [way faster than conventional loop though]
3. Filter function in python – O(N) time complexity [way faster than conventional loop though]
4. List comprehension in python – O(N) time complexity [easier way of putting elements via traversal in a list]
5. Ommiter user-defined – O(N) time complexity
6. Sea born Distribution plot – O(N Log N) time complexity
7. generateField user-defined –

1. O(M Log M) – time take to sort the final array

M is the number of unique characters in given dataset

1. O(N) for the join method used in joining elements in the array with an empty char ‘’.
2. generateHotCode user-defined –

O(M \* N) time complexity

Where M is time taken to search for character existence

N is time take to traverse and put elements into a list

1. Time taken by regression models
   1. Linear Regression Model – O(p^2 \* N)

p – number of features enlisted

N – length of dataset being fed

* 1. Decision Tree – O(Log(N))

N – length of data set, this one only covers time take to predict

* 1. RandomForestRegressor – O(E \* Log(N))

E – number of estimators

N – length of dataset

LIMITATIONS OF OUR PROGRAM

One limitation of our project is the reliance on one-hot encodings of smiles for predicting inhibition values. While one-hot encodings can capture the presence or absence of specific substructures, they may not fully capture the nuanced chemical information necessary for accurate predictions. This limitation arises from the loss of spatial and contextual information that is inherent in the original smiles representation. Consequently, the model's performance may be compromised, especially when dealing with complex chemical structures or subtle variations in molecular features. Additionally, the project's reliance on manual feature engineering and the absence of libraries like pandas and numpy may limit the scalability and efficiency of the implementation. Furthermore, the limited availability of data for training the regression models can adversely affect their predictive performance and generalizability.