Computational Drug Discovery

A project report submitted to
Dr. Babasaheb Ambedkar Technological University, Lonere
in partial fulfillment of the requirements for the award of the degree

Bachelor of Technology
in
Computer Engineering

Mini-Project – II
(BTCOM607)

by

Mr. Prajwal Rameshwar Gurnule (PRN: 2146491245011)
Mr. Pranav Vijay Ikhar (PRN: 2146491245009)
Miss Renuka Ashok Kothekar (PRN: 2146491245016)
Miss Ritika Vinayakrao Bhonge (PRN: 2146491245003)
(VI Semester)

under the guidance of

Mr. Amol Jumde
(Assistant Professor)

Department of Computer Engineering
Shiksha Mandal's
BAJAJ INSTITUTE OF TECHNOLOGY, WARDHA
Pipri, Arvi Road, Wardha - 442001.

(2023-24)

Dr. Babasaheb Ambedkar Technological University, Lonere
Bajaj Institute of Technology, Wardha
Pipri, Arvi Road, Wardha - 442001.

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

This is to certify that Mini-Project-II titled

Computational Drug Discovery

has been completed by

Mr. Prajwal Rameshwar Gurnule (PRN: 2146491245011)

Mr. Pranav Vijay Ikhar (PRN: 2146491245009)

Miss Renuka Ashok Kothekar (PRN: 2146491245016)

Miss Ritika Vinayakrao Bhonge (PRN: 2146491245003)

of VI Semester (Sec: A), Computer Engineering of academic year 2023-24 in partial fulfillment of Mini-Project-II (BTCOM607) course as prescribed by the Dr. Babasaheb Ambedkar Technological University, Lonere.

Mr. Amol Jumde
(Department of Computer Engineering)

Prof. Sheetal Kale	**Dr.N.M.Kanhe**
(Head of the Department)	Principal
	Bajaj Institute of Technology Wardha

Place: BIT, Wardha
Date: May 28, 2024

Declaration

We hereby declare that the Mini Project report titled **“Computational Drug Discovery”** submitted by us to the Bajaj Institute of Technology, Wardha, in partial fulfilment of the requirement for the award of Degree of B. Tech in Computer Engineering is a record of bonafide seminar work carried out by us under the guidance of Mr. Amol Jumde.

We, further declare that the work reported in this Mini Project report have not been submitted either in-part or in-full for the award of any other degree in any other Institute or University.

Report Title: Computational Drug Discovery

SN	**Student Name**	**PRN**	**Signature**
1	Prajwal Gurnule	2146491245011	
2	Pranav Ikhar	2146491245009	
3	Renuka Kothekar	2146491245016	
4	Ritika Bhonge	2146491245003	

Date: May 28, 2024
Place: BIT, Wardha

Acknowledgment

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report. Special thanks is due to our Guide Mr. Amol Jumde, Assistant Professor of Computer Engineering whose help, stimulating suggestions and encouragement helped us in all time of process and in writing this report.

We extend our heartfelt thanks to Prof. Sheetal Kale, Head of the Department Computer Engineering Bajaj Institute of Technology, Wardha for her insightful guidance, assistance and invaluable suggestions that significantly enriched the quality of these work. We also sincerely thanks for the time spent profreading and correcting our mistakes.

Our deepest thanks to Department of Computer Engineering And Dr. Narendra Kanhe, Principal Bajaj of Institute of Technology, Wardha. We would also like to acknowledge with much appreciation the crucial role of the staff in Computer Lab, who gave us a permission to use the Computers in the laboratory. Many thanks go to the all lecturers and go to all our classmates, especially to our friends for spending their time in helping and giving support whenever we need it in our Mini Project Topic.

Abstract

The process of drug discovery is a complex and multifaceted endeavor, traditionally characterized by high costs and extensive timelines. To address these challenges, our project leverages computational methodologies to streamline the identification and evaluation of potential drug candidates. This project is divided into two primary objectives: predicting potential drug candidates based on their molecular properties and predicting drug-target binding affinity.

The first objective focuses on the computational prediction of promising drug candidates through the analysis of molecular structures. We calculate molecular descriptors, relevant features, and predicted drug to identify compounds with desirable pharmacological profiles.We also mention and follow universal Food and Drug Administration approved rules those are Lipinski Rule and IC50 unit values for evaluation and shortlisting the applicable drug. This approach allows for the efficient screening of large chemical libraries, prioritizing molecules that exhibit potential efficacy and safety.

The second objective involves predicting the binding affinity between drug candidates and their target proteins. By employing molecular docking techniques and advanced machine learning models, we simulate the interactions between drugs and their targets, predicting the strength and nature of these interactions. These predictions are subsequently validated and optimized through experimental methods, ensuring accuracy and reliability.

By integrating these computational approaches, our project aims to enhance the efficiency and effectiveness of the drug discovery process. The combined methodologies enable the prioritization of compounds with both favorable molecular properties and strong target binding affinities, thereby accelerating the development of effective therapeutics. This computational strategy represents a significant advancement in the field of drug discovery, reducing both the time and cost associated with bringing new drugs to market.

Keywords- ***Mask Language Models***, ***Drug Target Affinity***, ***Multilayer Perceptron***, ***Embedding***, ***Exploratory Data Analysis***

Abbreviations

DTA	Drug-target affinity
MLP	Multilayer Perceptron
MLM	Mask Language Models
SVM	Support Vector Machine
GNN	Graph neural network
SMILES	Simplifed molecular input line entry system
BERT	Bidirectional encoder representations from transformers
GPT	Generative Pre-trained Transformer
NLP	Natural language processing
XGBoost	EXtreme Gradient Boosting
MSE	Mean square error
MAE	Mean absolute error
SGT	Sequence Graph Transform
TAPE	Tasks Assessing Protein Embeddings
RF	Random Forest
QSAR	Quantitative Structure-Activity Relationship

Contents

List of Figures

Chapter 1

Introduction

Drug discovery is an expensive process in both time and cost with a daunting attrition rate. As revealed by a recent study, the average cost of developing a new drug was 1 billion dollars and has been ever increasing. In the past decade, the practice of drug discovery has been undergoing radical transformations in light of the advancements in artificial intelligence, which, at its core, is molecular representation learning. Molecules are typically represented in three ways: fixed representations, including fingerprints and structural keys, that signify the presence of specific structural patterns; linear notations, such as Simplified Molecular Input Line Entry System (SMILES) strings; and molecular graphs. With the advent of deep learning, various neural networks have been proposed for molecular representation learning, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs) and graph neural networks (GNNs), among others. One major task for AI&ML in drug discovery is molecular property prediction, which seeks to learn a function that maps a structure to a property value.

Drug–target affinity (DTA) prediction is an important step in virtual screening, which can quickly match target and drug and speed up the process of drug development. DTA prediction provides information about the binding strength of drugs to target proteins, which can be used to show whether small molecules can bind to proteins. For proteins with known structure and site information, we can use molecular simulation and molecular docking to carry out detailed simulations, thus get more accurate results, which is called structure-based virtual screening.

In our drug discovery project, we divide tasks into two main categories: predicting potential drug candidates based on their molecular properties and predicting drug-target binding affinity. The first involves calculating molecular descriptors, engineering features, building predictive models, and identifying promising molecules. The second focuses on simulating interactions between drugs and targets using molecular docking and advanced machine learning models to predict binding affinities, followed by experimental validation and optimization. Combining these approaches streamlines the identification and evaluation of effective drug candidates, enhancing the efficiency of the drug discovery process.

1.1 Motivation

The motivation behind this drug discovery project is to enhance the efficiency and effectiveness of identifying new therapeutic compounds. Traditional drug discovery methods are often time-consuming, costly, and prone to high failure rates. By leveraging computational techniques to predict drug properties and their binding affinities to biological targets, we aim to streamline the initial screening process and focus on the most promising candidates. This approach not only reduces the time and cost involved in drug development but also increases the likelihood of discovering effective and safe drugs. Ultimately, our goal is to accelerate the development of new treatments for various diseases, improving patient outcomes and addressing unmet medical needs.

1.2 Benefits of the computational drug discovery

In these section, we elaborate on the Benifits of the computational drug discovery. There are a number of benefits of the computational drug discovery. These benefits include:

- **Increased Efficiency:** Computational methods significantly speed up the initial screening of potential drug candidates, reducing the overall drug discovery timeline. By identifying promising compounds early, the project also minimizes resources spent on testing unlikely candidates, thereby lowering costs.
- **Higher Success Rates:** Using predictive models allows the project to focus on compounds with a higher likelihood of success, thus increasing the chances of discovering effective and safe drugs. Additionally, by filtering out less promising candidates early, the project reduces the high failure rate typically associated with traditional drug discovery methods.
- **Potential for Innovation:** Computational tools can identify novel targets and previously unexplored molecular interactions, opening new avenues for drug development. This approach also facilitates the development of personalized medicine, tailoring drug discovery efforts to specific molecular profiles for more effective treatments.

1.3 Challenges Faced by the Project

In these section, we elaborate on the challenges faced in the computational drug discovery. These challenges include:

- **Data Quality and Availability:** High-quality, comprehensive datasets are crucial for building accurate predictive models. Incomplete or biased data can lead to unreliable predictions, and integrating data from various sources (e.g., biological, chemical, clinical) in a coherent manner presents additional challenges.
- **Model Accuracy and Validation:** Even advanced models may struggle with accurately predicting complex biological interactions, leading to false positives or negatives. Extensive experimental validation is necessary to confirm computational predictions, which can be resource-intensive.

- **Computational Complexity:** Sophisticated algorithms and large datasets require significant computational power and resources. Developing and refining these algorithms to accurately predict drug-target interactions is complex and demands specialized expertise.

1.4 Problem statement

The problem statement for the drug discovery project involves two primary objectives:
Predicting Drug Based on Molecular Properties: This entails developing methods to effectively analyze molecular structures, calculate molecular descriptors, and engineer features that characterize potential drug molecules. The goal is to build predictive models capable of identifying promising drug candidates from a pool of compounds. These models should accurately assess the likelihood of a molecule possessing desired pharmacological properties, such as efficacy and safety profiles.
Predicting Drug-Target Binding Affinity: This objective focuses on simulating the interactions between drug molecules and their target proteins. The aim is to predict the binding affinity, or the strength of interaction, between a drug and its intended target. This involves employing molecular docking techniques and advanced machine learning algorithms to simulate and predict how well a drug molecule binds to its target protein. Experimental validation and optimization are then conducted to verify and refine these predictions.
By combining these approaches, the project aims to streamline the process of identifying and evaluating effective drug candidates. This integration of molecular property prediction and binding affinity prediction enhances the efficiency of the drug discovery process by enabling researchers to prioritize compounds with both favorable molecular properties and strong binding interactions with their target proteins.

1.5 Objectives

The objectives of the drug discovery project divided into the two main categories:
1. Predicting Potential Drug Candidates Based on Molecular Properties and Lipinski's Rule
Molecular Descriptor Calculation:

- Develop or integrate computational tools to calculate various molecular descriptors (e.g., molecular weight, logP, hydrogen bond donors and acceptors).
- Ensure the descriptors cover a wide range of molecular properties relevant to drug-likeness and bioactivity.

Feature Engineering:

- Identify and construct relevant features from the calculated molecular descriptors.
- Employ techniques like dimensionality reduction and feature selection to optimize the feature set for predictive modeling.

Predictive Model Building:

- Develop machine learning models (e.g., regression, classification, ensemble methods) to predict the drug-likeness of molecules.
- Train and validate these models using datasets of known drug-like and non-drug-like compounds.

Lipinski's Rule Compliance:

- Implement checks to ensure potential drug candidates comply with Lipinski's Rule of Five.
- Filter and prioritize molecules that meet these criteria for further evaluation.

Identification of Promising Molecules:

- Screen large chemical libraries to identify molecules with favorable molecular properties.
- Use predictive models to rank and shortlist top candidates for subsequent experimental validation.

2. Predicting Drug-Target Binding Affinity
Molecular Docking Simulation:

- Develop or use existing molecular docking tools to simulate the interaction between drug candidates and target proteins.
- Optimize docking protocols to accurately predict binding poses and interaction energies.

Advanced Machine Learning Models:

- Build and train advanced machine learning models (e.g., deep learning, graph-based neural networks) to predict binding affinities.
- Incorporate features such as molecular interaction fields, protein-ligand contact maps, and structural descriptors.

Binding Affinity Prediction:

- Validate the accuracy of binding affinity predictions using benchmark datasets and cross-validation techniques.
- Integrate predictions from docking simulations and machine learning models to refine binding affinity estimates.

Experimental Validation:

- Select model validation as Mean-Squared error for the regression model also applying different forms of clustering .
- Design and conduct experiments to validate the predicted binding affinities and biological activity of top candidates.

By achieving these objectives, the project aims to create a robust and efficient pipeline for drug discovery, combining computational and experimental approaches to identify and develop new therapeutic agents.

1.6 Scope of work

This project aims to develop a computational framework to streamline drug discovery by:

1. Predicting Drug Based on Molecular Properties:

- Develop algorithms to analyze the 3D structure of candidate drug molecules.
- Implement methods to calculate relevant molecular descriptors, which are numerical representations capturing key features like size, shape, and charge distribution.
- Engineer additional features that characterize ideal drug candidates, such as solubility, permeability, and potential toxicity.

2. Predicting Drug-Target Binding Affinity:

- Develop computational models to simulate the interactions between drug molecules and their target proteins.
- Employ advanced machine learning algorithms to predict the binding affinity (strength of interaction) between a drug and its target protein based on the docked complexes.
- Refine the computational models based on the experimental data to improve prediction accuracy.

1.7 Organization of the Report

In these section, The organization of our report is elaborated. this is organized into six chapters. which are explained below:

- Chapter - 1 describes the introduction of our project, the motivation, objectives, problem statement, scope of the work, and the work flow diagram of our project Computational Drug Discovery.
- Chapter - 2 describes the Literature Survey of the project. Initially the literature review which include the related work of the project. furthermore, the identification of gaps in literature and brief summaries of the research paper are elaborated.
- Chapter - 3 descibes the methodology of the project which include the sections as proposed solution, methods and algorithms, Embedding techniques and the system architecture.
- Chapter - 4 provides the implementation of our project which include data processing, model building, model implementation, clustering which are carried out.
- Chapter - 5 shows the results of our drug discovery project which consist of the evaluation measures, comparison between the models, prediction of drug and the deployment.
- Chapter - 6 describes the Conclusion and Future Scope of our Computational Drug Discovery Project.

1.8 Computational Drug Discovery Work Flow

In these chapter, we elaborate on th work flow diagram of our Computational Drug Discovery Project.
In the realm of computational drug discovery, machine learning (ML) significantly enhances the efficiency and accuracy of identifying potential new drug candidates. In Figure 1.1, the work flow begins with the acquisition of diverse molecular data, sourced from humans, animals, and plants. This raw data, encompassing various biological and chemical characteristics, forms the foundation for subsequent analysis. Advanced embedding techniques, leveraging models such as BERT, GPT, and other transformers, alongside specific embeddings like SGT Embeds, Elmo, Encoder, and Tape Embeds, are employed to process and convert this molecular data into structured representations that are amenable to machine learning algorithms. Once embedded, this processed data is utilized to train predictive models that can infer molecular properties, a crucial step in understanding the potential efficacy and safety of molecules. The trained models are then applied to a range of downstream tasks. Notably, they predict drug target affinity, determining the likelihood of a molecule binding effectively to a specific target, which is essential for efficacy. Additionally, they assess drug target interactions at the atomic level, providing insights into the molecular mechanisms of interaction. By automating these intricate analyses, computational drug discovery accelerates the identification of promising drug candidates, reducing the time and cost traditionally associated with drug development. This integrated approach harnesses the power of ML to streamline the complex landscape of drug discovery, ultimately aiming to deliver new therapeutic solutions more rapidly and efficiently.

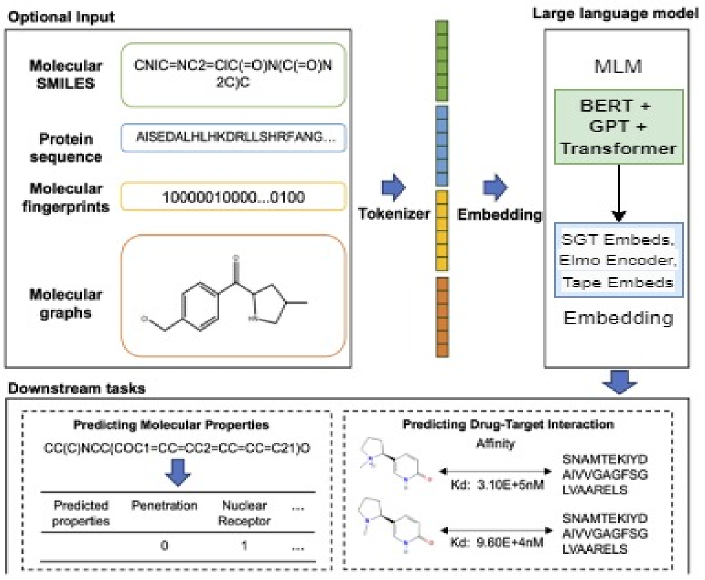


Figure 1.1: Computational Drug Discovery Experimental Workflow

In Figure 1.1, the workflow provide the flow for our drug discovery project which describe all the process, phases and the parameter which will carry while building these project.

Chapter 2

Literature Survey

This chapter outline the literature Survey for our Computational Drug Discovery, encomposing the theory and reserach carried for understanding the concepts. The chapter begins with the literature review which provide the related work, followed by the gap identified in the research paper and the brief summaries of papers.
This Chapter provides a comprehensive understanding of the concepts and theories for the drug discovery.

2.1 Literature review

This section discusses a few literature review for the Computational Drug Discovery in which tools and techniques are elaborated related to drug study.
According to (Jiajia Liu et al. 2024) present a summary of the prominent large language models used in natural language processing, such as BERT and GPT, and focus on exploring the applications of large language models at different omics levels in bioinformatics, mainly including applications of large language models in genomics, transcriptomics, proteomics, drug discovery and single cell analysis[1].
(Mingjian Jiang et al. 2022) Sequence-based drug target afnity prediction can predict the afnity according to protein sequence, which is fast and can be applied to large datasets. However, due to the lack of protein structure information, the accuracy needs to be improved.The proposed model which is called WGNN-DTA can be competent in drug-target afnity (DTA) and compound-protein interaction (CPI) prediction tasks. Various experiments are designed to verify the performance of the proposed method in diferent scenarios, which proves that WGNN-DTA has the advantages of simplicity and high accuracy[2].
(Leiming Xia et al. 2023) proposed a DTA prediction model using the deep learning method is proposed, which uses an undirected-CMPNN for molecular embedding and combines CPCProt and MLM models for protein embedding. An attention mechanism is introduced to discover the important part of the protein sequence. The proposed method is evaluated on the datasets Ki and Davis, and the model outperformed other deep learning methods[3].
(Mingjian Jiang et al. 2020) propose a method called DGraphDTA for DTA prediction. Two graphs are built—one for drug molecules and another for proteins. The structural information of molecules and proteins is utilized to construct these graphs. Graph neural networks (GNNs) are introduced to obtain representations for both drug molecules and proteins[4].

(Thin Nguyen et al. 2021) propose a new model called GraphDTA that represents drugs as graphs and uses graph neural networks to predict drug-target affinity. We show that graph neural networks not only predict drug-target affinity better than non-deep learning models, but also outperform competing deep learning methods. Our results confirm that deep learning models are appropriate for drug-target binding affinity prediction, and that representing drugs as graphs can lead to further improvements[5].
(Maha A Thafar et al. 2022) conducted extensive experiments to evaluate and demonstrate the robustness and efficiency of the proposed method on benchmark datasets used in state-of-the-art non-structured-based drug-target binding affinity studies. Affinity2Vec showed superior and competitive results compared to the state-of-the-art methods based on several evaluation metrics, including mean squared error, concordance index, and area under the precision-recall curve[6].

2.2 Gap identification in the literature

Identifying the gap in the literature survey between the two research papers Large language models in bioinformatics: applications and perspectives by Jiajia Liu and Sequence-based drug-target affinity prediction using weighted graph neural networks by Mingjian Jiang and Drug-target binding affinity prediction using message passing neural network and self supervised learning by Leiming Xia.

- The first research paper, 'Large language models in bioinformatics: applications and perspectives' seems like an outlier compared to the other two. It focuses on large language models (LLMs) in general bioinformatics applications, whereas the other two papers specifically target drug-target affinity prediction using deep learning techniques.

- 'Sequence-based drug-target affinity prediction using weighted graph neural networks' and 'Drug-target binding afnity prediction using message passing neural network and self supervised learning' both concentrate on predicting drug-target affinity using specific deep learning architectures (graph neural networks).

- While both papers focus on deep learning for drug-target affinity prediction, there might be a gap in:
  1. There's no mention of incorporating large language models (like the one discussed in the first paper) into the deep learning architectures for drug-target affinity prediction.
  2. There could be a gap in how the papers address the limitations of current data representation methods for molecules and proteins.

This review highlights a potential gap in bioinformatics research. Existing studies explored either LLMs for general bioinformatics tasks or deep learning techniques for drug-target affinity prediction. The opportunity lies in combining these approaches. Future research could investigate incorporating LLMs into deep learning models for improved drug-target affinity prediction. Additionally, there's potential to address limitations in how molecules and proteins are represented in these models, leading to further performance gains.

2.3 Summary of Literature Survey

In these section, we elaborate the summary of the Literature Survey -
1. Large language models in bioinformatics: applications and perspectives
The paper highlights the applications of LLMs across various areas of bioinformatics, including genomics, transcriptomics, proteomics, drug discovery, and single-cell analysis. For instance, LLMs can be used to analyze vast genomic datasets, identify patterns, and predict gene function. Similarly, they can aid in deciphering complex interactions between genes and proteins, ultimately accelerating drug discovery processes. The authors emphasize that LLMs have the potential to extend far beyond their natural language processing strengths. By integrating them with domain-specific tools and databases, researchers can leverage LLMs to unlock new avenues in bioinformatics research.
2. Sequence-based drug-target affinity prediction using weighted graph neural networks
This research addresses this limitation by incorporating graph neural networks (GNNs) into the prediction process. The authors construct graphs representing both the drug molecule (based on SMILES) and the protein (based on its amino acid sequence). By incorporating detailed contact information within the protein graph, the GNN can extract features that enhance the prediction of binding affinity between the drug and the target protein. The proposed WGNN-DTA method offers advantages in terms of both simplicity and high accuracy. Additionally, it bypasses complex steps like multiple sequence alignment (MSA), making it particularly suitable for screening large databases of drugs and targets.
3. Drug-target binding afnity prediction using message passing neural network and self supervised learning
This paper introduces a novel deep learning model that tackles this challenge. The model leverages message passing neural networks (MPNNs) to analyze the structures of both the drug molecule and the protein, considering them as interconnected graphs. This allows the model to capture the intricate relationships between different parts of the molecule and protein. Furthermore, the model incorporates self-supervised learning, a technique where the model learns from auxiliary tasks related to the structures, even without explicitly labeled DTA data. The study demonstrates that this model surpasses existing deep learning methods in DTA prediction accuracy, offering a powerful new strategy for virtual screening in drug discovery.

Chapter 3

Methodology

This chapter discuss the proposed solution for our Computational drug discovery model. This section serves as a roadmap, outlining the steps, and techniques used to collect and analyze the ML model. It also Outline the Models and Algorithms and followed by the System Architecture of our model .

3.1 Proposed Solution

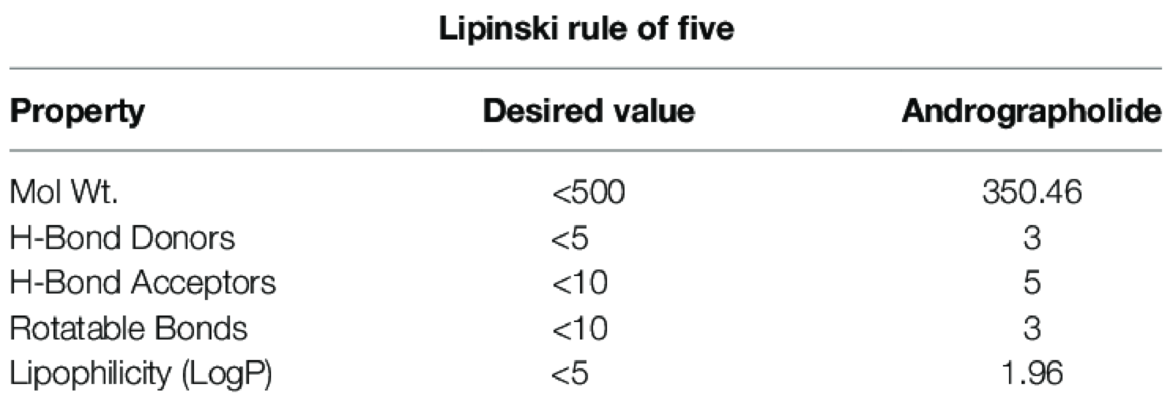
In these section, we elaborate on the proposed solution of our computational drug discovery project. these section divide the proposed solution as depending on the task to perform.

3.1.1 Predicting Potential Drug Candidates Based on Molecular Properties and Lipinski's Rule

Initially, we utilize databases like ChEMBL, PubChem, and DrugBank to gather comprehensive data on molecular properties, bioactivity, and known drug candidates. Ensure data quality by filtering out incomplete or erroneous entries. So we are chooseing ChEMBL as it serves as a valuable tool for researchers in drug discovery, medicinal chemistry, and other fields related to pharmacology and biomedicine. Calculate a variety of molecular descriptors for each compound, such as molecular weight, hydrogen bond donors (HBD), hydrogen bond acceptors (HBA), octanol-water partition coefficient (LogP), and others. Utilize tools like RDKit for descriptor calculation.
Apply Lipinski's Rule of Five criteria to the dataset. The rules state that in general, an orally active drug has no more than one violation of the following criteria:

1. Molecular weight $\leq$ 500 Dalton.
2. LogP ≤ 5
3. Hydrogen bond donors (HBD) ≤ 5
4. Hydrogen bond acceptors (HBA) ≤ 10

Filter out compounds that do not meet these criteria or that violate more than one of these rules. Maintain a list of compounds that strictly adhere to Lipinski's criteria for further analysis. as explained in Figure 3.1 for the drug andrographolide.



Lipinski rule of five

Property	Desired value	Andrographolide
Mol Wt.	<500	350.46
H-Bond Donors	<5	3
H-Bond Acceptors	<10	5
Rotatable Bonds	<10	3
Lipophilicity (LogP)	<5	1.96

Figure 3.1: Drug-likeness (Lipinski rule of five) for andrographolide.

Develop machine learning models to predict drug-likeness and bioactivity. Techniques like Random Forest, Support Vector Machines (SVM), and neural networks can be employed. Train models using labeled datasets (e.g., known drugs vs. non-drugs) to identify patterns and predictive features.
Construct Quantitative Structure-Activity Relationship (QSAR) models to relate molecular properties to biological activity.Use regression models to predict the activity of new compounds. Based on experimental data, iteratively optimize lead compounds to enhance efficacy, reduce toxicity, and improve pharmacokinetic properties. Employ medicinal chemistry strategies for structural modifications.
This comprehensive approach integrates data mining, computational modeling, and experimental validation to predict potential drug candidates effectively. By leveraging molecular properties and adhering to Lipinski's Rule of Five, the method ensures the identification of compounds with favorable drug-like characteristics, thereby enhancing the efficiency and success rate of drug discovery efforts.

3.1.2 Predicting Drug-Target Binding Affinity

SMILES is a notation that allows a user to represent a chemical structure in a way that can be easily used by computers.As depicted in Figure 3.2, SMILES is a string of characters where each character or set of characters represents a specific atom or bond. Example: The given SMILES string 'CC(=O)NCCC(S(=O)(=O)O)' represents the molecular structure of a specific compound.

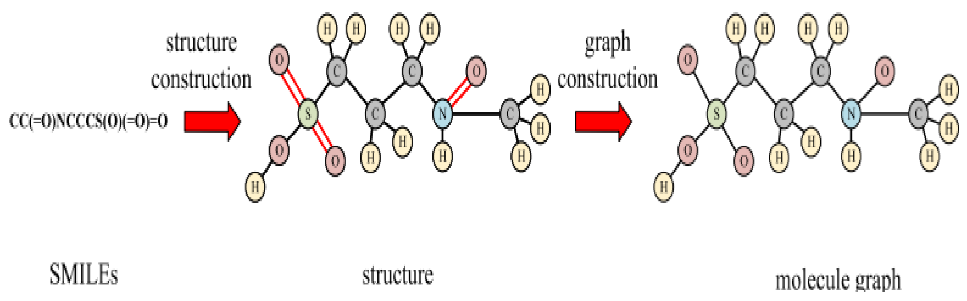


Figure 3.2: Construction of molecular graph from SMILES

Converting the SMILES string into a two-dimensional (2D) chemical structure. The chemical structure visually represents the arrangement of atoms and the bonds between

them. 2D chemical structure is further converted into a molecular graph. This is a more abstract representation where the molecule is represented as a graph.
The protein sequence, which is a linear string of amino acids. Each letter in the sequence represents a specific amino acid. Example: The sequence shown is 'EGLPQEAS', where each letter corresponds to an amino acid (e.g., E for Glutamic acid, G for Glycine, L for Leucine, etc.).

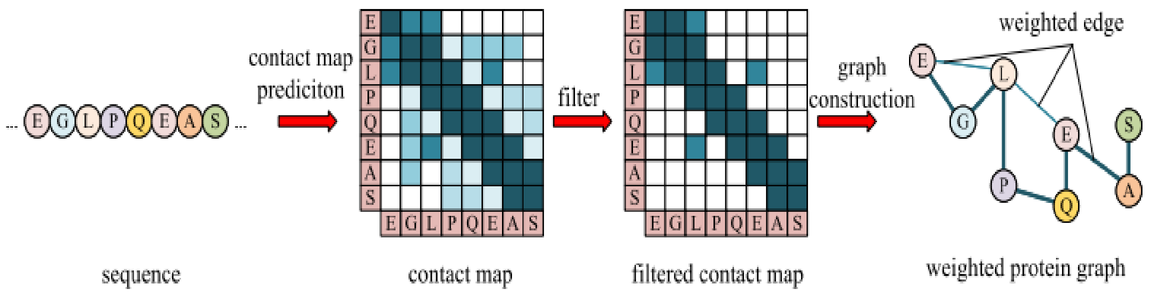


Figure 3.3: Construction of weighted protein graph from protein sequence

In Figure 3.3, A contact map is a matrix that represents the spatial proximity of amino acids in the protein. Each cell in the matrix indicates whether two amino acids are in contact within a specified distance threshold. The raw contact map is often filtered to remove noise and focus on the most significant contacts. Convert the filtered contact map into a graph representation where nodes represent amino acids and edges represent contacts between them.
GNNs are employed to process the graphs generated from both the protein and the molecule.
WGNN (Weighted Graph Neural Network): Specialized for processing the protein weighted graph.
GNN (Graph Neural Network): Used for processing the molecule graph.

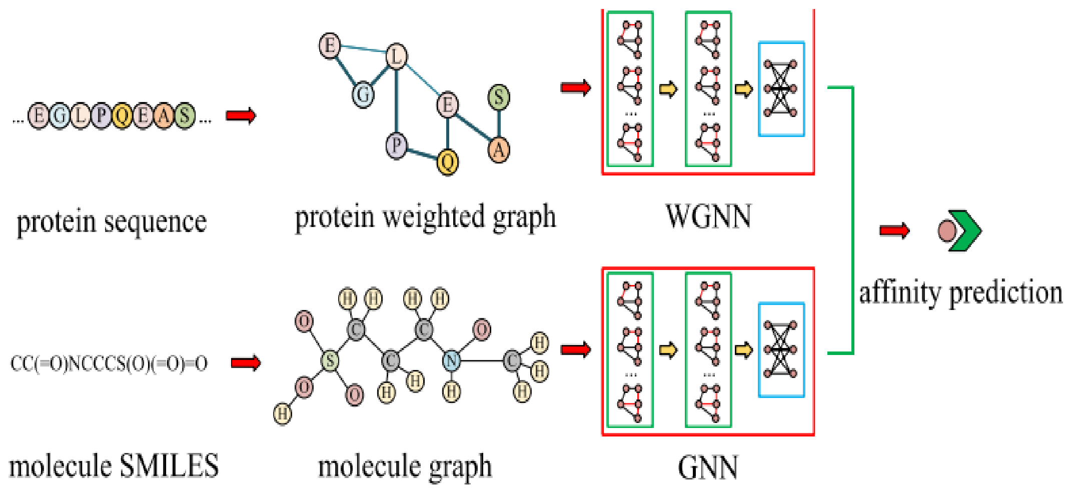


Figure 3.4: Construction of molecular graph from SMILES

The features learned by the WGNN and GNN are combined to form a joint representation.In Figure 3.4, The combined representation integrates the information from both the protein and the drug molecule, capturing how the two interact at a detailed level. A final model uses this combined representation to predict the binding affinity

3.2 Models and Algorithms

In these section, we elaborate on the Model and Algorithms that are applicable in our our computational Drug Discovery Project.

3.2.1 Lipinski's rule

Lipinski's Rule of Five is a key concept in drug discovery that predicts whether a biologically active molecule is likely to have the necessary properties for oral bioavailability. The rule evaluates pharmacokinetic properties such as absorption, distribution, metabolism, and excretion (ADME) based on specific criteria: no more than 5 hydrogen bond donors, no more than 10 hydrogen bond acceptors, a molecular mass less than 500 Daltons (Da), and a partition coefficient (logP) not greater than 5. In the context of "Computational Drug Discovery using Molecular Properties and Binding Affinity Predictions," Lipinski's Rule of Five is crucial for the initial screening of drug candidates.

3.2.2 Regression model

A regression model in machine learning is used to predict continuous output values based on input data. It estimates the relationships between a dependent variable (target) and one or more independent variables (features). Common types include linear regression, which fits a straight line to the data, and polynomial regression, which fits a curve. Ridge and Lasso regression add regularization to prevent overfitting. In computational drug discovery, regression models predict properties like drug binding affinities and solubility, helping to identify potential drug candidates.

3.2.3 MLP (Multi-Layer Perceptron) regressor

An MLP (Multi-Layer Perceptron) regressor is a type of neural network used for regression tasks, which predicts continuous output values. It consists of multiple layers of interconnected neurons that learn to map input features to target outputs through a process called backpropagation. The MLP regressor is used to predict complex relationships between molecular properties and outcomes like drug binding affinities or pharmacokinetic properties. Its ability to model non-linear relationships makes it particularly useful for accurately forecasting the efficacy and behavior of potential drug candidates.

3.2.4 Random Forest

The Random Forest algorithm is an ensemble learning method used for classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputting either the mode of the classes (classification) or the mean prediction (regression) of the individual trees. Each tree is built from a random subset of the data and features, which helps to improve the model's accuracy and reduce overfitting. In "Computational Drug Discovery" Random Forest can be utilized to predict various drug-related properties such as binding affinities, solubility, and other pharmacokinetic characteristics. By analyzing complex and high-dimensional datasets, Random Forest helps in identifying significant molecular descriptors and their interactions, ultimately aiding in the selection and optimization of potential drug candidates.

3.2.5 Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a supervised machine learning algorithm commonly used for classification and regression tasks. It works by finding the optimal hyperplane that best separates the data into different classes or, in the case of regression, fits the data to predict continuous values. SVMs are known for their effectiveness in high-dimensional spaces and their ability to handle non-linear relationships through the use of kernel functions. SVMs are employed to predict outcomes such as drug binding affinities and classify molecules based on their potential efficacy. By analyzing molecular descriptors and binding data, SVMs can help identify and prioritize drug candidates with desirable properties, thereby enhancing the efficiency of the drug discovery process.

3.3 Embedding Techniques

In these section, we will elaborate on the Embedding Technique that are applied on our machine language model to extract the feature outoff protein sequence.

3.3.1 SGT embeddings

Sequence Graph Transform (SGT), a feature embedding function, that can extract a varying amount of short- to long-term dependencies without increasing the computation is proposed. SGT features yield significantly superior results in sequence clustering and classification with higher accuracy and lower computation as compared to the existing methods, including the state-of-the-art sequence/string Kernels and LSTM.It can extract useful information from amino acid sequences passed into it. You can find more information about it in here arXiv:1608.03533.
Using it is very easy as we can simply use the library associated with it.

3.3.2 Elmo encoder

ELMo is an NLP framework developed by AllenNLP. ELMo word vectors are calculated using a two-layer bidirectional language model (biLM). Each layer comprises forward and backward pass.lmo embeddings are learned from the internal state of a bidirectional LSTM and represent contextual features of the input text. It's been shown to outperform previously existing pre-trained word embeddings like word2vec and glove on a wide variety of NLP tasks. Some of those tasks are Question Answering, Named Entity Extraction and Sentiment Analysis.You can find more information about it in here Modeling aspects of the language of life through transfer-learning protein sequences.

3.3.3 Tape embeds

Task assessing protein embeds is based of on the paper that is available at arxiv1906.08230. On their github repor which you can find [here], there are two pre trained models, one on bert-base (Transformer model) and the other on babbler-1900 (UniRep model).

3.4 System Architecture

In this section, we will elaborate on the key design ideas and concepts regarding the architecture of the proposed Computational Drug Discovery. As depicted in Figure 3.5

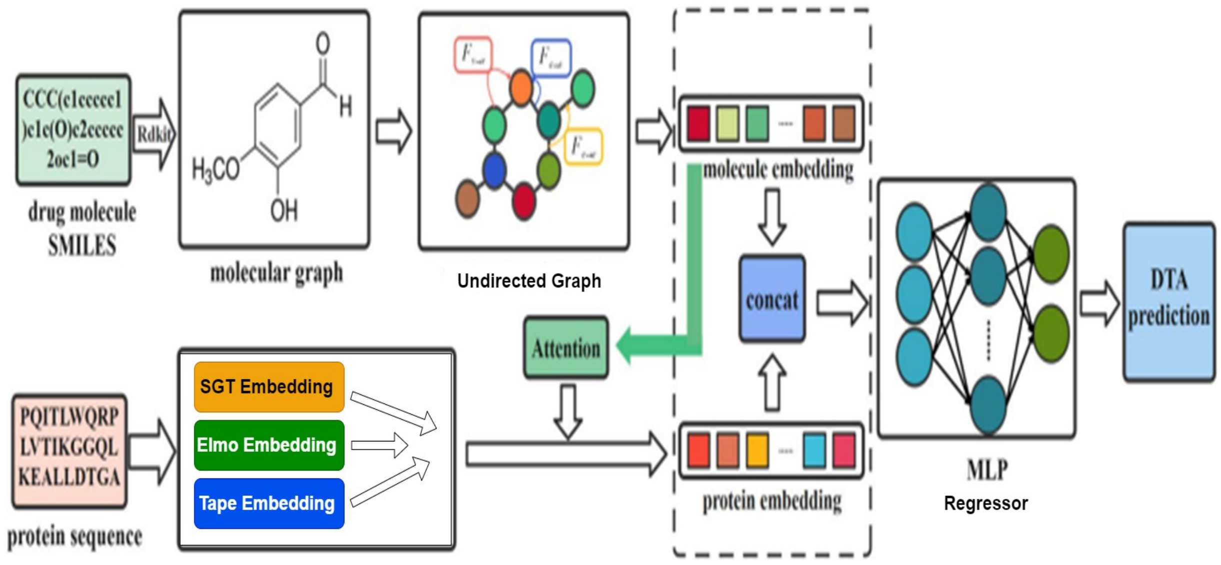


Figure 3.5: System Architecture for the Drug Target Binding affinity prediction

The proposed architecture process is shown in Fig. 3.5, The input of the molecule is in Simplifed Molecular Input Line Entry System (SMILES) format, which is converted to the graph structure, and the undirected graph. is used to update the information of atoms and bonds in the molecular graph to obtain the representation of the entire molecule. The protein sequence, which is a linear string of amino acids. Each letter in the sequence represents a specific amino acid.Now, these sequence is passed to the embeding we have three types of the embedding technique SGT(sequence graph transform),Elmo(developed by Allen NLP) and Tape(Task assessing protein embeds). Now simentenously, as the undirected graph of the Smile molecule and the protein sequence will proced for the embedding in these phase the selected attribute from each is concatenated and unlabelled data. these data gets evaluated with various model, outof which MLP regresser is most prominant one to predict more accurate scores. Considering whether the model can learn the important part of the protein for binding affinity, an attention mechanism is introduced. Finally, representations of the protein and molecule are concatenated and fed into the MLP to predict the binding afnity.

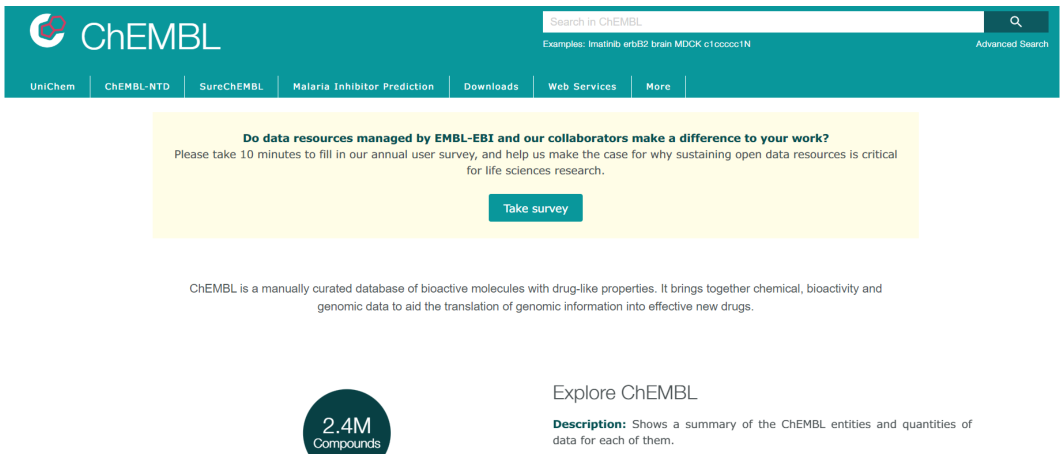
Chapter 4

Implementation

This chapter outline the implementation detail of our Computational Drug Discovery, encomposing the technology used and the Machine Learning Model. The chapter begins by introducing the chosen Dataset followed by the Data pre-processing and the Feature Extraction.
This Chapter provides a comprehensive understanding of the technical aspects behind our model.

4.1 Dataset

1. Predicting Potential Drug Candidates Based on Molecular Properties and Lipinski's Rule
Initially, we Install the ChEMBL web service package as depicted in Figure 4.1 so that we can retrieve bioactivity data from the ChEMBL Database. ChEMBL Database is a database that contains curated bioactivity data of more than 2 million compounds. It is compiled from more than 76,000 documents, 1.2 million assays and the data spans 13,000 targets and 1,800 cells and 33,000 indications.
ChEMBL or ChEMBLdb is a manually curated chemical database of bioactive molecules with drug inducing properties. It is maintained by the European Bioinformatics Institute (EBI), of the European Molecular Biology Laboratory (EMBL), based at the Wellcome Trust Genome Campus, Hinxton, UK.



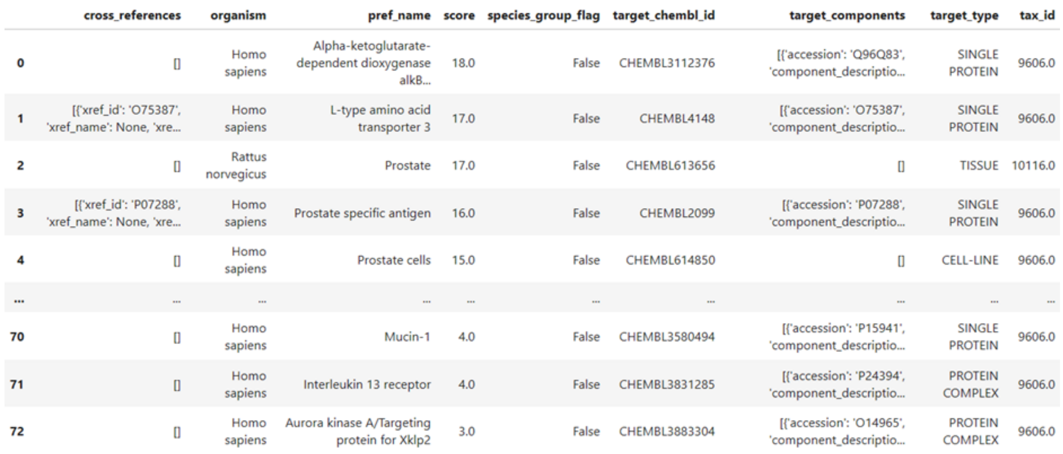
Do data resources managed by EMBL-EBI and our collaborators make a difference to your work?
Please take 10 minutes to fill in our annual user survey, and help us make the case for why sustaining open data resources is critical for life sciences research.

ChEMBL is a manually curated database of bioactive molecules with drug-like properties. It brings together chemical, bioactivity and genomic data to aid the translation of genomic information into effective new drugs.

Explore ChEMBL

Description: Shows a summary of the ChEMBL entities and quantities of data for each of them.

Figure 4.1: CHEMBL Website Database

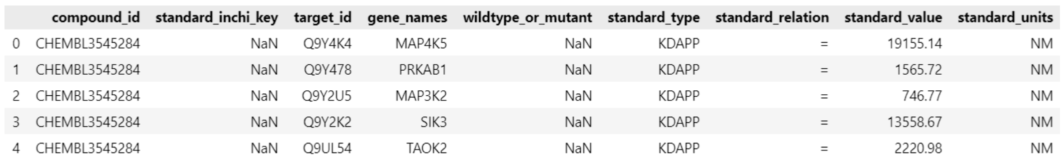


	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Homo sapiens	Alpha-ketoglutarate-dependent dioxygenase alkB...	18.0	False	CHEMBL3112376	[{'accession': 'Q96Q83', 'component_descriptio...	SINGLE PROTEIN	9606.0
1	[{'xref_id': 'O75387', 'xref_name': None, 'xre...	Homo sapiens	L-type amino acid transporter 3	17.0	False	CHEMBL4148	[{'accession': 'O75387', 'component_descriptio...	SINGLE PROTEIN	9606.0
2	[]	Rattus norvegicus	Prostate	17.0	False	CHEMBL613656	[]	TISSUE	10116.0
3	[{'xref_id': 'P07288', 'xref_name': None, 'xre...	Homo sapiens	Prostate specific antigen	16.0	False	CHEMBL2099	[{'accession': 'P07288', 'component_descriptio...	SINGLE PROTEIN	9606.0
4	[]	Homo sapiens	Prostate cells	15.0	False	CHEMBL614850	[]	CELL-LINE	9606.0
...	...	...	...	...	...	...	...	...	...
70	[]	Homo sapiens	Mucin-1	4.0	False	CHEMBL3580494	[{'accession': 'P15941', 'component_descriptio...	SINGLE PROTEIN	9606.0
71	[]	Homo sapiens	Interleukin 13 receptor	4.0	False	CHEMBL3831285	[{'accession': 'P24394', 'component_descriptio...	PROTEIN COMPLEX	9606.0
72	[]	Homo sapiens	Aurora kinase A/Targeting protein for Xklp2	3.0	False	CHEMBL3883304	[{'accession': 'O14965', 'component_descriptio...	PROTEIN COMPLEX	9606.0

Figure 4.2: ChEMBL Dataset

cross-references: This column likely contains identifiers for the chemical compound in other databases.
organism: This column specifies the organism the compound targets.
pref-name: This column contains the preferred name of the chemical compound.
score: This column contains a score, possibly indicating the relevance of the compound to a specific target.
species: This column provides more specific information about the organism the compound targets.
group-flag: This column likely indicates whether the target is a group of related molecules or a single molecule.
target-chembl-id: This column contains an identifier for the target molecule in the ChEMBL database, a database of bioactive molecules.
target-components: This column lists the components of the target molecule.
target-type: This column indicates the type of the target molecule.
tax-id: This column contains the taxonomic ID of the organism the compound targets.

2. Predicting Drug-Target Binding Affinity
The data utilized in this task was generated by the IDG-Kinase group with funding from the NIH Common Fund. It can be orginally accessed through a synapse account from the website here DTC Dataset as depicted in Figure 4.3.
compound_id: This column contains a unique identifier for each compound in the dataset.
standard_inchi_key: This column contains a unique identifier for the chemical structure of each compound using the InChIKey standard. InChI stands for International Chemical Identifier.
target_id: This column contains an identifier for the biological target of each compound.
gene_names: This column contains a list of gene names associated with the target of each compound.
wildtype_or_mutant: This column indicates whether the target is a wildtype (normal) protein or a mutant protein.
standard_type: This column describes the type of standard measurement used in the dataset. In this case, all entries are "KDapp", which likely refers to the dissociation



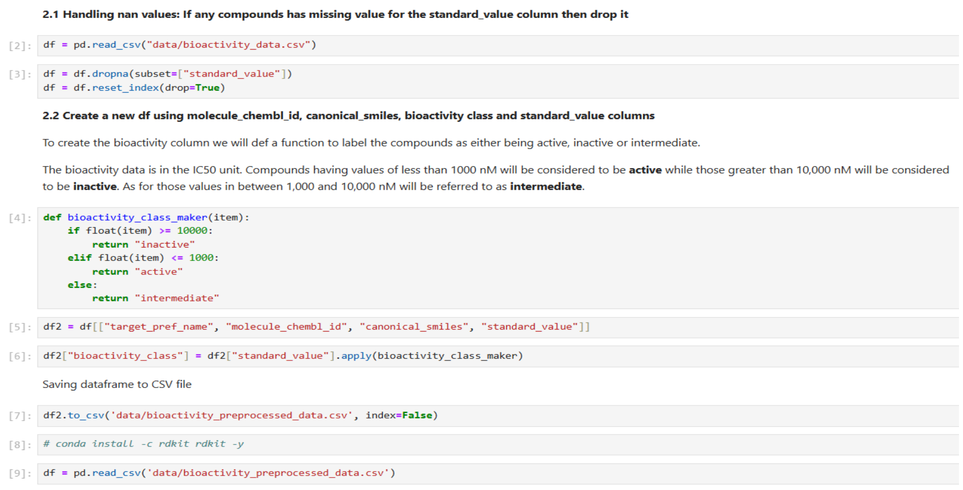
	compound_id	standard_inchi_key	target_id	gene_names	wildtype_or_mutant	standard_type	standard_relation	standard_value	standard_units
0	CHEMBL3545284	NaN	Q9Y4K4	MAP4K5	NaN	KDAPP	=	19155.14	NM
1	CHEMBL3545284	NaN	Q9Y478	PRKAB1	NaN	KDAPP	=	1565.72	NM
2	CHEMBL3545284	NaN	Q9Y2U5	MAP3K2	NaN	KDAPP	=	746.77	NM
3	CHEMBL3545284	NaN	Q9Y2K2	SIK3	NaN	KDAPP	=	13558.67	NM
4	CHEMBL3545284	NaN	Q9UL54	TAOK2	NaN	KDAPP	=	2220.98	NM

Figure 4.3: DTC dataset

constant (Kd) apparent affinity.
standard_relation: This column describes the relationship between the standard value and the target molecule. In this case, all entries are "=" indicating a measured value.
standard_value: This column contains the actual measured value, presumably the dissociation constant (Kd) for the interaction between the compound and its target.
standard_units: This column specifies the units of the standard value. In this case, all entries are "nM" for nanomolar.

4.2 Data Cleaning and Processing

Dataset preprocessing is a crucial step in developing a robust machine learning model. For the Computational Drug Discovery, it involves cleaning, normalizing, and transforming the data to ensure its accuracy and suitability for machine learning algorithms.
1. Predicting Potential Drug Based on Molecular Properties and Lipinski's Rule
Here we removes any rows where the standard_value column, likely indicating potency, is missing. In Figure 4.4, The code creates a new dataframe containing only specific columns relevant for bioactivity analysis. These columns include molecule ID, chemical structure identifier, bioactivity class, and the standard value (potency). The code defines a function to categorize the compounds in the dataframe into three classes based on their standard values (potency). Compounds with a potency value less than 1000 nM are considered active, those greater than 10,000 nM are inactive, and those in between are intermediate. The code saves the cleaned and processed dataframe as a new CSV file.



2.1 Handling nan values: If any compounds has missing value for the standard_value column then drop it

```
[2]: df = pd.read_csv("data/bioactivity_data.csv")
```

```
[3]: df = df.dropna(subset=["standard_value"])
     df = df.reset_index(drop=True)
```

2.2 Create a new df using molecule_chembl_id, canonical_smiles, bioactivity class and standard_value columns

To create the bioactivity column we will def a function to label the compounds as either being active, inactive or intermediate.

The bioactivity data is in the IC50 unit. Compounds having values of less than 1000 nM will be considered to be **active** while those greater than 10,000 nM will be considered to be **inactive**. As for those values in between 1,000 and 10,000 nM will be referred to as **intermediate**.

```
[4]: def bioactivity_class_maker(item):
         if float(item) >= 10000:
             return "inactive"
         elif float(item) <= 1000:
             return "active"
         else:
             return "intermediate"
```

```
[5]: df2 = df[["target_pref_name", "molecule_chembl_id", "canonical_smiles", "standard_value"]]
```

```
[6]: df2["bioactivity_class"] = df2["standard_value"].apply(bioactivity_class_maker)
```

Saving dataframe to CSV file

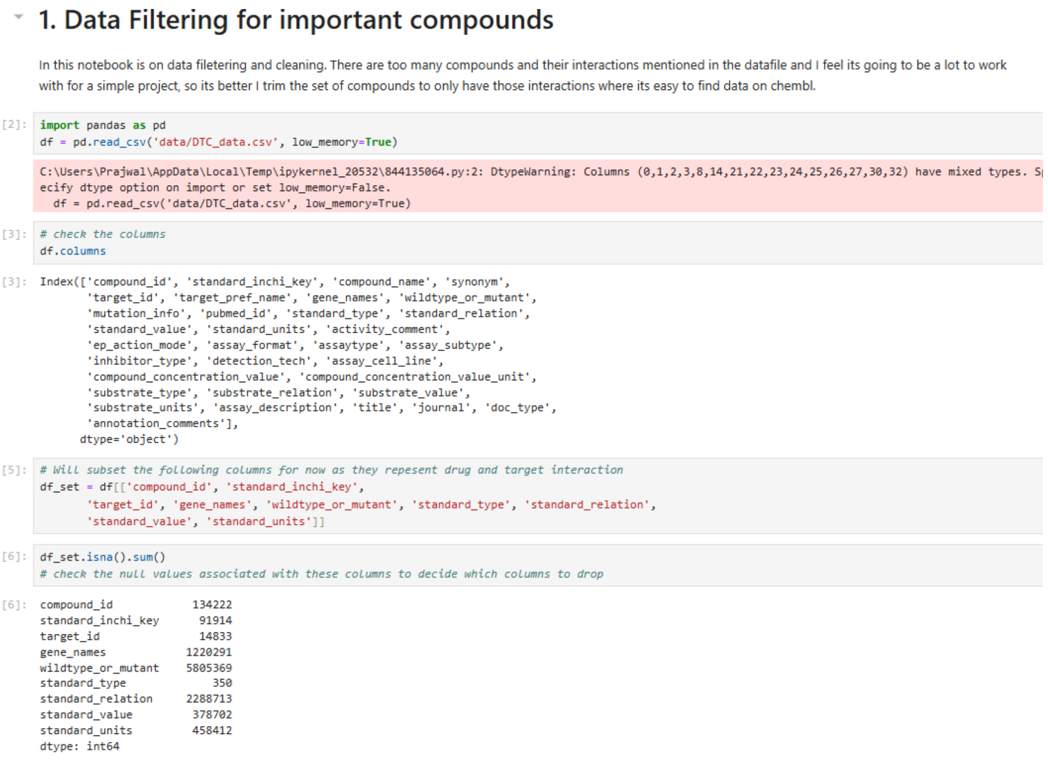
```
[7]: df2.to_csv('data/bioactivity_preprocessed_data.csv', index=False)
```

```
[8]: # conda install -c rdkit rdkit -y
```

```
[9]: df = pd.read_csv('data/bioactivity_preprocessed_data.csv')
```

Figure 4.4: Data cleaning and processing for important columns implementation

2. Predicting Drug-Target Binding Affinity
The script acknowledges that the original dataset contains information on many compounds and interactions, but only a limited subset is needed for the current project. The goal is to focus on compounds where data can be easily found in the ChEMBL database, likely a resource for bioactive molecules.The code uses Python libraries (pandas specifically) to read the data from a CSV file (DTC_data.csv). The code checks for potential data type inconsistencies within the dataframe. This is important for data manipulation and analysis. The code creates a new dataframe (df_set) containing only a specific set of columns deemed relevant for the analysis. These columns include compound ID, standard InChI key (unique identifier for chemical structure), target ID, gene names associated with the target, mutation status of the target (wildtype or mutant), type of standard measurement, relationship between the standard value and the target, standard value (likely potency), and standard unit. As depicted in Figure 4.5, The code checks for missing values (null values) within the newly created dataframe (df_set). This helps identify any potential data quality issues.


1. Data Filtering for important compounds

In this notebook is on data filetering and cleaning. There are too many compounds and their interactions mentioned in the datafile and I feel its going to be a lot to work with for a simple project, so its better I trim the set of compounds to only have those interactions where its easy to find data on chembl.

```
[2]: import pandas as pd
     df = pd.read_csv('data/DTC_data.csv', low_memory=True)

     C:\Users\Prajwal\AppData\Local\Temp\ipykernel_20532\844135064.py:2: DtypeWarning: Columns (0,1,2,3,8,14,21,22,23,24,25,26,27,30,32) have mixed types. Sp
     ecify dtype option on import or set low_memory=False.
       df = pd.read_csv('data/DTC_data.csv', low_memory=True)

[3]: # check the columns
     df.columns

[3]: Index(['compound_id', 'standard_inchi_key', 'compound_name', 'synonym',
            'target_id', 'target_pref_name', 'gene_names', 'wildtype_or_mutant',
            'mutation_info', 'pubmed_id', 'standard_type', 'standard_relation',
            'standard_value', 'standard_units', 'activity_comment',
            'ep_action_mode', 'assay_format', 'assaytype', 'assay_subtype',
            'inhibitor_type', 'detection_tech', 'assay_cell_line',
            'compound_concentration_value', 'compound_concentration_value_unit',
            'substrate_type', 'substrate_relation', 'substrate_value',
            'substrate_units', 'assay_description', 'title', 'journal', 'doc_type',
            'annotation_comments'],
           dtype='object')

[5]: # Will subset the following columns for now as they repesent drug and target interaction
     df_set = df[['compound_id', 'standard_inchi_key',
            'target_id', 'gene_names', 'wildtype_or_mutant', 'standard_type', 'standard_relation',
            'standard_value', 'standard_units']]

[6]: df_set.isna().sum()
     # check the null values associated with these columns to decide which columns to drop

[6]: compound_id             134222
     standard_inchi_key       91914
     target_id                14833
     gene_names             1220291
     wildtype_or_mutant     5805369
     standard_type              350
     standard_relation      2288713
     standard_value          378702
     standard_units          458412
     dtype: int64
```

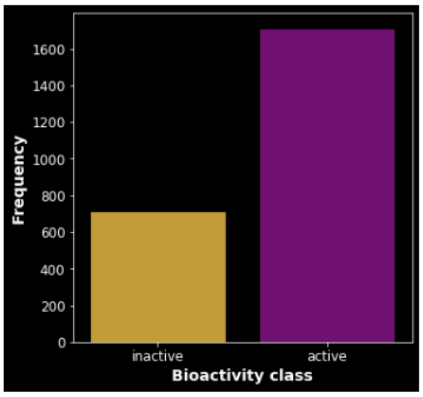
Figure 4.5: Data filtering implementation

4.3 Exploratory Data Analysis

Exploitative data analysis (EDA) refers to practices that manipulate data to mislead viewers or reach a predetermined conclusion, rather than uncovering the truth within the data. So, in our project we will extracting the active and inactive element from the dataset and plot a graph of solubility over molecular weight with the r2 score. and the active element which lie between the r2 score are shortlisted and saved to another csv file

3.1. Lets plot the frequency distribution of the two bioactivity classes

```
[30]: plt.figure(figsize=(5.5, 5.5))

      sns.countplot(x='bioactivity_class', data=df_EDA, palette = ["goldenrod", "purple"], edgecolor='black')

      plt.xlabel('Bioactivity class', fontsize=14, fontweight='bold')
      plt.ylabel('Frequency', fontsize=14, fontweight='bold')
      plt.xticks(fontsize=12)
      plt.yticks(fontsize=12)
      #plt.savefig('figures/plot_bioactivity_class.png', dpi=300, bbox_inches="tight")
      plt.show()
```



3.2 Next lets do a scatter plot of the MW vs the solubility to see the relactionship between these two features

```
[31]: slope, intercept, r_value, p_value, std_err = stats.linregress(df_EDA['MW'],df_EDA['LogP'])

[32]: slope

[32]: 0.004310648135502124

[33]: intercept

[33]: 1.9029421901917214
```

```
[34]: r_value = round(r_value,2)

[35]: r_value

[35]: 0.25

[36]: p_value

[36]: 1.1698854733545527e-36

[37]: x = np.linspace(100,1700,100)
      y = slope*x+intercept # defining the equation to plot the trending line together with the scatterplot

[38]: plt.figure(figsize=(10, 10))

      sns.scatterplot(x='MW', y='LogP', data=df_EDA, hue='target_pref_name', size='standard_value_transformed', edgecolor='black', alpha=0.7)
      plt.plot(x, y,  '--', label='trending_line $R^2$ = {}'.format(r_value))
      plt.xlabel('MW', fontsize=14, fontweight='bold')
      plt.ylabel('LogP', fontsize=14, fontweight='bold')
      plt.legend(bbox_to_anchor=(1.05, 1),  loc=2, borderaxespad=0)
      plt.xticks(fontsize=12)
      plt.yticks(fontsize=12)
      plt.title("Solubility vs Molecular Weight of compounds \n targetting molecules associated to Prostate Cancer ", fontsize=14, fontweight='bold')
      # plt.savefig('figures/plot_MW_vs_LogP.png', dpi=300, bbox_inches="tight")
      plt.show()
```

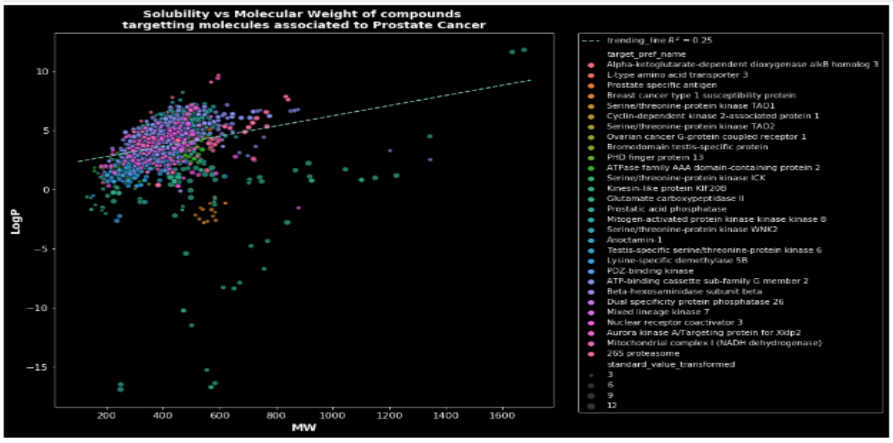
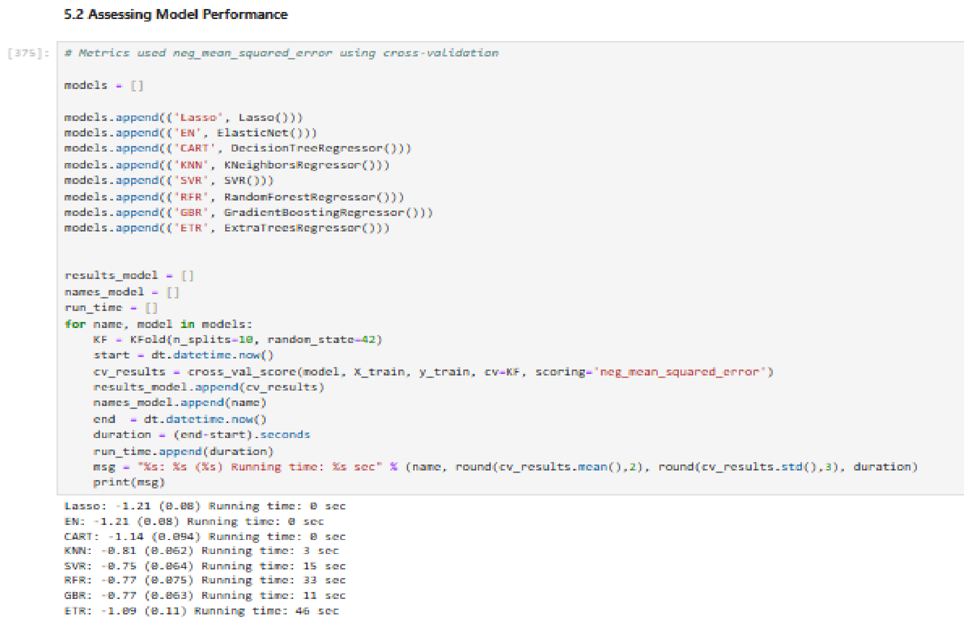


Figure 4.6: Exploratory Data analysis on Molecular data implementation

4.4 Model Building

In these section, we initially process Data split and applied different model algorithm based on the mean squared error using cross validation and simantanuosly we also find the running time for all algorithm. we implemented model building in below Figure 4.7



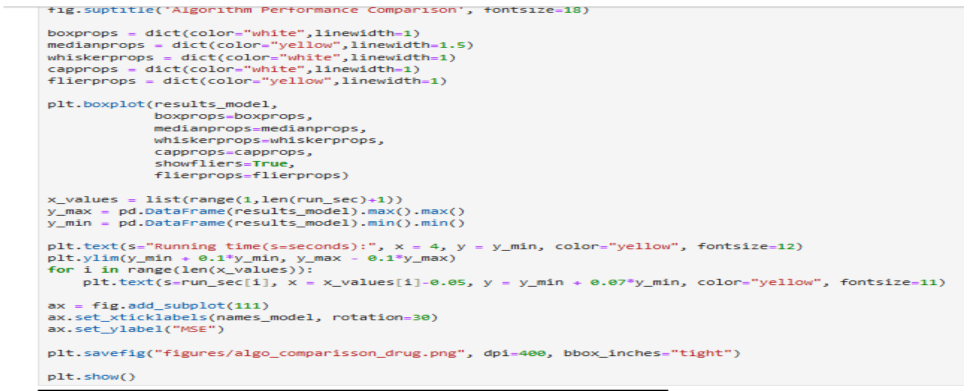
5.2 Assessing Model Performance

```
[375]: # Metrics used neg_mean_squared_error using cross-validation

models = []

models.append(('Lasso', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('SVR', SVR()))
models.append(('RFR', RandomForestRegressor()))
models.append(('GBR', GradientBoostingRegressor()))
models.append(('ETR', ExtraTreesRegressor()))


results_model = []
names_model = []
run_time = []
for name, model in models:
    KF = KFold(n_splits=10, random_state=42)
    start = dt.datetime.now()
    cv_results = cross_val_score(model, X_train, y_train, cv=KF, scoring='neg_mean_squared_error')
    results_model.append(cv_results)
    names_model.append(name)
    end  = dt.datetime.now()
    duration = (end-start).seconds
    run_time.append(duration)
    msg = "%s: %s (%s) Running time: %s sec" % (name, round(cv_results.mean(),2), round(cv_results.std(),3), duration)
    print(msg)
```

```
Lasso: -1.21 (0.08) Running time: 0 sec
EN: -1.21 (0.08) Running time: 0 sec
CART: -1.14 (0.094) Running time: 0 sec
KNN: -0.81 (0.062) Running time: 3 sec
SVR: -0.75 (0.064) Running time: 15 sec
RFR: -0.77 (0.075) Running time: 33 sec
GBR: -0.77 (0.063) Running time: 11 sec
ETR: -1.09 (0.11) Running time: 46 sec
```

Figure 4.7: Model building implementation

Now, we plot a candle chart visualization for checking which algorithm perform better performance based on Mean square error.it is revealed Support Vector Machine (SVM) and Random Forest Regressor (RFR) as the most promissing models. Lets fine tune them using GridSearchCV. Below Figure 4.8 show implementation of these.



```
fig.suptitle('Algorithm Performance Comparison', fontsize=18)

boxprops = dict(color="white",linewidth=1)
medianprops = dict(color="yellow",linewidth=1.5)
whiskerprops = dict(color="white",linewidth=1)
capprops = dict(color="white",linewidth=1)
flierprops = dict(color="yellow",linewidth=1)

plt.boxplot(results_model,
            boxprops=boxprops,
            medianprops=medianprops,
            whiskerprops=whiskerprops,
            capprops=capprops,
            showfliers=True,
            flierprops=flierprops)

x_values = list(range(1,len(run_sec)+1))
y_max = pd.DataFrame(results_model).max().max()
y_min = pd.DataFrame(results_model).min().min()

plt.text(s="Running time(s=seconds):", x = 4, y = y_min, color="yellow", fontsize=12)
plt.ylim(y_min + 0.1*y_min, y_max - 0.1*y_max)
for i in range(len(x_values)):
    plt.text(s=run_sec[i], x = x_values[i]-0.05, y = y_min + 0.07*y_min, color="yellow", fontsize=11)

ax = fig.add_subplot(111)
ax.set_xticklabels(names_model, rotation=30)
ax.set_ylabel("MSE")

plt.savefig("figures/algo_comparisson_drug.png", dpi=400, bbox_inches="tight")

plt.show()
```

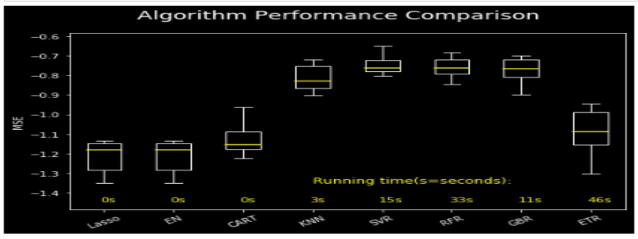


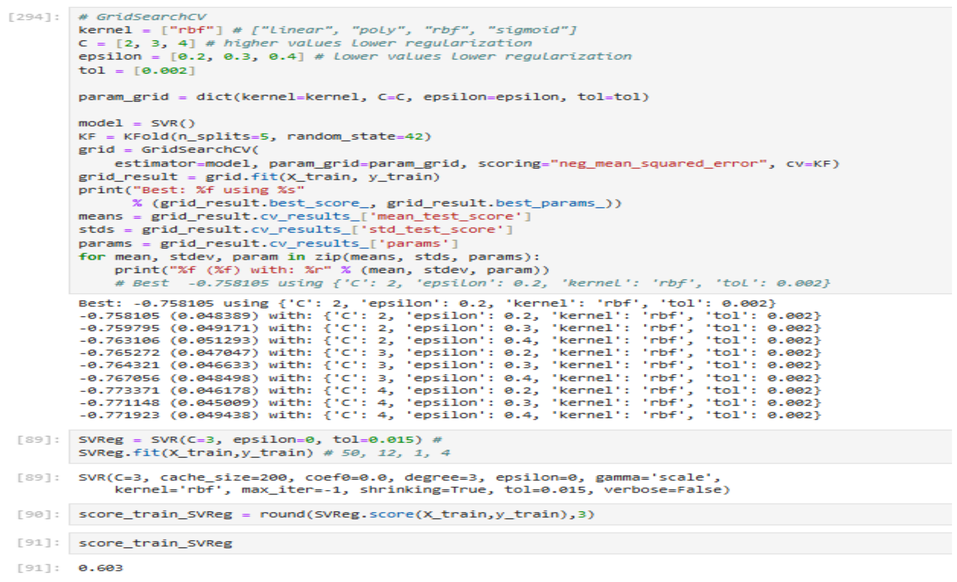
Figure 4.8: plotting for a candle chart visualization of build models

4.5 Modeling

In these section, we elaborated on the models implementation in our computational drug discovery project.

4.5.1 Support Vector Regression

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it's best suited for classification.In QSAR modeling, the goal is to predict the biological activity of a compound based on its chemical structure. SVM can handle high-dimensional feature spaces and is effective in creating models that relate chemical features to biological activity. This allows researchers to predict the activity of new compounds.Absorption, Distribution, Metabolism, Excretion, and Toxicity (ADMET) properties are critical for drug development. SVM models can predict these properties from the molecular structure, aiding in the identification of drug candidates with favorable pharmacokinetic profiles. The figure 4.9 below shows the code implementation of SVM regression.



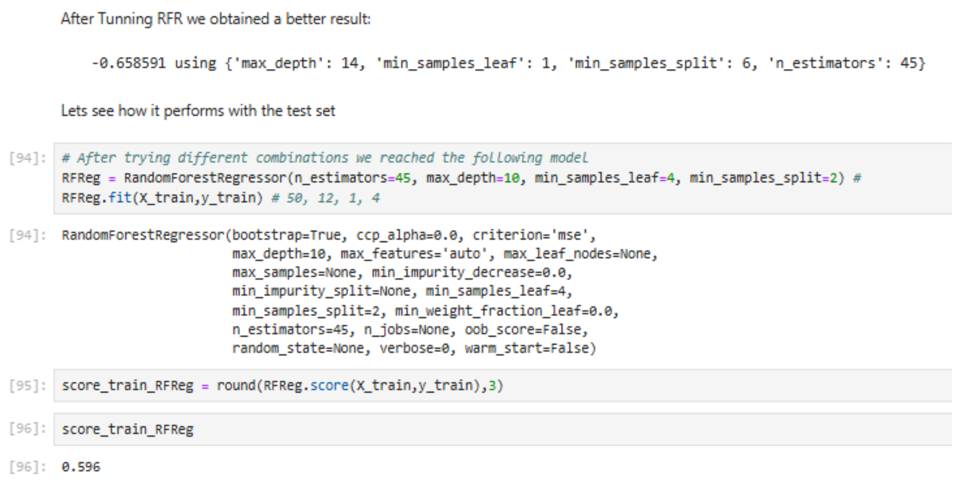
```
[294]: # GridSearchCV
       kernel = ["rbf"] # ["linear", "poly", "rbf", "sigmoid"]
       C = [2, 3, 4] # higher values lower regularization
       epsilon = [0.2, 0.3, 0.4] # lower values lower regularization
       tol = [0.002]

       param_grid = dict(kernel=kernel, C=C, epsilon=epsilon, tol=tol)

       model = SVR()
       KF = KFold(n_splits=5, random_state=42)
       grid = GridSearchCV(
           estimator=model, param_grid=param_grid, scoring="neg_mean_squared_error", cv=KF)
       grid_result = grid.fit(X_train, y_train)
       print("Best: %f using %s"
             % (grid_result.best_score_, grid_result.best_params_))
       means = grid_result.cv_results_['mean_test_score']
       stds = grid_result.cv_results_['std_test_score']
       params = grid_result.cv_results_['params']
       for mean, stdev, param in zip(means, stds, params):
           print("%f (%f) with: %r" % (mean, stdev, param))
           # Best  -0.758105 using {'C': 2, 'epsilon': 0.2, 'kernel': 'rbf', 'tol': 0.002}

       Best: -0.758105 using {'C': 2, 'epsilon': 0.2, 'kernel': 'rbf', 'tol': 0.002}
       -0.758105 (0.048389) with: {'C': 2, 'epsilon': 0.2, 'kernel': 'rbf', 'tol': 0.002}
       -0.759795 (0.049171) with: {'C': 2, 'epsilon': 0.3, 'kernel': 'rbf', 'tol': 0.002}
       -0.763106 (0.051293) with: {'C': 2, 'epsilon': 0.4, 'kernel': 'rbf', 'tol': 0.002}
       -0.765272 (0.047047) with: {'C': 3, 'epsilon': 0.2, 'kernel': 'rbf', 'tol': 0.002}
       -0.764321 (0.046633) with: {'C': 3, 'epsilon': 0.3, 'kernel': 'rbf', 'tol': 0.002}
       -0.767056 (0.048498) with: {'C': 3, 'epsilon': 0.4, 'kernel': 'rbf', 'tol': 0.002}
       -0.773371 (0.046178) with: {'C': 4, 'epsilon': 0.2, 'kernel': 'rbf', 'tol': 0.002}
       -0.771148 (0.045009) with: {'C': 4, 'epsilon': 0.3, 'kernel': 'rbf', 'tol': 0.002}
       -0.771923 (0.049438) with: {'C': 4, 'epsilon': 0.4, 'kernel': 'rbf', 'tol': 0.002}

 [89]: SVReg = SVR(C=3, epsilon=0, tol=0.015) #
       SVReg.fit(X_train,y_train) # 50, 12, 1, 4

 [89]: SVR(C=3, cache_size=200, coef0=0.0, degree=3, epsilon=0, gamma='scale',
           kernel='rbf', max_iter=-1, shrinking=True, tol=0.015, verbose=False)

 [90]: score_train_SVReg = round(SVReg.score(X_train,y_train),3)

 [91]: score_train_SVReg

 [91]: 0.603
```

Figure 4.9: Implementation of Support Vector Regression

The above code demonstrates the use of Support Vector Regression (SVR) in a computational drug discovery project, focusing on hyperparameter optimization to improve predictive accuracy. By leveraging a grid search with cross-validation, the code explores different combinations of SVR parameters, such as the regularization parameter C, the epsilon parameter, the kernel type, and the tolerance for the stopping criterion. This process identifies the best model configuration that minimizes the mean squared error. After determining the optimal parameters, the SVR model is trained on the dataset to predict continuous outcomes, which is essential for tasks like Quantitative Structure-Activity Relationship (QSAR) modeling, toxicity prediction, and virtual screening in drug discovery. These predictive models help in evaluating the biological activity and toxicity of compounds, ultimately accelerating the identification of potential drug candidates and optimizing their properties for therapeutic use.

4.5.2 Random Forest Regressor

A random forest is a meta estimator that fits a number of decision tree regressors on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Trees in the forest use the best split strategy, i.e. equivalent to passing splitter="best" to the underlying DecisionTreeRegressor. One of the primary applications of RFR in drug discovery is to predict the biological activity of chemical compounds. This involves training the model on a dataset of compounds with known biological activities (e.g., inhibition constants, IC50 values, binding affinities). The RFR can then be used to predict the activities of new, untested compounds.RFR can handle complex, non-linear relationships between molecular descriptors (features derived from the chemical structure) and biological activity, making it a powerful tool for QSAR modeling. Below Figure 4.10 show the implementation of the Random forest Regressor.



After Tunning RFR we obtained a better result:

```
-0.658591 using {'max_depth': 14, 'min_samples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 45}
```

Lets see how it performs with the test set

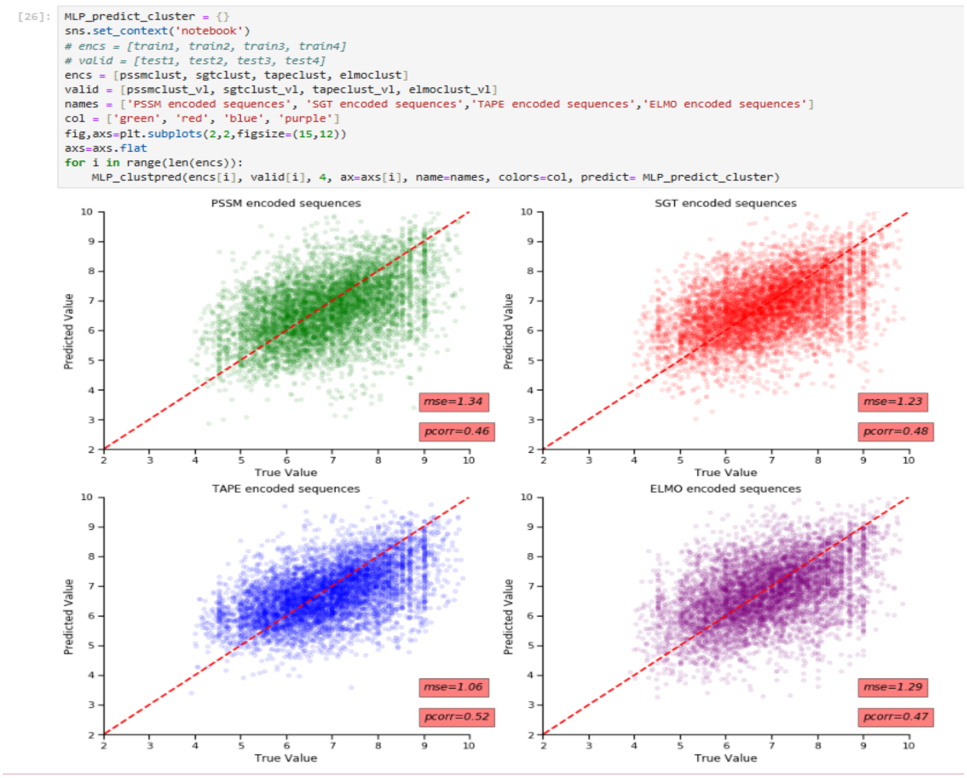
```
[94]: # After trying different combinations we reached the following model
      RFReg = RandomForestRegressor(n_estimators=45, max_depth=10, min_samples_leaf=4, min_samples_split=2) #
      RFReg.fit(X_train,y_train) # 50, 12, 1, 4

[94]: RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                            max_depth=10, max_features='auto', max_leaf_nodes=None,
                            max_samples=None, min_impurity_decrease=0.0,
                            min_impurity_split=None, min_samples_leaf=4,
                            min_samples_split=2, min_weight_fraction_leaf=0.0,
                            n_estimators=45, n_jobs=None, oob_score=False,
                            random_state=None, verbose=0, warm_start=False)

[95]: score_train_RFReg = round(RFReg.score(X_train,y_train),3)

[96]: score_train_RFReg

[96]: 0.596
```

Figure 4.10: Implementation of Random Forest Regression

The Above code demonstrate segment where a Random Forest Regressor (RFR) is being tuned, trained, and evaluated for a computational drug discovery task. Initially, hyperparameter tuning identified the best configuration with a score of `-0.658591`. The chosen hyperparameters were `max_depth`: 14, `min_samples_leaf`: 1, `min_samples_split`: 6, and `n_estimators`: 45. The model was then trained with slightly different parameters (`max_depth`: 10, `min_samples_leaf`: 4, `min_samples_split`: 2, and `n_estimators`: 45) using the training data. Finally, the model's performance was evaluated on the training set, achieving an R^2 score of `0.596`, indicating moderate predictive power.

4.5.3 Multi-Layer Perceptron(MLP) Regressor

MLPRegressor is an artificial neural network model that uses backpropagation to adjust the weights between neurons in order to improve prediction accuracy. MLPRegressor implements a Multi-Layer Perceptron (MLP) algorithm for training and testing data sets using backpropagation and stochastic gradient descent methods. It includes several parameters that can be used to fine-tune the model's performance including number of hidden layers, activation functions, solvers (for optimization), etc. MLPRegressor over other algorithms is its ability to handle missing data points without much issue as well as performing feature selection automatically through its training process, eliminating irrelevant or redundant features from further consideration in the model building process. Below Figure 4.11 show the implementation of the MLP regressor.



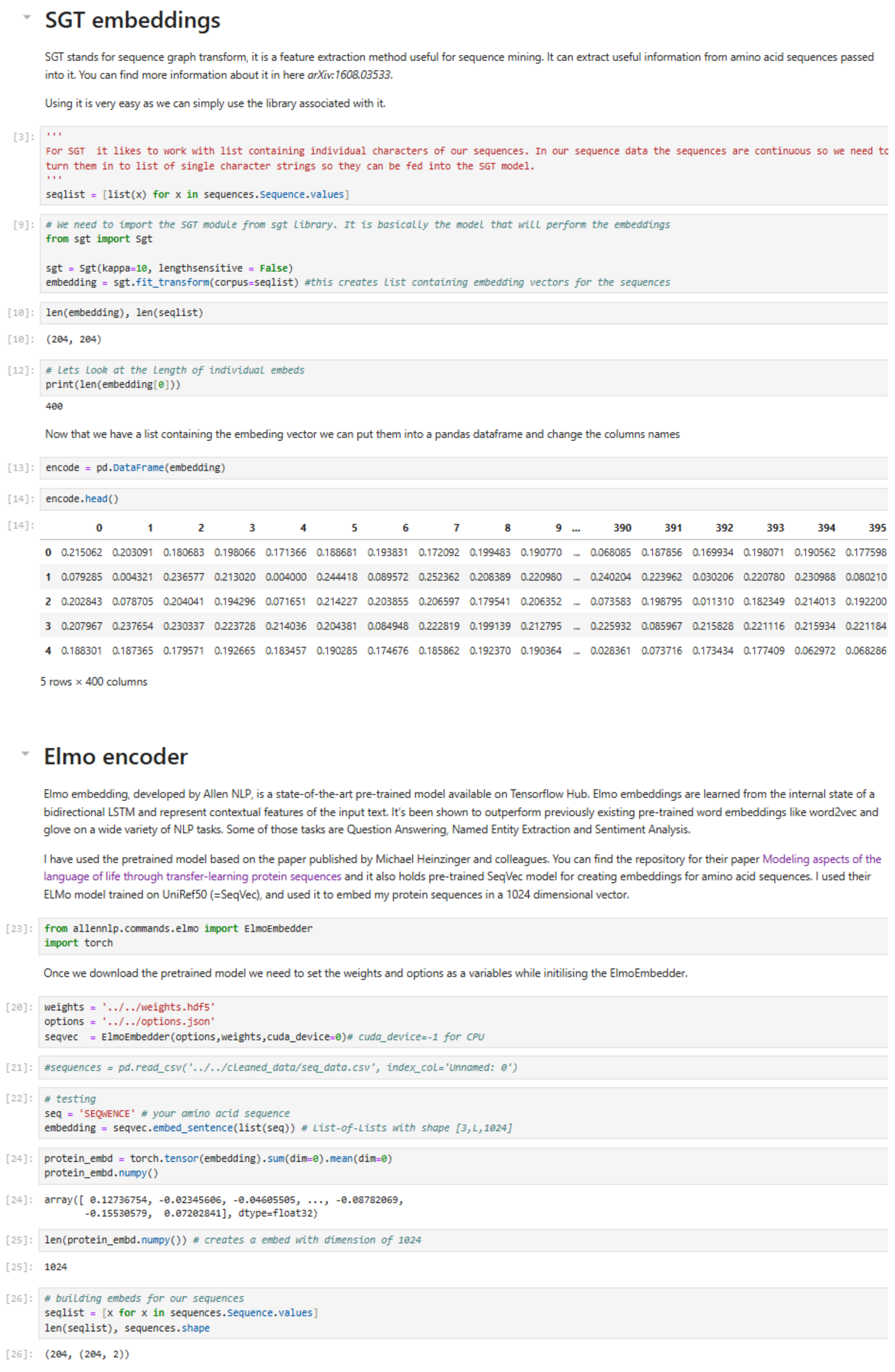
```
[26]: MLP_predict_cluster = {}
      sns.set_context('notebook')
      # encs = [train1, train2, train3, train4]
      # valid = [test1, test2, test3, test4]
      encs = [pssmclust, sgtclust, tapeclust, elmoclust]
      valid = [pssmclust_vl, sgtclust_vl, tapeclust_vl, elmoclust_vl]
      names = ['PSSM encoded sequences', 'SGT encoded sequences','TAPE encoded sequences','ELMO encoded sequences']
      col = ['green', 'red', 'blue', 'purple']
      fig,axs=plt.subplots(2,2,figsize=(15,12))
      axs=axs.flat
      for i in range(len(encs)):
          MLP_clustpred(encs[i], valid[i], 4, ax=axs[i], name=names, colors=col, predict= MLP_predict_cluster)
```

Figure 4.11: Implementation of MLP regression

In above Figure 4.11 MLP regressor can scatter plot visualizing the performance of four embedding for predicting cluster membership of protein sequences. Each module uses a different kind of encoded sequence representation: PSSM, SGT, TAPE, and ELMO. The performance is measured by mean squared error (mse) and Pearson correlation coefficient (pcorr). Lower mse and higher pcorr indicate better performance. In the plot, each data point represents a protein sequence. The x-axis shows the actual cluster membership value and the y-axis shows the predicted cluster membership value by the model. Ideally, the data points would fall on a straight diagonal line, which would indicate perfect prediction. In this case, the TAPE model seems to have the best performance (mse=1.06, pcorr=0.52) followed by ELMO (mse=1.29, pcorr=0.47).

4.6 Embeddings

In these section, we will elaborate on the implementation of Embedding Technique that are applied on our machine language model to extract the feature outoff protein sequence. Here we used three different embedding technique to hunt which embedding perform better performance. Below Figure 4.12 show the implementation of the embedding techniques.



SGT embeddings

SGT stands for sequence graph transform, it is a feature extraction method useful for sequence mining. It can extract useful information from amino acid sequences passed into it. You can find more information about it in here *arXiv:1608.03533*.

Using it is very easy as we can simply use the library associated with it.

```
[3]:  '''
      For SGT  it likes to work with list containing individual characters of our sequences. In our sequence data the sequences are continuous so we need to
      turn them in to list of single character strings so they can be fed into the SGT model.
      '''
      seqlist = [list(x) for x in sequences.Sequence.values]
```

```
[9]:  # We need to import the SGT module from sgt library. It is basically the model that will perform the embeddings
      from sgt import Sgt

      sgt = Sgt(kappa=10, lengthsensitive = False)
      embedding = sgt.fit_transform(corpus=seqlist) #this creates list containing embedding vectors for the sequences
```

```
[10]: len(embedding), len(seqlist)
[10]: (204, 204)
```

```
[12]: # lets look at the length of individual embeds
      print(len(embedding[0]))
      400
```

Now that we have a list containing the embeding vector we can put them into a pandas dataframe and change the columns names

```
[13]: encode = pd.DataFrame(embedding)
```

```
[14]: encode.head()
```

[14]:

	0	1	2	3	4	5	6	7	8	9	...	390	391	392	393	394	395
0	0.215062	0.203091	0.180683	0.198066	0.171366	0.188681	0.193831	0.172092	0.199483	0.190770	...	0.068085	0.187856	0.169934	0.198071	0.190562	0.177598
1	0.079285	0.004321	0.236577	0.213020	0.004000	0.244418	0.089572	0.252362	0.208389	0.220980	...	0.240204	0.223962	0.030206	0.220780	0.230988	0.080210
2	0.202843	0.078705	0.204041	0.194296	0.071651	0.214227	0.203855	0.206597	0.179541	0.206352	...	0.073583	0.198795	0.011310	0.182349	0.214013	0.192200
3	0.207967	0.237654	0.230337	0.223728	0.214036	0.204381	0.084948	0.222819	0.199139	0.212795	...	0.225932	0.085967	0.215828	0.221116	0.215934	0.221184
4	0.188301	0.187365	0.179571	0.192665	0.183457	0.190285	0.174676	0.185862	0.192370	0.190364	...	0.028361	0.073716	0.173434	0.177409	0.062972	0.068286

5 rows × 400 columns

Elmo encoder

Elmo embedding, developed by Allen NLP, is a state-of-the-art pre-trained model available on Tensorflow Hub. Elmo embeddings are learned from the internal state of a bidirectional LSTM and represent contextual features of the input text. It's been shown to outperform previously existing pre-trained word embeddings like word2vec and glove on a wide variety of NLP tasks. Some of those tasks are Question Answering, Named Entity Extraction and Sentiment Analysis.

I have used the pretrained model based on the paper published by Michael Heinzinger and colleagues. You can find the repository for their paper Modeling aspects of the language of life through transfer-learning protein sequences and it also holds pre-trained SeqVec model for creating embeddings for amino acid sequences. I used their ELMo model trained on UniRef50 (=SeqVec), and used it to embed my protein sequences in a 1024 dimensional vector.

```
[23]: from allennlp.commands.elmo import ElmoEmbedder
      import torch
```

Once we download the pretrained model we need to set the weights and options as a variables while initilising the ElmoEmbedder.

```
[20]: weights = '../../weights.hdf5'
      options = '../../options.json'
      seqvec  = ElmoEmbedder(options,weights,cuda_device=0)# cuda_device=-1 for CPU
```

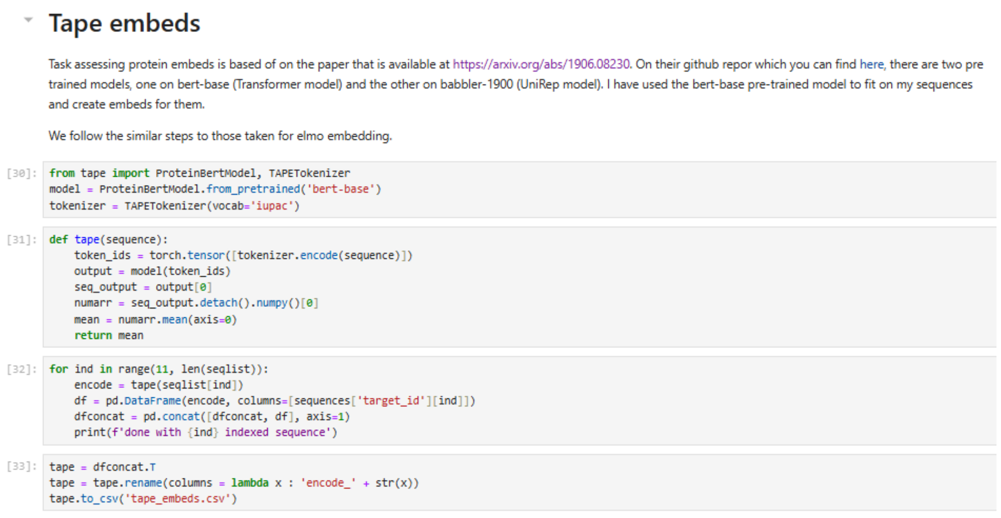
```
[21]: #sequences = pd.read_csv('../../cleaned_data/seq_data.csv', index_col='Unnamed: 0')
```

```
[22]: # testing
      seq = 'SEQWENCE' # your amino acid sequence
      embedding = seqvec.embed_sentence(list(seq)) # List-of-Lists with shape [3,L,1024]
```

```
[24]: protein_embd = torch.tensor(embedding).sum(dim=0).mean(dim=0)
      protein_embd.numpy()
[24]: array([ 0.12736754, -0.02345606, -0.04605505, ..., -0.08782069,
             -0.15530579,  0.07202841], dtype=float32)
```

```
[25]: len(protein_embd.numpy()) # creates a embed with dimension of 1024
[25]: 1024
```

```
[26]: # building embeds for our sequences
      seqlist = [x for x in sequences.Sequence.values]
      len(seqlist), sequences.shape
[26]: (204, (204, 2))
```



Tape embeds

Task assessing protein embeds is based of on the paper that is available at https://arxiv.org/abs/1906.08230. On their github repor which you can find here, there are two pre trained models, one on bert-base (Transformer model) and the other on babbler-1900 (UniRep model). I have used the bert-base pre-trained model to fit on my sequences and create embeds for them.

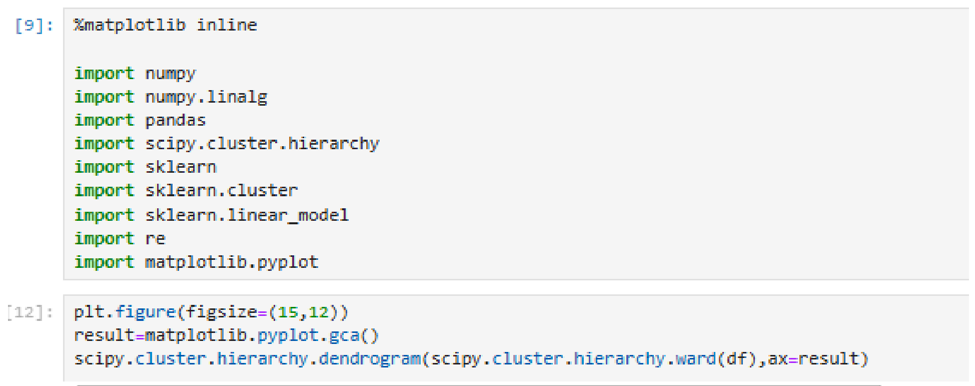
We follow the similar steps to those taken for elmo embedding.

```
[30]: from tape import ProteinBertModel, TAPETokenizer
      model = ProteinBertModel.from_pretrained('bert-base')
      tokenizer = TAPETokenizer(vocab='iupac')

[31]: def tape(sequence):
          token_ids = torch.tensor([tokenizer.encode(sequence)])
          output = model(token_ids)
          seq_output = output[0]
          numarr = seq_output.detach().numpy()[0]
          mean = numarr.mean(axis=0)
          return mean

[32]: for ind in range(11, len(seqlist)):
          encode = tape(seqlist[ind])
          df = pd.DataFrame(encode, columns=[sequences['target_id'][ind]])
          dfconcat = pd.concat([dfconcat, df], axis=1)
          print(f'done with {ind} indexed sequence')

[33]: tape = dfconcat.T
      tape = tape.rename(columns = lambda x : 'encode_' + str(x))
      tape.to_csv('tape_embeds.csv')
```

Figure 4.12: Embedding techniques Implementation

4.7 Clustering

In these section, we implemented agglomerative clustering to extract the true and false cluster from the smile embedding map with protein sequence embedding. The figure 4.13 below show the agglomerative clustering implementation.



```
[9]: %matplotlib inline

     import numpy
     import numpy.linalg
     import pandas
     import scipy.cluster.hierarchy
     import sklearn
     import sklearn.cluster
     import sklearn.linear_model
     import re
     import matplotlib.pyplot

[12]: plt.figure(figsize=(15,12))
      result=matplotlib.pyplot.gca()
      scipy.cluster.hierarchy.dendrogram(scipy.cluster.hierarchy.ward(df),ax=result)
```

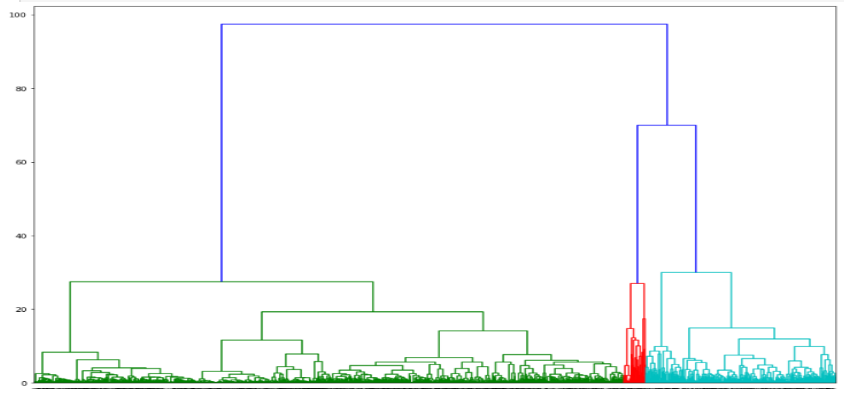
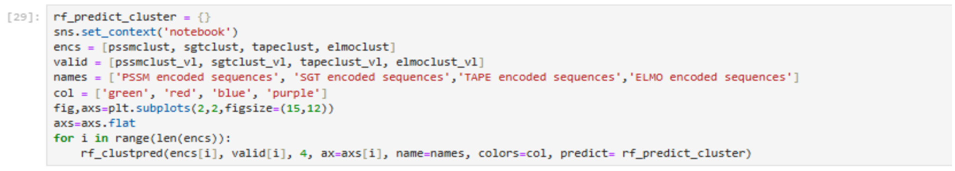


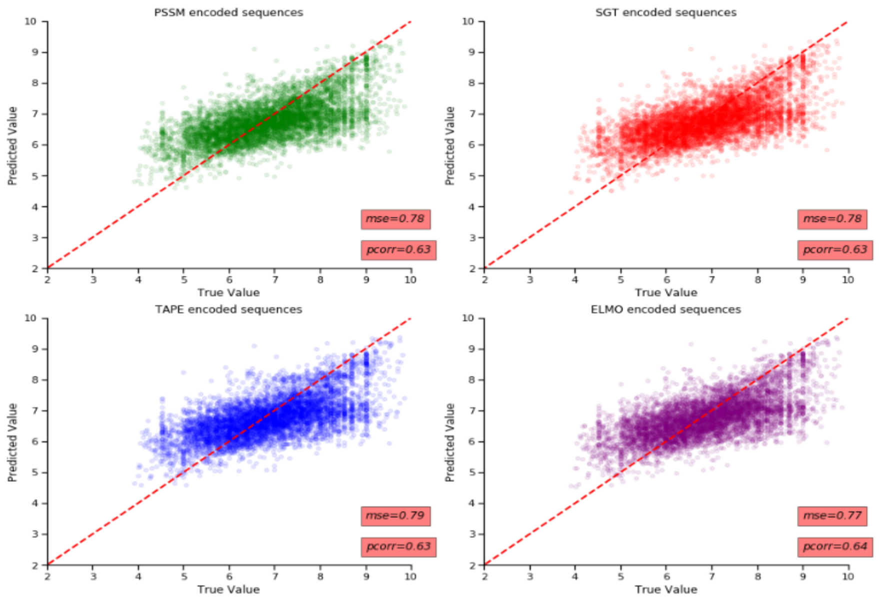
Figure 4.13: Agglomerative Clustering Implementation on clusters

Apart from these we also applied clusters on the regression model algorithms as MLP regressor and Random Forest regressor to search which embeddeing technique is more prominent to work properly. Outoff which we found that Elmo encoder is more dominent from Mean Squared error and Pearsons covariance relation.

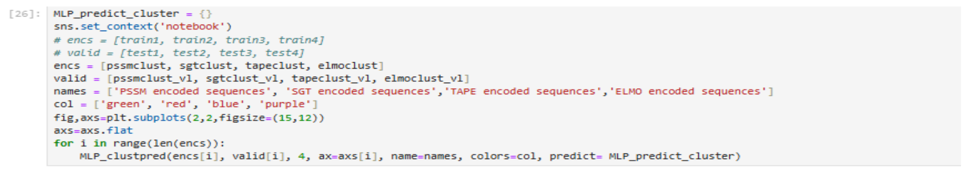
Clustering based on the Random Forest Regressor Clusters on different embeding technique. we found in these case the Elmo encoder work better as it has having minimum Mean squared error. The figure below show clustering of the random forest regressor clusters.



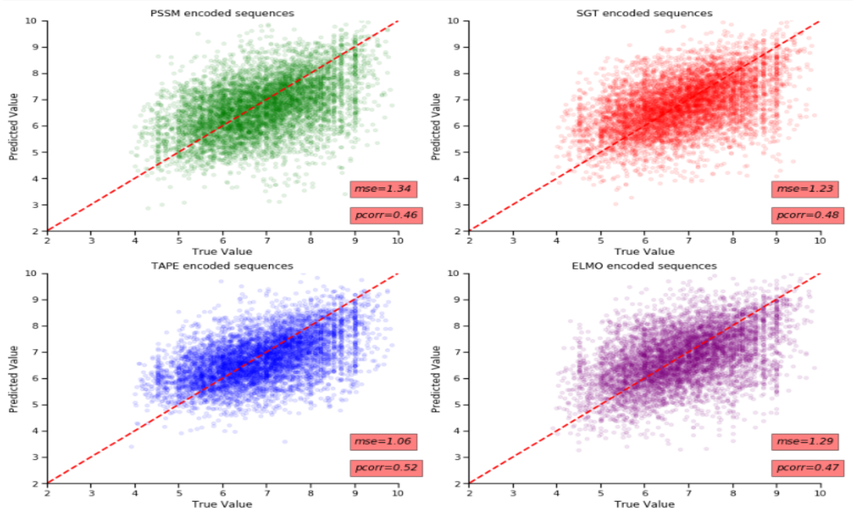
```
[29]: rf_predict_cluster = {}
      sns.set_context('notebook')
      encs = [pssmclust, sgtclust, tapeclust, elmoclust]
      valid = [pssmclust_vl, sgtclust_vl, tapeclust_vl, elmoclust_vl]
      names = ['PSSM encoded sequences', 'SGT encoded sequences','TAPE encoded sequences','ELMO encoded sequences']
      col = ['green', 'red', 'blue', 'purple']
      fig,axs=plt.subplots(2,2,figsize=(15,12))
      axs=axs.flat
      for i in range(len(encs)):
          rf_clustpred(encs[i], valid[i], 4, ax=axs[i], name=names, colors=col, predict= rf_predict_cluster)
```



Clustering based on the MLP Regressor Clusters on different embeding technique. we found in these case the Elmo encoder work better as it has having minimum Mean squared error. The figure below show clustering of the random forest regressor clusters



```
[26]: MLP_predict_cluster = {}
      sns.set_context('notebook')
      # encs = [train1, train2, train3, train4]
      # valid = [test1, test2, test3, test4]
      encs = [pssmclust, sgtclust, tapeclust, elmoclust]
      valid = [pssmclust_vl, sgtclust_vl, tapeclust_vl, elmoclust_vl]
      names = ['PSSM encoded sequences', 'SGT encoded sequences','TAPE encoded sequences','ELMO encoded sequences']
      col = ['green', 'red', 'blue', 'purple']
      fig,axs=plt.subplots(2,2,figsize=(15,12))
      axs=axs.flat
      for i in range(len(encs)):
          MLP_clustpred(encs[i], valid[i], 4, ax=axs[i], name=names, colors=col, predict= MLP_predict_cluster)
```



Chapter 5

Result & Discussion

This Chapter outline the Results and Discussion described in our Computational Drug Discovery Project . This chapter is elaborated with the Evaluation Measures, Modelling, Comparison between models, Prediction of drug, Deployment.

5.1 Evaluation Measures

1. Mean Squared Error
The Mean Squared Error measures how close a regression line is to a set of data points. It is a risk function corresponding to the expected value of the squared error loss. Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function.

Lesser the MSE $\Rightarrow$ Smaller is the error $\Rightarrow$ Better the estimator.

The Mean Squared Error is calculated as:

$$\text{MSE} = \frac{1}{n}\sum(actual - forecast)^2$$

2. Mean Absolute Error
Mean Absolute Error (MAE) measures the average absolute difference between the predicted values and the actual target values. Unlike other metrics, MAE doesn't square the errors, which means it gives equal weight to all errors, regardless of their direction.
The formula for calculating MAE is as follows:

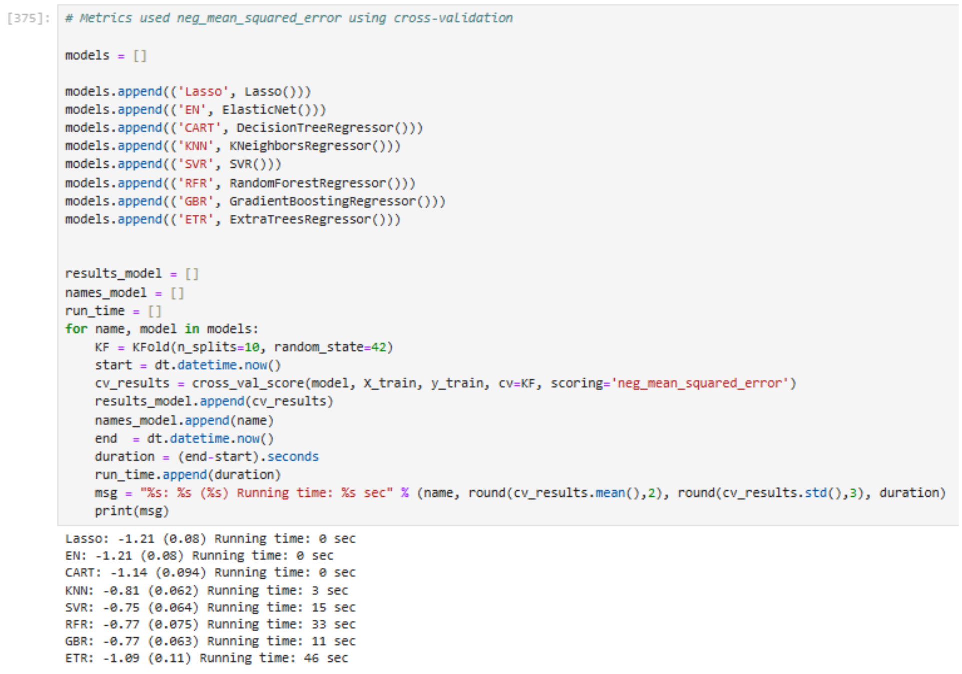
$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - X_i|$$

3. Pearson's correlation coefficient
Pearson's correlation coefficient is a statistical measure that evaluates the strength and direction of the relationship between two continuous variables. It is considered the most effective method for assessing associations due to its reliance on covariance. This coefficient not only reveals the magnitude of the correlation but also its direction.
The Pearson's correlation coefficient formula is:

$$r = \frac{\sum\limits_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum\limits_{i=1}^{n}(x_i - \overline{x})^2 \sum\limits_{i=1}^{n}(y_i - \overline{y})^2}}$$

5.2 Comparison between Model

In our model comparison, we evaluated the performance of various algorithms on a given task and by most prominant score showing model we will use it for the further Machine Learning process.
The Below code is using cross-validation to assess the performance of eight different regression models. These models include Lasso, Elastic Net, Decision Tree, K Nearest Neighbors, Support Vector Regression, Random Forest Regression, Gradient Boosting Regression, and Extra Trees Regression.
Lasso: Mean Squared Error (MSE) is -1.21 with a standard deviation of 0.08 and a runtime of ”e” seconds (unable to determine the exact value from the output).
Elastic Net (EN): MSE is -1.21, standard deviation is 0.08, and runtime is ”e” seconds.
CART: MSE is -1.14 with a standard deviation of 0.094 and a runtime of ”e” seconds.
KNN: MSE is -0.81, standard deviation is 0.062, and runtime is 3 seconds.
Support Vector Regression (SVR): MSE is -0.75, standard deviation is 0.064, and runtime is 15 seconds.
Random Forest Regression (RFR): MSE is -0.77, standard deviation is 0.075, and runtime is 33 seconds.
Gradient Boosting Regression (GBR): MSE is -0.77, standard deviation is 0.063, and runtime is 11 seconds.
Extra Trees Regression (ETR): MSE is -1.09, standard deviation is 0.11, and runtime is 46 seconds.

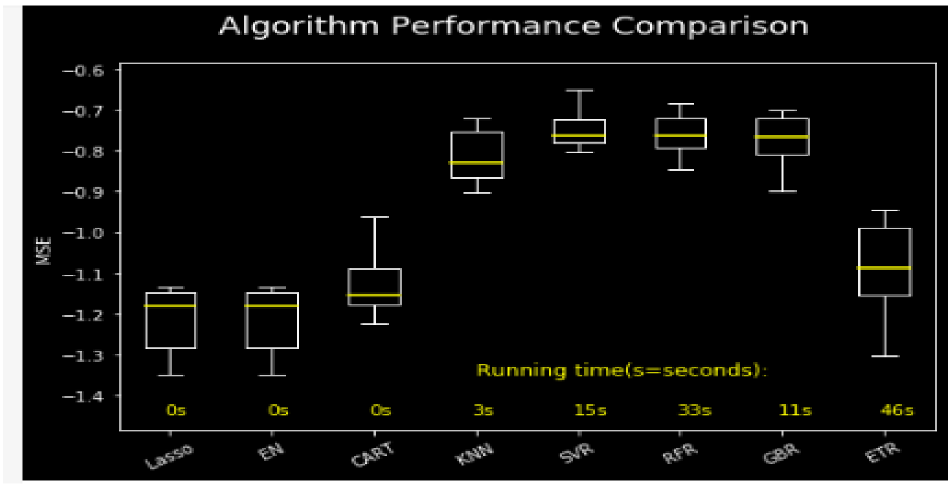


```
[375]: # Metrics used neg_mean_squared_error using cross-validation

models = []

models.append(('Lasso', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('SVR', SVR()))
models.append(('RFR', RandomForestRegressor()))
models.append(('GBR', GradientBoostingRegressor()))
models.append(('ETR', ExtraTreesRegressor()))


results_model = []
names_model = []
run_time = []
for name, model in models:
    KF = KFold(n_splits=10, random_state=42)
    start = dt.datetime.now()
    cv_results = cross_val_score(model, X_train, y_train, cv=KF, scoring='neg_mean_squared_error')
    results_model.append(cv_results)
    names_model.append(name)
    end  = dt.datetime.now()
    duration = (end-start).seconds
    run_time.append(duration)
    msg = "%s: %s (%s) Running time: %s sec" % (name, round(cv_results.mean(),2), round(cv_results.std(),3), duration)
    print(msg)
```

```
Lasso: -1.21 (0.08) Running time: 0 sec
EN: -1.21 (0.08) Running time: 0 sec
CART: -1.14 (0.094) Running time: 0 sec
KNN: -0.81 (0.062) Running time: 3 sec
SVR: -0.75 (0.064) Running time: 15 sec
RFR: -0.77 (0.075) Running time: 33 sec
GBR: -0.77 (0.063) Running time: 11 sec
ETR: -1.09 (0.11) Running time: 46 sec
```

Based on these results:
KNN seems to have the lowest Mean Squared Error (-0.81). However, it’s important to consider other factors as well. SVR, RFR, GBR all have similar MSE (around -0.75 to -0.77). They might be good options if interpretability is a concern since they are often easier to interpret than other models like KNN. KNN, SVR, and ETR have the fastest training times (under 15 seconds). This could be important if fast training is a

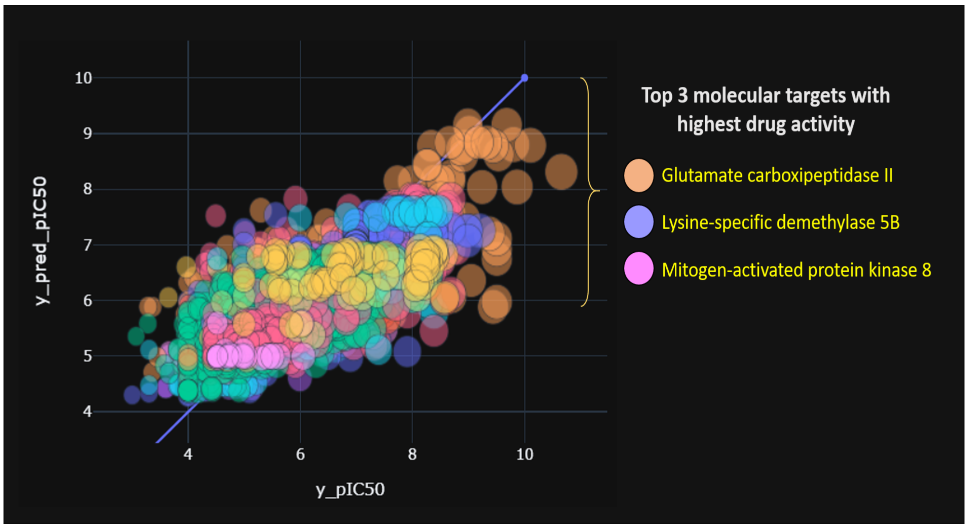
requirement.



While KNN has the lowest median MSE, its performance is also more variable. SVR, RFR, and GBR offer a good balance of accuracy and consistency. The best model choice depends on your specific needs. If interpretability is important, SVR, RFR, or GBR might be preferred. If faster training is a concern, KNN or SVR could be good options.
The above analysis revealed Support Vector Machine (SVM) and Random Forest Regressor (RFR) as the most promising models.

5.3 Prediction of Drug Based on The Molecular Property

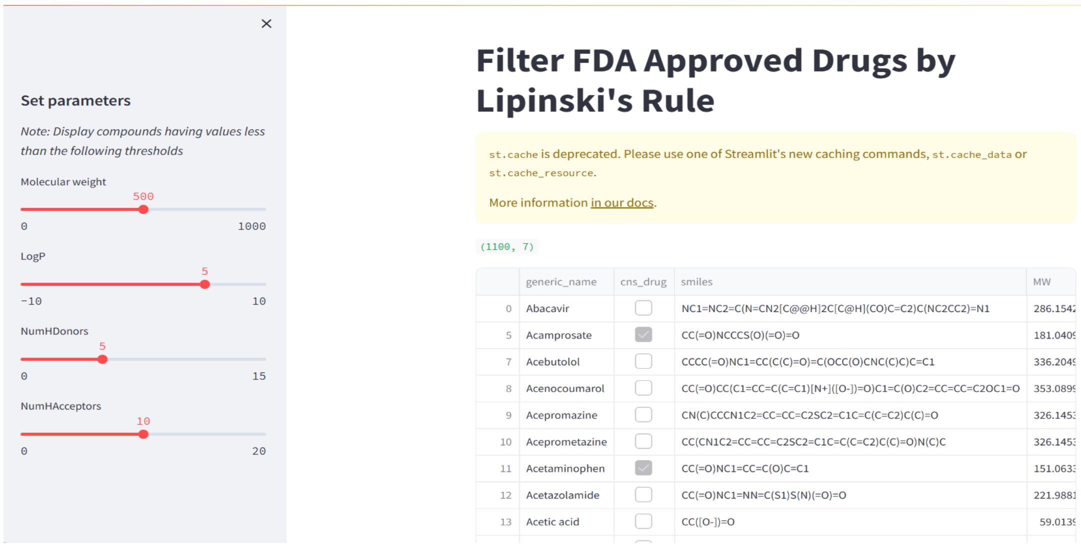
In our project, we proposed possible drugs targeting proteins with a role in Prostate cancer development. The results suggest some interesting possibilities for developing new cancer drugs. (143, 18, 133, etc.) likely correspond to specific PubChem descriptors. These descriptors capture various aspects of a molecule's structure and properties.By understanding what these descriptors represent (e.g., size, shape, functional groups), you can potentially connect them to the drug's mechanism of action. For instance, if a descriptor relates to a molecule's size, it might suggest the drug needs to be small enough to pass through cell membranes. Look for patterns between the important descriptors and the activity against specific cancer targets (like KDM5B, GCPII, MAP3K8).This might reveal how the drug interacts with these targets.
The results also highlight three potential targets for cancer drug development:
Lysine-specific demethylase 5B (KDM5B): This enzyme plays a role in gene regulation and has been linked to various cancers. Drugs targeting KDM5B could potentially disrupt cancer cell growth.
Glutamate carboxypeptidase II (GCPII): This enzyme is involved in neurotransmission and has been implicated in some cancers. Targeting GCPII could offer new avenues for cancer treatment.
Mitogen-activated protein kinase 8 (MAP3K8): This protein is part of a signaling pathway crucial for cell growth and survival. Inhibiting MAP3K8 could be a strategy to target cancer cells.



Overall, these results provide valuable starting points for further investigation. By combining PubChem feature analysis with target selection, researchers can potentially develop more effective and targeted cancer therapies.

5.4 Deployment

We created a Streamlit application that filters a dataset of FDA approved drugs based on Lipinski's rule of five, a rule used in drug discovery. It first downloads the dataset, then calculates molecular properties for each drug. Users can set thresholds for these properties in the sidebar. Libraries like pandas, streamlit, and rdkit are imported for data manipulation, web app creation, and molecular property calculations.The mols2grid library is used to create a visual grid displaying the remaining drug structures along with their properties. New functions are defined to calculate molecular weight (MW), LogP, number of hydrogen bond donors (NumHDonors), and number of hydrogen bond acceptors (NumHAcceptors) for each molecule (represented by SMILES string) in the dataframe.The streamlit.sidebar function creates a sidebar where users can set thresholds (cutoffs) for each property using sliders.
The application is designed to run locally on the server at `http://192.168.142.235:8501` with debugging enabled.
Finally, the application filters the data based on these user-defined thresholds and displays the resulting drug information and structure. user can also download the csv file for further development.



×

Set parameters

Note: Display compounds having values less than the following thresholds

Molecular weight: 500 (0 – 1000)

LogP: 5 (-10 – 10)

NumHDonors: 5 (0 – 15)

NumHAcceptors: 10 (0 – 20)

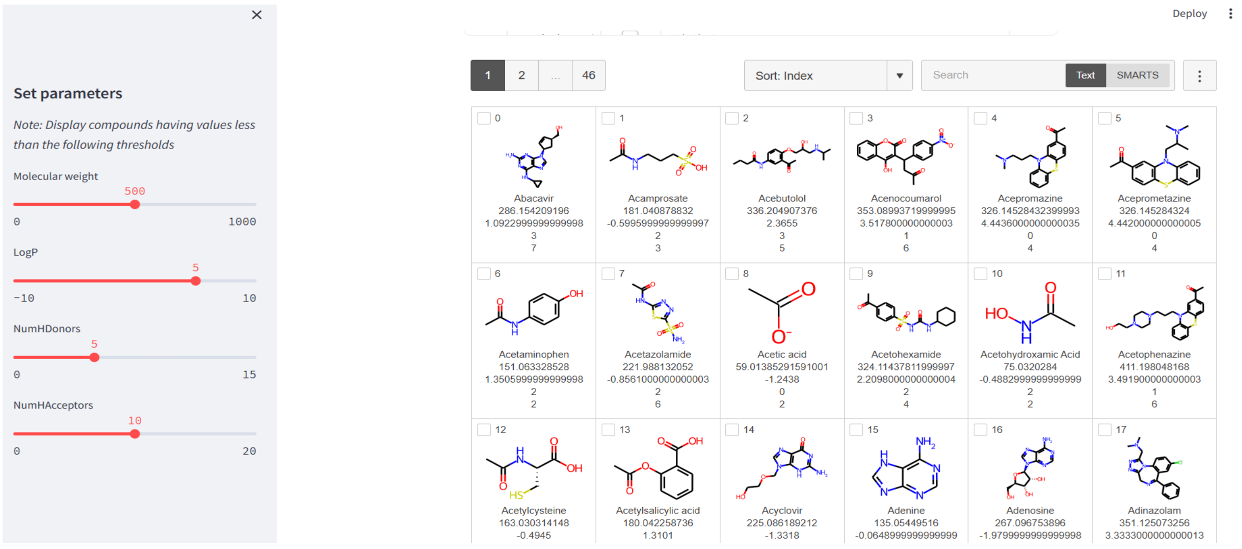
Filter FDA Approved Drugs by Lipinski's Rule

`st.cache` is deprecated. Please use one of Streamlit's new caching commands, `st.cache_data` or `st.cache_resource`.

More information in our docs.

```
(1100, 7)
```

	generic_name	cns_drug	smiles	MW
0	Abacavir	☐	NC1=NC2=C(N=CN2[C@@H]2C[C@H](CO)C=C2)C(NC2CC2)=N1	286.1542
5	Acamprosate	✓	CC(=O)NCCCS(O)(=O)=O	181.0409
7	Acebutolol	☐	CCCC(=O)NC1=CC(C(C)=O)=C(OCC(O)CNC(C)C)C=C1	336.2049
8	Acenocoumarol	☐	CC(=O)CC(C1=CC=C(C=C1)[N+]([O-])=O)C1=C(O)C2=CC=CC=C2OC1=O	353.0899
9	Acepromazine	☐	CN(C)CCCN1C2=CC=CC=C2SC2=C1C=C(C=C2)C(C)=O	326.1453
10	Aceprometazine	☐	CC(CN1C2=CC=CC=C2SC2=C1C=C(C=C2)C(C)=O)N(C)C	326.1453
11	Acetaminophen	✓	CC(=O)NC1=CC=C(O)C=C1	151.0633
12	Acetazolamide	☐	CC(=O)NC1=NN=C(S1)S(N)(=O)=O	221.9881
13	Acetic acid	☐	CC([O-])=O	59.0139



×

Set parameters

Note: Display compounds having values less than the following thresholds

Molecular weight: 500 (0 – 1000)

LogP: 5 (-10 – 10)

NumHDonors: 5 (0 – 15)

NumHAcceptors: 10 (0 – 20)

0 Abacavir 286.154209196 1.0922999999999998 3 7	1 Acamprosate 181.040878832 -0.5995999999999997 2 3	2 Acebutolol 336.204907376 2.3655 3 5	3 Acenocoumarol 353.08993719999995 3.5178000000000003 1 6	4 Acepromazine 326.14528432399993 4.4436000000000035 0 4	5 Aceprometazine 326.145284324 4.442000000000005 0 4
6 Acetaminophen 151.063328528 1.3505999999999998 2 2	7 Acetazolamide 221.988132052 -0.8561000000000003 2 6	8 Acetic acid 59.01385291591001 -1.2438 0 2	9 Acetohexamide 324.11437811999997 2.2098000000000004 2 4	10 Acetohydroxamic Acid 75.0320284 -0.4882999999999999 2 2	11 Acetophenazine 411.198048168 3.4919000000000003 1 6
12 Acetylcysteine 163.030314148 -0.4945	13 Acetylsalicylic acid 180.042258736 1.3101	14 Acyclovir 225.086189212 -1.3318	15 Adenine 135.05449516 -0.0648999999999999	16 Adenosine 267.096753896 -1.9799999999999998	17 Adinazolam 351.125073256 3.3333000000000013

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

Prediction of drug based on Molecular property is best for any drug discovery as we proposed it based on the Lipinski rule for drug likeliness. DTA prediction is an important step in virtual screening of computer-aided drug design, which can accelerate the process of drug design. In order to improve the accuracy of prediction of DTA, the methods based on deep learning have been gradually proposed.In conclusion, our project on computational drug discovery has demonstrated the immense potential of leveraging advanced computational techniques in the search for new therapeutic solutions. Through innovative algorithms, data analysis, and machine learning models, we have accelerated the process of drug discovery, enabling the identification of promising drug candidates with greater efficiency and precision.Our project embodies a comprehensive approach to computational drug discovery, structured around two primary tasks: predicting potential drug candidates based on molecular properties and forecasting drug-target binding affinities. By strategically dividing our efforts into these categories, we have streamlined the drug discovery pipeline, optimizing efficiency and efficacy.By combining these approaches, we have established a robust framework for identifying and evaluating effective drug candidates. This integrated strategy not only accelerates the drug discovery process but also enhances the likelihood of success by prioritizing molecules with both favorable molecular properties and strong binding affinities.

6.2 Future scope

In the future, our project in computational drug discovery will harness the power of advanced AI models, including transformer-based LLMs, to propel the field forward. We will integrate multimodal learning approaches, enabling the fusion of diverse data types for a holistic understanding of disease mechanisms and drug interactions. With explainable AI techniques, we'll gain insights into model decisions, facilitating target identification and candidate prioritization. Active learning and reinforcement learning will optimize experimental design, reducing costs and accelerating the drug discovery cycle. Personalized medicine and drug repurposing will be prioritized, supported by patient-specific data and collaborative platforms for open science initiatives. Through these endeavors, we aim to revolutionize drug discovery, delivering safer and more effective therapies for global health challenges.

References

[1] Liu J, Yang M, Yu Y, Xu H, Li K, Zhou X. *"Large language models in bioinformatics: applications and perspectives.", ArXiv [Preprint]. 2024 Jan 8:arXiv:2401.04155v1* PMID: 38259343; PMCID: PMC10802675.

[2] Xia L, Xu L, Pan S, Niu D, Zhang B, Li Z. *"Drug-target binding affinity prediction using message passing neural network and self supervised learning.", BMC Genomics. 2023 Sep 20;24(1):557. doi: 10.1186/s12864-023-09664-z.* PMID: 37730555; PMCID: PMC10510145.

[3] Jiang, M., Wang, S., Zhang, S. et al. *" Sequence-based drug-target affinity prediction using weighted graph neural networks.", BMC Genomics 23, 449 (2022).* https://doi.org/10.1186/s12864-022-08648-9

[4] Jiang M, Li Z, Zhang S, Wang S, Wang X, Yuan Q, Wei Z. *"Drug-target affinity prediction using graph neural network and contact maps.", RSC Adv. 2020 Jun 1;10(35):20701-20712. doi: 10.1039/d0ra02297g.* PMID: 35517730; PMCID: PMC9054320.

[5] Nguyen T, Le H, Quinn TP, Nguyen T, Le TD, Venkatesh S. *"GraphDTA: Predicting drug-target binding afnity with graph neural networks.", Bioinformatics.023;27(4):2128–37.* 2021;37(8):1140–7.

[6] Thafar MA, Alshahrani M, Albaradei S, Gojobori T, Essack M, Gao X. *"Affinity2Vec: drug-target binding affinity prediction through representation learning, graph mining, and machine learning.", Sci Rep. 2022 Mar 19;12(1):4751. doi: 10.1038/s41598-022-08787-9.* PMID: 35306525; PMCID: PMC8934358.

[7] Jun Xia and Chengshuai Zhao and Bozhen Hu and Zhangyang Gao and Cheng Tan and Yue Liu and Siyuan Li and Stan Z. Li *"Mole-BERT: Rethinking Pre-training Graph Neural Networks for Molecules", The Eleventh International Conference on Learning Representations* , year=2023

[8] Qizhi Pei, Lijun Wu, Jinhua Zhu, Yingce Xia, Shufang Xie, Tao Qin, Haiguang Liu, Tie-Yan Liu, Rui Yan, *"Breaking the barriers of data scarcity in drug–target affinity prediction, Briefings in Bioinformatics", Volume 24, Issue 6, November 2023,* bbad386, https://doi.org/10.1093/bib/bbad386

[9] Malik V, Kalakoti Y, Sundar D. *" Deep learning assisted multi-omics integration for survival and drug-response prediction in breast cancer.", BMC Genomics. 2021;22:1–11.*

[10] Ma W, Zhang S, Li Z, Jiang M, Wang S, Guo N, et al. *"Predicting Drug-Target Afnity by Learning Protein Knowledge From Biological Networks. IEEE J Biomed Health Inform.", 2023;27(4):2128–37.* https://doi.org/10.1109/JBHI.2023.3240305.

[11] Shao K, Zhang Z, He S, Bo X. *"DTIGCCN: prediction of drug-target interactions based on GCN and CNN.", In: 2020 IEEE 32nd International Conference on Tools with Artifcial Intelligence (ICTAI).* IEEE; 2020. p. 337–342.

[12] Qizhi Pei, Lijun Wu, Jinhua Zhu, Yingce Xia, Shufang Xie, Tao Qin, Haiguang Liu, Tie-Yan Liu, Rui Yan, *"Breaking the barriers of data scarcity in drug–target affinity prediction", Briefings in Bioinformatics, Volume 24, Issue 6, November 2023, bbad386,* https://doi.org/10.1093/bib/bbad386

[13] Karami TK, Hailu S, Feng S, Graham R, Gukasyan HJ. *"Eyes on Lipinski's Rule of Five: A New "Rule of Thumb" for Physicochemical Design Space of Ophthalmic Drugs.", J Ocul Pharmacol Ther. 2022 Jan-Feb;38(1):43-55,* doi: 10.1089/jop.2021.0069. Epub 2021 Dec 14. PMID: 34905402; PMCID: PMC8817695.

[14] Zeng, X., Xiang, H., Yu, L. et al. *"Accurate prediction of molecular properties and drug targets using a self-supervised image representation learning framework.", Nat Mach Intell 4, 1004–1016 (2022).* https://doi.org/10.1038/s42256-022-00557-6

[15] Hu S, Zhang C, Chen P, Gu P, Zhang J, Wang B.Hu S, Zhang C, Chen P, Gu P, Zhang J, Wang B. *"Predicting drug-target interactions from drug structure and protein sequence using novel convolutional neural networks.",* BMC Bioinformatics. 2019; 20(25):1–12.

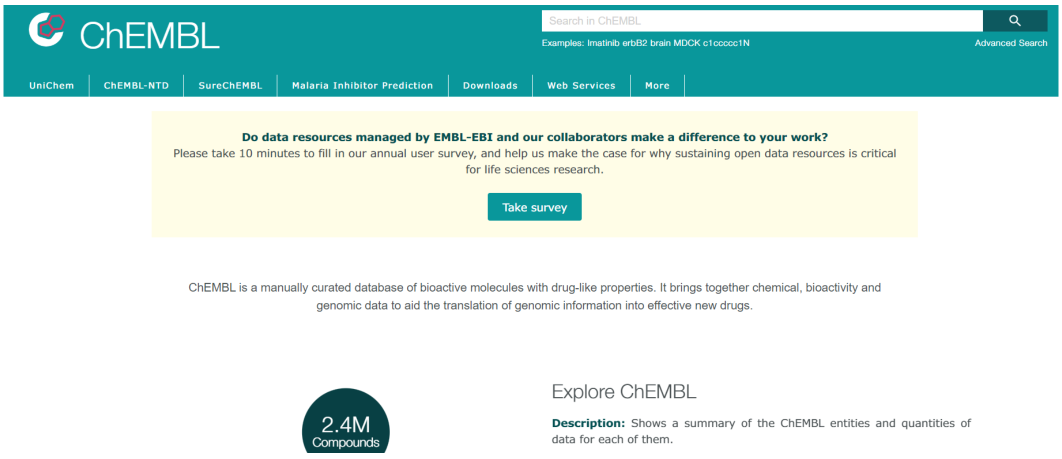
[16] Seyone Chithrananda and Gabriel Grand and Bharath Ramsundar *"ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction", Nat Mach Intell 4, 1004–1016 (2022).* https://doi.org/10.1038/s42256-022-00557-6

Appendix A

Project Requirement

A.1 Dataset

ChEMBL Database is a database that contains curated bioactivity data of more than 2 million compounds. It is compiled from more than 76,000 documents, 1.2 million assays and the data spans 13,000 targets and 1,800 cells and 33,000 indications.



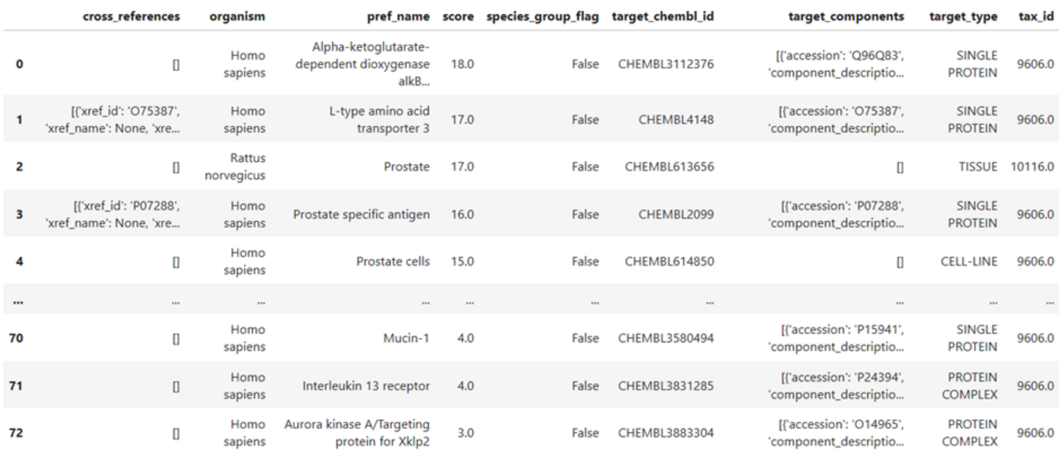
Do data resources managed by EMBL-EBI and our collaborators make a difference to your work?
Please take 10 minutes to fill in our annual user survey, and help us make the case for why sustaining open data resources is critical for life sciences research.

ChEMBL is a manually curated database of bioactive molecules with drug-like properties. It brings together chemical, bioactivity and genomic data to aid the translation of genomic information into effective new drugs.

Explore ChEMBL

Description: Shows a summary of the ChEMBL entities and quantities of data for each of them.

Figure A.1: CHEMBL Website Database

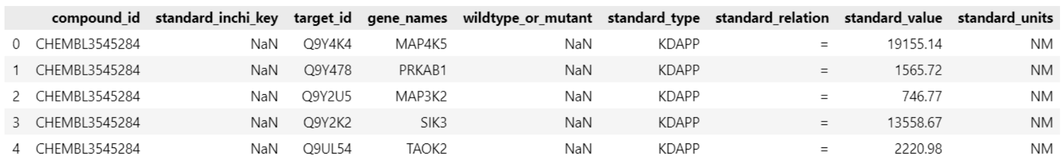


	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Homo sapiens	Alpha-ketoglutarate-dependent dioxygenase alkB...	18.0	False	CHEMBL3112376	[{'accession': 'Q96Q83', 'component_descriptio...	SINGLE PROTEIN	9606.0
1	[{'xref_id': 'O75387', 'xref_name': None, 'xre...	Homo sapiens	L-type amino acid transporter 3	17.0	False	CHEMBL4148	[{'accession': 'O75387', 'component_descriptio...	SINGLE PROTEIN	9606.0
2	[]	Rattus norvegicus	Prostate	17.0	False	CHEMBL613656	[]	TISSUE	10116.0
3	[{'xref_id': 'P07288', 'xref_name': None, 'xre...	Homo sapiens	Prostate specific antigen	16.0	False	CHEMBL2099	[{'accession': 'P07288', 'component_descriptio...	SINGLE PROTEIN	9606.0
4	[]	Homo sapiens	Prostate cells	15.0	False	CHEMBL614850	[]	CELL-LINE	9606.0
...	...	...	...	...	...	...	...	...	...
70	[]	Homo sapiens	Mucin-1	4.0	False	CHEMBL3580494	[{'accession': 'P15941', 'component_descriptio...	SINGLE PROTEIN	9606.0
71	[]	Homo sapiens	Interleukin 13 receptor	4.0	False	CHEMBL3831285	[{'accession': 'P24394', 'component_descriptio...	PROTEIN COMPLEX	9606.0
72	[]	Homo sapiens	Aurora kinase A/Targeting protein for Xklp2	3.0	False	CHEMBL3883304	[{'accession': 'O14965', 'component_descriptio...	PROTEIN COMPLEX	9606.0

Figure A.2: ChEMBL Dataset

This column likely contains identifiers for the chemical compound in other databases.
organism: This column specifies the organism the compound targets.
pref-name: This column contains the preferred name of the chemical compound.
score: This column contains a score, possibly indicating the relevance of the compound to a specific target.
species: This column provides more specific information about the organism the compound targets.
group-flag: This column likely indicates whether the target is a group of related molecules or a single molecule.
target-chembl-id: This column contains an identifier for the target molecule in the ChEMBL database, a database of bioactive molecules.
target-components: This column lists the components of the target molecule.
target-type: This column indicates the type of the target molecule.
tax-id: This column contains the taxonomic ID of the organism the compound targets.

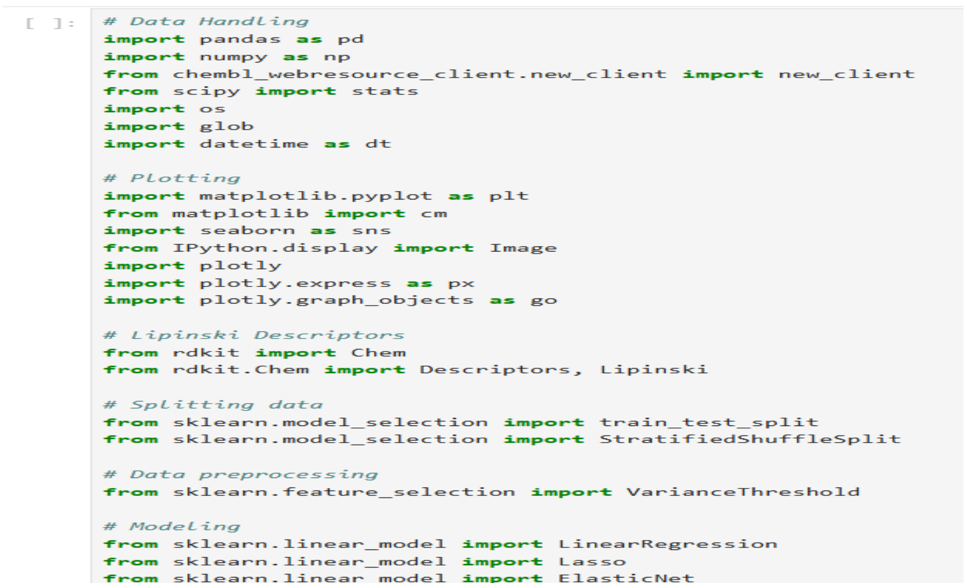
Predicting Drug-Target Binding Affinity
The data utilized in this task was generated by the IDG-Kinase group with funding from the NIH Common Fund. It can be orginally accessed through a synapse account from the website here DTC Dataset.
compound_id: This column contains a unique identifier for each compound in the dataset.
standard_inchi_key: This column contains a unique identifier for the chemical structure of each compound using the InChIKey standard. InChI stands for International Chemical Identifier.
target_id: This column contains an identifier for the biological target of each compound.
gene_names: This column contains a list of gene names associated with the target of each compound.
wildtype_or_mutant: This column indicates whether the target is a wildtype (normal) protein or a mutant protein.
standard_type: This column describes the type of standard measurement used in the dataset. In this case, all entries are "KDapp", which likely refers to the dissociation constant (Kd) apparent affinity.
standard_relation: This column describes the relationship between the standard value and the target molecule. In this case, all entries are "=" indicating a measured value.
standard_value: This column contains the actual measured value, presumably the dissociation constant (Kd) for the interaction between the compound and its target.
standard_units: This column specifies the units of the standard value. In this case, all entries are "nM" for nanomolar.



	compound_id	standard_inchi_key	target_id	gene_names	wildtype_or_mutant	standard_type	standard_relation	standard_value	standard_units
0	CHEMBL3545284	NaN	Q9Y4K4	MAP4K5	NaN	KDAPP	=	19155.14	NM
1	CHEMBL3545284	NaN	Q9Y478	PRKAB1	NaN	KDAPP	=	1565.72	NM
2	CHEMBL3545284	NaN	Q9Y2U5	MAP3K2	NaN	KDAPP	=	746.77	NM
3	CHEMBL3545284	NaN	Q9Y2K2	SIK3	NaN	KDAPP	=	13558.67	NM
4	CHEMBL3545284	NaN	Q9UL54	TAOK2	NaN	KDAPP	=	2220.98	NM

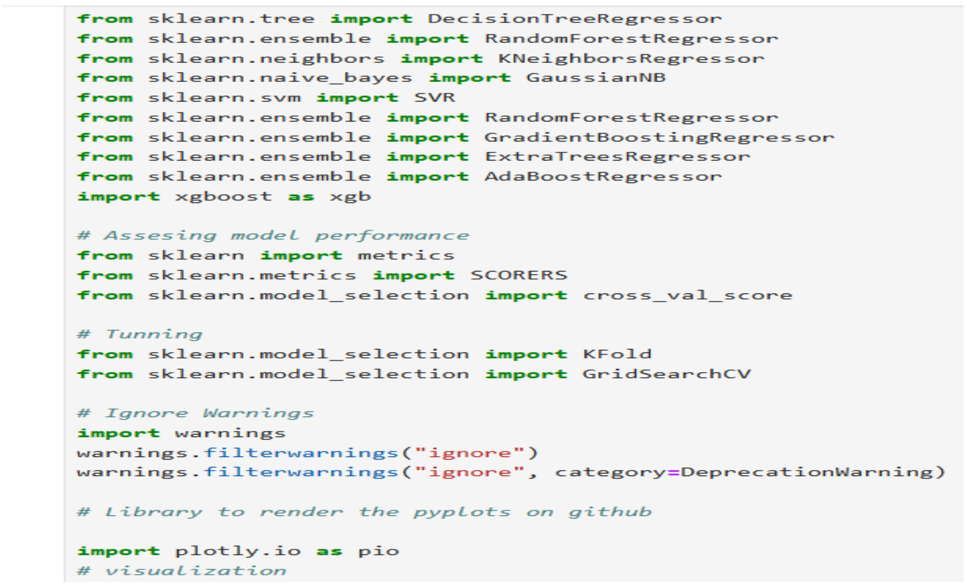
Figure A.3: DTC dataset

A.2 Packages



```
[ ]: # Data Handling
     import pandas as pd
     import numpy as np
     from chembl_webresource_client.new_client import new_client
     from scipy import stats
     import os
     import glob
     import datetime as dt

     # Plotting
     import matplotlib.pyplot as plt
     from matplotlib import cm
     import seaborn as sns
     from IPython.display import Image
     import plotly
     import plotly.express as px
     import plotly.graph_objects as go

     # Lipinski Descriptors
     from rdkit import Chem
     from rdkit.Chem import Descriptors, Lipinski

     # Splitting data
     from sklearn.model_selection import train_test_split
     from sklearn.model_selection import StratifiedShuffleSplit

     # Data preprocessing
     from sklearn.feature_selection import VarianceThreshold

     # Modeling
     from sklearn.linear_model import LinearRegression
     from sklearn.linear_model import Lasso
     from sklearn.linear_model import ElasticNet
```

Figure A.4: Importing all the necessary libraries and packages for ML Model



```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost as xgb

# Assesing model performance
from sklearn import metrics
from sklearn.metrics import SCORERS
from sklearn.model_selection import cross_val_score

# Tunning
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)

# Library to render the pyplots on github

import plotly.io as pio
# visualization
```

Figure A.5: Importing all the necessary libraries and packages for ML Model

Numpy : Numpy is a powerful numerical computing library for Python. It provides a wide range of functions for working with arrays, matrices, and other numerical data.
Pandas : Pandas is a powerful data manipulation and analysis library for Python. It provides a wide range of tools for loading, cleaning, transforming, analyzing, and visualizing data.
Logging: Logging is a critical aspect of software development, enabling the recording of events, errors, and other relevant information during the execution of a program. In drug target affinity projects, logging helps track the progress of data processing, model training, and evaluation.

RDKit: RDKit is a comprehensive cheminformatics library in Python widely used in drug discovery and computational chemistry. It provides a vast array of tools for handling, analyzing, and visualizing chemical structures and data. RDKit allows for tasks such as molecular fingerprinting, substructure searching, 2D and 3D molecular visualization, molecular similarity calculations, and pharmacophore modeling. In drug target affinity projects, RDKit is indispensable for tasks ranging from compound library screening to ligand-receptor docking and lead optimization.
ElmoEmbedder: ElmoEmbedder is a Python library for leveraging pre-trained ELMo models in natural language processing tasks. It's valuable in drug target affinity projects for analyzing textual data like biomedical literature or drug descriptions to extract features for predictive modeling or knowledge extraction.
Matplotlib: Matplotlib is a popular Python library used for creating static, interactive, and publication-quality plots. It provides a wide range of plotting functionalities for visualizing data in various formats, including line plots, scatter plots, histograms, bar charts, and more. In drug target affinity projects, Matplotlib can be utilized to visualize molecular structures, plot distribution of molecular properties, display model performance metrics, and illustrate relationships between molecular features and binding affinities.
Scikit-learn: Scikit-learn, often abbreviated as sklearn, is a powerful machine learning library in Python. It provides simple and efficient tools for data mining and data analysis, including classification, regression, clustering, dimensionality reduction, and model selection. In drug target affinity projects, scikit-learn is indispensable for building predictive models to estimate binding affinities between molecules and targets, performing feature selection, and evaluating model performance through cross-validation and other techniques.
Seaborn: Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations such as scatter plots, bar plots, box plots, heatmaps, and more. In drug target affinity projects, Seaborn can be used to visualize the distribution of molecular properties, analyze relationships between features, and present model performance metrics in a visually appealing manner.
Pickle: The pickle module in Python provides a way to serialize and deserialize Python objects, allowing them to be saved to a file and later reconstructed. In drug target affinity projects, pickle can be useful for saving trained machine learning models, molecular data structures, or other complex objects to disk, allowing for easy storage and retrieval for later use without the need for retraining or recalculating.
ProteinBERTModel: ProteinBERTModel is a model developed by the DeepMind team that extends the BERT (Bidirectional Encoder Representations from Transformers) architecture to the domain of protein sequences. It's trained on large-scale protein sequence data and can generate embeddings for protein sequences, enabling various downstream tasks such as protein function prediction, protein-protein interaction prediction, and drug target affinity prediction. In drug target affinity projects, ProteinBERTModel is valuable for extracting features from protein sequences and predicting their interactions with drug molecules.