

Drug Repurposing for Neurological Disorder - A Computational Approach using RDKit Cheminformatics Tool

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Science in Bioinformatics and Data Science

BY

S. SOWNDHARYA (41738015)



**DEPARTMENT OF BIOINFORMATICS
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A++" by NAAC | Category-1 University by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC | Category-1 University by UGC |
Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

CERTIFICATE

This is to certify that the dissertation titled **"Drug Repurposing for Neurological Disorder - A Computational Approach using RDKit Cheminformatics Tool"** was carried out by **Ms. S. SOWNDHARYA (Reg No. 41738015)**, Department of Bioinformatics and Data Science, Sathyabama Institute of Science and Technology, Chennai in partial fulfillment of the requirements of the award of the **Degree of Bachelor of Science in Bioinformatics and Data Science 2021-2024**, is a bonafide record of the project work done under my guidance and supervision. This thesis or part, therefore, has not been submitted elsewhere for any other degree or diploma.

Date: 27/04/24

Place: Chennai

A handwritten signature in black ink, consisting of a stylized 'J' followed by a horizontal line and a small flourish.

Dr. H. Jemmy Christy

ACKNOWLEDGEMENTS

I am delighted to express my heartfelt acknowledgement to the **BOARD OF MANAGEMENT OF SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement and support throughout the duration of my project. Their guidance has been invaluable in ensuring the successful completion of this project, and I am sincerely grateful for their contribution.

I convey my thanks to **Dr. Rekha Chakravarthy, Dean, School of Science and Humanities** for providing me necessary support and details at the right time during the progressive reviews.

I sincerely thank **Dr. H. Jemmy Christy, Head of the Department, Dept. of Bioinformatics**, for her unwavering support and timely provision of necessary details during the progressive reviews. Her support has been instrumental in helping me to overcome obstacles and achieve success in this project.

I am deeply indebted to my **Project Guide, Dr. H. Jemmy Christy**, for her invaluable guidance, suggestions, and constant encouragement throughout the project. Her expertise and insights have been instrumental in shaping my work and ensuring its successful completion.

I also wish to extend my gratitude to my family for their support and assistance in various ways, which have contributed to the successful completion of this project.

S. SOWNDHARYA



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC I Category-1 University by UGC I
Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF BIOINFORMATICS

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
(41738015), who carried out the final year project entitled "Drug Repurposing
for Neurological Disorder - A Computational Approach using RDKit
Cheminformatics Tool" under our supervision from January 2024 to April 2024.

Dr. H. JEMMY CHRISTY

Internal Guide

Dr. H. JEMMY CHRISTY

Head of the Department

Submitted for Viva Voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **S. SOWNDHARYA**, hereby declare that the Project Report entitled **Drug Repurposing for Neurological Disorder - A Computational Approach using RDKit Cheminformatics Tool** one by me under the guidance of **Dr. H. JEMMY CHRISTY** at Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Rajiv Gandhi Road, Chennai-600119, Tamil Nadu, India, is submitted in partial fulfillment of the requirements for the award of **Bachelor of Science in Bioinformatics and Data Science**.

DATE: 27.04.2024

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ABSTRACT

Neurological disorders pose significant healthcare challenges due to their complex nature and limited treatment options. Drug repurposing emerges as a promising strategy to expedite the discovery of new therapies by exploring existing drugs for novel therapeutic indications. Using computational approaches, particularly utilizing the RDKit cheminformatics tool, this study aims to identify potential drug candidates for neurological disorders through repurposing efforts. Addressing the urgent need for alternative therapeutic strategies, this research employs computational methods to repurpose existing drugs for neurological indications. While previous studies have highlighted the potential of computational tools in drug discovery, the specific application of RDKit in identifying drug candidates for neurological disorders remains relatively unexplored. Through a systematic computational approach utilizing RDKit, encompassing molecular descriptors, molecular fingerprinting, and similarity searching algorithms, we conducted comprehensive analysis of existing drugs. From this analysis, we identified several compounds with promising repurposing potential for neurological disorders. Notable identified compounds include Metoprolol, Fexofenadine, Loratadine, Loperamide, Clopidogrel, Lorcaserin, Tamsulosin, Nebivolol, Salmeterol, Cyclizine, Cromolyn, Dimenhydrinate, Meclizine, Scopolamine, and Nifedipine. These compounds exhibit molecular properties and structural similarities indicative of latent therapeutic potential, thereby laying the groundwork for further investigation and validation. The findings of this study underscore the effectiveness of computational methods, particularly RDKit, in identifying repurposable drugs for neurological disorders.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF ABBREVIATION	IV
	LIST OF FIGURES	V
	LIST OF TABLES	VII
1	INTRODUCTION	1
	1.1 BACKGROUND AND SIGNIFICANCE	1
	1.1.1 CHALLENGES IN TRADITIONAL DRUG DISCOVERY APPROACHES	2
	1.1.2 RATIONALE BEHIND DRUG REPURPOSING AS A VIABLE STRATEGY	2
	1.2 COMPUTATIONAL APPROACHES IN DRUG DISCOVERY	3
	1.2.1 IMPORTANCE OF CHEMINFORMATICS TOOLS FOR ANALYSING MOLECULAR STRUCTURES AND PROPERTIES	4
	1.2.2 ADVANTAGES OF USING COMPUTATIONAL TECHNIQUES FOR DRUG REPURPOSING	5
	1.3 RDKit CHEMINFORMATICS TOOL	5
	1.3.1 MOLECULAR DESCRIPTORS	6
	1.3.2 MOLECULAR FINGERPRINTING	6
	1.3.3 SIMILARITY SEARCHING ALGORITHMS	7
	1.4 EXISTING DRUGS	7
2	REVIEW OF LITERATURE	9
3	AIM AND OBJECTIVES	11
4	METHODOLOGY	12
	4.1 DATA COLLECTION	12
	4.2 DATABASES	12
	4.3 IMPORTING LIBRARIES	14
	4.4 FEATURE EXTRACTION	15
	4.4.1 COMPUTING MOLEUCULAR DESCRIPTORS	15
	4.4.2 MOLECULAR FINGERPRINT GENERATION	16

	4.4.3 SIMILARITY SEARCHING	16
5	RESULT AND DISCUSSION	18
	5.1 RESULTS	18
	5.2 DISCUSSION	19
	5.3 RESULTS OF MOLECULAR DESCRIPTOR	21
	5.4 RESULTS OF MOLECULAR FINGERPRINTING	25
	5.5 RESULTS OF SIMILARITY SEARCHING	29
6	SUMMARY AND CONCLUSION	50
	6.1 SUMMARY	50
	6.2 CONCLUSION	51
	REFERENCES	52

LIST OF ABBREVIATION

RDKit	Reaction Discovery Toolkit
QSAR	Quantitative Structure-Activity Relationship
BBB	Blood-Brain Barrier
ND	Neurological Disorders
CSV	Comma Separated Value
NCBI	National Center for Biotechnology Information
SMILES	simplified molecular-input line-entry system
LogP	Partition Coefficient
TPSA	Topological Polar Surface Area
ADME	Absorption, Distribution, Metabolism, and Excretion
MACCS	Molecular ACCess System
FDA	Food and Drug Administration

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
5.1	Existing drugs of neurological disorder (1-30)	19
5.2	Existing drugs of neurological disorder (31-60)	19
5.3	Existing drugs of neurological disorder (61-82)	20
5.4	Installing RDKit in a Jupyter Notebook environment	20
5.5	Function to calculate descriptors for a molecule	21
5.6	Loading data from CSV file (1)	21
5.7	Adding columns for descriptors	22
5.8	Calculating descriptors for each compound	22
5.9	Saving the updated DataFrame to CSV (1)	23
5.10	Descriptors output (1)	23
5.11	Descriptors output (2)	24
5.12	Descriptors output (3)	24
5.13	Function to calculate Morgan fingerprints for a molecule (1)	25
5.14	Loading data from CSV file (2)	25
5.15	Adding column for Morgan fingerprints (1)	26
5.16	Calculating Morgan fingerprints for each compound (1)	26
5.17	Saving the updated DataFrame to CSV (2)	27
5.18	Molecular fingerprinting output (1)	27
5.19	Molecular fingerprinting output (2)	28
5.20	Molecular fingerprinting output (3)	28
5.21	Molecular fingerprinting output (4)	29
5.22	Function to calculate Morgan fingerprints for a molecule (2)	29
5.23	Loading data from CSV (3)	30
5.24	Adding column for Morgan fingerprints (2)	30
5.25	Calculating Morgan fingerprints for each compound (2)	31
5.26	Function to calculate Tanimoto similarity	31
5.27	Defining a function for similarity searching with threshold	32
5.28	Potential Drug candidate – Metoprolol	32
5.29	Potential Drug candidate – Metoprolol	33
5.30	Performing Similarity searching – Fexofenadine	34

5.31	Potential Drug candidate – Fexofenadine	34
5.32	Performing Similarity searching – Loratadine	35
5.33	Potential Drug candidate – Loratadine	35
5.34	Performing Similarity searching – Loperamide	36
5.35	Potential Drug candidate – Loperamide	36
5.36	Performing Similarity searching – Clopidogrel	37
5.37	Potential Drug candidate – Clopidogrel	37
5.38	Performing Similarity searching – Lorcaserin	38
5.39	Potential Drug candidate – Lorcaserin	38
5.40	Performing Similarity searching – Tamsulosin	39
5.41	Potential Drug candidate – Tamsulosin	39
5.42	Performing Similarity searching – Nebivolol	40
5.43	Potential Drug candidate – Nebivolol	40
5.44	Performing Similarity searching – Salmeterol	41
5.45	Potential Drug candidate – Salmeterol	41
5.46	Performing Similarity searching – Cyclizine	42
5.47	Potential Drug candidate – Cyclizine	42
5.48	Performing Similarity searching – Cromolyn	43
5.49	Potential Drug candidate – Cromolyn	43
5.50	Performing Similarity searching – Dimenhydrinate	44
5.51	Potential Drug candidate – Dimenhydrinate	44
5.52	Performing Similarity searching – Meclizine	45
5.53	Potential Drug candidate – Meclizine	45
5.54	Performing Similarity searching – Scopolamine	46
5.55	Potential Drug candidate – Scopolamine	46
5.56	Performing Similarity searching – Nifedipine	47
5.57	Potential Drug candidate – Nifedipine	

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
5.1	Potential drug candidates with indication	49

CHAPTER 1

INTRODUCTION

Neurological disorders encompass a broad spectrum of conditions affecting the central and peripheral nervous systems, including the brain, spinal cord, and nerves. These disorders, such as Alzheimer's disease, Parkinson's disease, multiple sclerosis, epilepsy, and stroke, pose significant challenges to individuals, families, and healthcare systems worldwide (*WHO, 2020*). With an aging population and increasing life expectancy, the prevalence of neurological disorders is on the rise, placing a growing burden on healthcare resources and infrastructure (*GBD 2019 Diseases and Injuries Collaborators, 2020*).

1.1 BACKGROUND AND SIGNIFICANCE

Neurological disorders are often chronic and debilitating, leading to cognitive impairment, motor dysfunction, sensory deficits, and other debilitating symptoms that profoundly impact quality of life (*National Institute of Neurological Disorders and Stroke [NINDS], 2020*). These conditions not only result in substantial healthcare expenditures but also impose indirect costs through loss of productivity and caregiving burden (*Feigin et al., 2019*).

The complexity of neurological disorders presents unique challenges in diagnosis, treatment, and management, requiring interdisciplinary approaches and innovative therapeutic strategies (*GBD 2016 Neurology Collaborators, 2019*).

There is an urgent need for the development of effective therapies to alleviate symptoms, slow disease progression, and improve outcomes for individuals living with neurological disorders. However, traditional drug discovery approaches are often time-consuming, expensive, and prone to failure, necessitating novel strategies to expedite the identification and development of therapeutics.

Drug repurposing has emerged as a promising strategy to accelerate the discovery of new treatments by leveraging existing drugs for alternative indications, offering potential cost and time savings while maximizing therapeutic impact (*Ashburn & Thor, 2004*).

1.1.1 CHALLENGES IN TRADITIONAL DRUG DISCOVERY APPROACHES

Traditional drug discovery processes are fraught with numerous challenges that impede the efficient identification and development of novel therapeutics. One significant challenge arises from the high attrition rates associated with preclinical and clinical trials, where the majority of candidate compounds fail to demonstrate efficacy or safety profiles suitable for therapeutic use (*Paul et al., 2010*).

The conventional approach to drug discovery often relies on target-based screening, wherein candidate compounds are designed to interact with specific molecular targets implicated in disease pathology. However, this approach overlooks the complex interplay of biological systems and may fail to account for off-target effects or unintended consequences (*Kola & Landis, 2004*).

The time-consuming and costly nature of traditional drug discovery, coupled with increasing regulatory requirements, exacerbates the financial risks associated with bringing new drugs to market (*DiMasi et al., 2016*). These challenges underscore the need for alternative strategies to streamline the drug discovery process and improve the efficiency of therapeutic development.

1.1.2 RATIONALE BEHIND DRUG REPURPOSING AS A VIABLE STRATEGY

Drug repurposing, also known as drug repositioning or drug reprofiling, has gained traction as a promising strategy in pharmaceutical research and development due to several compelling reasons. Firstly, repurposing existing drugs for new therapeutic indications offers a more efficient and cost-effective approach compared to traditional drug discovery methods (*Pushpakom et al., 2019*). Since repurposed drugs have already undergone extensive preclinical and clinical testing for their original indications, the safety profiles, pharmacokinetics, and toxicology profiles are well-characterized, significantly reducing the time and resources required for development (*Ashburn & Thor, 2004*).

Repurposing can breathe new life into shelved or failed drug candidates, maximizing the value of existing pharmaceutical assets and mitigating the risks associated with developing entirely novel compounds (*Nosengo, 2016*). Drug repurposing capitalizes on the vast amount of existing biomedical knowledge and data,

leveraging insights from diverse fields such as systems biology, pharmacology, and clinical medicine (*Corsello et al., 2017*).

By systematically exploring the relationships between drug molecules and biological targets, researchers can identify novel therapeutic uses for existing drugs based on shared mechanisms of action or pathways implicated in different diseases (*Dudley et al., 2011*).

This knowledge-driven approach enables the rapid identification of promising candidates with the potential for clinical translation, accelerating the pace of therapeutic innovation and addressing unmet medical needs in a timely manner (*Ashburn & Thor, 2004*).

Drug repurposing aligns with the principles of precision medicine and personalized healthcare by facilitating targeted therapies tailored to individual patients' needs (*Hurle et al., 2013*).

By repurposing drugs with known efficacy and safety profiles, clinicians can optimize treatment regimens, minimize adverse effects, and improve patient outcomes while reducing healthcare costs (*Langedijk et al., 2015*). Repurposing can uncover unexpected therapeutic benefits or synergistic effects, leading to serendipitous discoveries and novel treatment paradigms (*Kaltenbach et al., 2007*).

1.2 COMPUTATIONAL APPROACHES IN DRUG DISCOVERY

Computational methods play essential role in accelerating drug discovery processes by enabling the rapid and efficient analysis of vast amounts of chemical and biological data (*Proudfoot, 2002*).

These methods encompass a wide range of techniques, including molecular modelling, virtual screening, machine learning, and network analysis, which leverage computational algorithms to predict and optimize drug-target interactions, identify lead compounds, and elucidate drug mechanisms of action (*Schneider & Fechner, 2005*).

The power of computational approaches, researchers can expedite the identification of potential drug candidates, prioritize compounds for experimental validation, and

optimize molecular structures for enhanced efficacy and safety (Kitchen *et al.*, 2004).

Computational methods facilitate the integration of diverse data sources, including genomic, proteomic, and chemical datasets, enabling the comprehensive characterization of drug-target interactions and biological pathways implicated in disease pathogenesis (Chen *et al.*, 2020).

Through iterative cycles of computational modelling, experimental validation, and data-driven optimization, computational methods empower researchers to explore vast chemical and biological spaces, accelerating the drug discovery pipeline and facilitating the development of novel therapeutics for unmet medical needs (Hartenfeller *et al.*, 2012).

1.2.1 IMPORTANCE OF CHEMINFORMATICS TOOLS FOR ANALYSING MOLECULAR STRUCTURES AND PROPERTIES

Cheminformatics tools play a crucial role in drug discovery and development by facilitating the systematic analysis of molecular structures and properties, thereby enabling researchers to make informed decisions at various stages of the drug discovery pipeline (Oprea & Gottfries, 2001).

These tools encompass a diverse array of software platforms and algorithms designed to handle chemical data, including molecular descriptors, fingerprints, and structural representations, allowing researchers to extract meaningful information from chemical structures (Fourches *et al.*, 2010).

By cheminformatics tools, researchers can predict physicochemical properties, such as solubility, lipophilicity, and bioavailability, which are critical determinants of a compound's drug-like characteristics (Tetko *et al.*, 2005). Cheminformatics tools enable the identification of structural features associated with biological activity, facilitating the rational design of lead compounds and optimization of molecular structures for improved potency and selectivity (Bajorath, 2002).

Through the integration of computational models and experimental data, cheminformatics tools streamline the drug discovery process, guiding researchers

towards promising candidates and reducing the time and resources required for experimental validation (*Koutsoukas et al., 2013*).

1.2.2 ADVANTAGES OF USING COMPUTATIONAL TECHNIQUES FOR DRUG REPURPOSING

Computational techniques for drug repurposing offers several advantages, making it an attractive strategy in pharmaceutical research and development. Firstly, computational methods allow for the rapid screening of large chemical libraries and prediction of potential interactions with target proteins or biological pathways, thereby accelerating the identification of promising candidates (*Dudley et al., 2011*).

Computational techniques enable the exploration of diverse chemical and biological spaces, facilitating the discovery of novel drug-target interactions and unforeseen therapeutic opportunities (*Corsello et al., 2017*).

These methods provide insights into the underlying mechanisms of drug action and disease pathology, guiding rational decision-making in drug repurposing efforts (*Hurle et al., 2013*).

1.3 RDKit CHEMINFORMATICS TOOL

RDKit is a powerful open-source cheminformatics software library widely utilized in drug discovery and development for the analysis and manipulation of chemical structures and data (*Landrum, 2006*).

RDKit: Open-source cheminformatics, <http://www.rdkit.org>

Developed in C++ with Python bindings, RDKit offers a comprehensive suite of tools for molecular informatics, including molecular descriptor calculation, substructure searching, molecular fingerprinting, and molecular similarity analysis (*Landrum, 2006*). These capabilities enable researchers to extract valuable insights from chemical data, such as physicochemical properties, structural features, and biological activities, facilitating various aspects of the drug discovery process (*Wang et al., 2019*).

RDKit provides seamless integration with other computational tools and platforms, making it a versatile and indispensable resource for cheminformaticians, medicinal chemists, and computational biologists alike (Landrum, 2006). By empowering users to perform complex analyses and visualizations of chemical structures and properties, RDKit has become a cornerstone of modern cheminformatics research, driving innovation and accelerating the development of novel therapeutics.

1.3.1 MOLECULAR DESCRIPTORS

Molecular descriptors play a crucial role in characterizing chemical compounds and are essential for various cheminformatics analyses. RDKit offers a comprehensive set of molecular descriptors that encode key physicochemical properties and structural features of molecules (Landrum, 2006).

These descriptors encompass a wide range of molecular properties, including size, shape, polarity, flexibility, and electronic distribution (Todeschini & Consonni, 2009). By calculating molecular descriptors using RDKit, researchers can quantitatively describe and compare chemical compounds, facilitating tasks such as virtual screening, quantitative structure-activity relationship (QSAR) modelling, and molecular diversity analysis.

1.3.2 MOLECULAR FINGERPRINTING

Molecular fingerprinting is a powerful method for encoding structural information of molecules into binary or bit-string representations, facilitating similarity analysis and substructure searching. RDKit provides vigorous support for generating and manipulating molecular fingerprints, enabling users to capture structural features and patterns within chemical compounds (Landrum, 2006).

These fingerprints serve as compact representations of molecular structures, facilitating rapid comparison and clustering of compounds based on structural similarities (Rogers & Hahn, 2010). By molecular fingerprinting algorithms in RDKit, researchers can efficiently search chemical databases, identify structurally similar compounds, and explore chemical spaces for potential drug candidates.

1.3.3 SIMILARITY SEARCHING ALGORITHMS

Similarity searching algorithms are essential tools for identifying compounds with structural similarities to query molecules within chemical databases. RDKit offers a variety of similarity searching algorithms, including Tanimoto similarity and Dice similarity, which leverage molecular fingerprints for efficient searching (Landrum, 2006).

These algorithms enable users to retrieve molecules with similar structural and property profiles, facilitating tasks such as virtual screening, compound clustering, and lead optimization (Landrum, 2006). By employing similarity searching algorithms in RDKit, researchers can expedite the process of identifying potential drug candidates and prioritize compounds for further experimental validation.

1.4 EXISTING DRUGS

Existing drugs for neurological disorders encompass a diverse range of pharmacological agents that target various aspects of neurological function and pathology. For instance, drugs like levodopa and dopamine agonists are commonly prescribed for the symptomatic management of Parkinson's disease, aiming to alleviate motor symptoms by replenishing dopamine levels in the brain (Olanow & Schapira, 2013).

Antiepileptic drugs such as carbamazepine, phenytoin, and valproate are widely used for the treatment of epilepsy, acting on neuronal ion channels to suppress abnormal electrical activity and prevent seizures (Brodie A). Additionally, drugs like donepezil, rivastigmine, and memantine are approved for the treatment of Alzheimer's disease, targeting cholinergic and glutamatergic neurotransmission to improve cognitive function and alleviate symptoms of dementia (Birks, 2006).

While existing drugs provide valuable therapeutic options for neurological disorders, there remains a significant unmet need for more effective treatments with fewer side effects and greater disease-modifying potential. Continued research and innovation in drug discovery and development are essential for addressing these challenges and improving outcomes for individuals living with neurological disorders.

CHAPTER 2

REVIEW OF LITERATURE

Pushpakom *et al.* (2019) conducted a comprehensive review on drug repurposing, highlighting progress, challenges, and recommendations. They discussed the increasing interest in repurposing existing drugs for new therapeutic indications. Challenges such as limited understanding of mechanisms and regulatory hurdles were identified. Recommendations included the need for standardized methods, data sharing, and collaboration to advance drug repurposing efforts. Overall, their review provided valuable insights into the current status and future directions of drug repurposing research.

Corsello *et al.* (2017) developed the Drug Repurposing Hub, a comprehensive drug library and information resource. The Hub facilitates the exploration of drug repurposing opportunities by providing access to a vast collection of compounds with known bioactivities. It serves as a valuable resource for researchers seeking to identify new therapeutic uses for existing drugs. The Hub includes data on drug targets, mechanisms of action, and disease associations, enabling systematic analysis and exploration of drug repurposing hypotheses. Overall, the Drug Repurposing Hub represents a significant advancement in the field of drug repurposing, offering a platform for accelerating therapeutic discovery and development.

Baillargeon *et al.* (2021) explored drug repurposing strategies, applications, and future perspectives specifically in neurological diseases. They reviewed current approaches for repurposing existing drugs to address neurological disorders. The study discussed the potential of repurposed drugs to offer novel therapeutic options for neurological conditions. Various strategies, including computational methods and experimental validation, were examined. The authors provided

insights into the opportunities and challenges in drug repurposing for neurological diseases, offering a roadmap for future research directions.

Kotla *et al.* (2021) investigated the challenges and opportunities in drug repurposing for neurological diseases. They identified key obstacles such as limited understanding of disease mechanisms and the blood-brain barrier (BBB). The study highlighted the potential of repurposed drugs to offer innovative therapeutic solutions for neurological disorders. Recommendations included the need for interdisciplinary collaboration and innovative research approaches.

Patel *et al.* (2021) conducted a comprehensive review on cheminformatics approaches for drug repurposing in ND. They highlighted current trends and future directions in the field, focusing on the application of computational methods to identify new therapeutic uses for existing drugs. The study discussed the challenges and opportunities associated with drug repurposing efforts targeting neurological diseases. Recommendations included the integration of diverse data sources and the development of innovative computational tools for improved drug repurposing outcomes. Overall, their work provided valuable insights into the state-of-the-art techniques and future prospects of drug repurposing for neurological disorders.

Stumpfe and Bajorath (2011) reviewed similarity searching methods in computational molecular science. They discussed the principles and applications of similarity searching for compound retrieval and lead discovery. The review covered various similarity measures and algorithms used in virtual screening. The authors highlighted the importance of structure-based and ligand-based approaches in similarity searching. Overall, their work provided insights into the strategies and challenges associated with similarity searching in drug discovery.

CHAPTER 3

AIM AND OBJECTIVE

3.1 AIM

The aim of this project is to utilize computational techniques, specifically employing RDKit cheminformatics tools, to explore the potential of drug repurposing for neurological disorders.

3.2 OBJECTIVE

1. To compile a comprehensive dataset of existing drugs various neurological disorders.
2. To employ molecular descriptors in RDKit to characterize the chemical properties and structural features of existing drugs used in the treatment of neurological disorders.
3. To generate molecular fingerprints for the compiled dataset of drugs, enabling the identification of structurally similar compounds.
4. To perform similarity searching using RDKit algorithms to identify potential drug candidates with similar structural profiles to existing drugs but with different pharmacological targets.
5. To prioritize candidate compounds based on their predicted pharmacological properties and potential for repurposing as treatments for neurological disorders.
6. To validate the efficacy of selected candidate compounds through in silico and, if feasible, in vitro experiments to assess their potential for clinical translation.

CHAPTER 4

METHODOLOGY

4.1 DATA COLLECTION

For the data collection phase, a comprehensive dataset of existing drugs used in the treatment of neurological disorders was gathered from PubChem, a publicly available database. This dataset, stored in a .csv file, contains drug names and their respective SMILES notations, providing a structured representation of their chemical structures.

Each drug entry is associated with its therapeutic indication, indicating its intended use in treating specific neurological disorders. By collecting data from PubChem, a diverse range of neurological drugs with varied chemical structures and pharmacological profiles was obtained, enabling subsequent analyses and computational modelling.

This dataset serves as a valuable resource, facilitating the exploration of drug repurposing opportunities for neurological disorders using computational techniques and RDKit cheminformatics tools. Through curation and organization of this data, a foundation is established for identifying potential candidate compounds, prioritizing their evaluation, and ultimately advancing therapeutic development in the field of neurology.

4.2 DATABASES

PUBCHEM

PubChem, maintained by the National Center for Biotechnology Information (NCBI), is a vast and freely accessible database containing information on millions of chemical compounds. PubChem serves as a valuable resource for gathering data on existing drugs and their chemical structures. By accessing PubChem's extensive

collection of compounds, we can compile a comprehensive dataset including drug names and their corresponding SMILES notations, which encode the molecular structure of each compound in a concise format. PubChem provides detailed information on the biological activities, pharmacological properties, and therapeutic indications of these compounds, enabling you to identify drugs with potential for repurposing in neurological disorders. The database also offers access to experimental and computational data. Through its user-friendly interface and rich dataset, PubChem facilitates efficient data retrieval, analysis, and visualization, empowering researchers to explore novel therapeutic opportunities and accelerate drug discovery efforts in neurology. Utilizing PubChem as a primary data source enhances the rigor and comprehensiveness of your project, providing a solid foundation for computational analyses and experimental validation.

PUBCHEM :(<https://pubchem.ncbi.nlm.nih.gov/>)

DRUGBANK

DrugBank, an online database curated by the University of Alberta, serves as a valuable resource with its extensive collection of comprehensive drug data. DrugBank offers detailed information on approved drugs, experimental compounds, and investigational agents, making it a pivotal asset in identifying potential candidates for repurposing. By accessing DrugBank's vast repository, we can compile a comprehensive dataset encompassing drug names, chemical structures, therapeutic indications, and pharmacological properties relevant to neurological disorders. Additionally, DrugBank provides comprehensive annotations on drug targets, metabolic pathways, drug-drug interactions, and adverse effects, offering insights into the mechanisms of action and safety profiles of candidate compounds. The database's structured data format and user-friendly interface facilitate efficient data retrieval and analysis, enabling you to identify drugs with known pharmacological targets implicated in neurological disorders. Furthermore, DrugBank's integration with other bioinformatics resources, such as UniProt and PubChem, enhances its utility for drug repurposing efforts by providing access to complementary datasets and enabling cross-referencing of information. Leveraging

DrugBank as a primary data source enriches the rigor and comprehensiveness to explore novel therapeutic opportunities and accelerate the discovery of repurposable drugs for neurological disorders.

DRUGBANK :(<https://go.drugbank.com/>)

4.3 IMPORTING LIBRARIES

- pandas (imported as pd): This library is used for data manipulation and analysis in Python. It provides data structures like DataFrames, which are particularly useful for handling tabular data.
- rdkit: This is a powerful cheminformatics toolkit widely used for molecular modelling, cheminformatics, and bioinformatics. It provides functions and classes for molecular structure handling, molecular descriptors calculation, molecular fingerprinting, molecular similarity calculations, and much more.
- Chem from rdkit: This submodule of RDKit provides functions and classes for working with molecular structures, such as reading and writing molecular files (e.g., SDF, SMILES), generating 2D and 3D molecular structures, manipulating molecules, and calculating molecular properties.
- Descriptors from rdkit.Chem: This submodule contains functions for calculating molecular descriptors, which are quantitative representations of molecular properties used in cheminformatics.
- AllChem from rdkit.Chem: This submodule provides additional functions for molecular modeling and cheminformatics tasks, including molecular conformer generation, substructure searching, and more.

- DataStructs from rdkit.Chem: This submodule contains functions for working with molecular fingerprints and similarity calculations, such as Tanimoto similarity and Jaccard similarity.

4.4 FEATURE EXTRACTION

4.4.1 COMPUTING MOLECULAR DESCRIPTORS

- Molecular Weight, a fundamental property, represents the sum of the atomic weights of all atoms in a molecule and serves as a measure of its mass.
- NumValenceElectrons quantifies the total number of valence electrons present in a molecule, providing insights into its electronic structure and chemical reactivity.
- LogP, a crucial physicochemical parameter, predicts the lipophilicity or hydrophobicity of a compound and influences its absorption, distribution, metabolism, and excretion (ADME) properties.
- TPSA (topological polar surface area), characterizes the polar surface area of a molecule, which correlates with its ability to form hydrogen bonds and interact with biological targets.

By calculating these descriptors, RDKit enables the characterization of the chemical properties and structural features of existing drugs used in the treatment of neurological disorders. These descriptors serve as quantitative representations of molecular structures, facilitating the comparison, classification, and prediction of drug properties relevant to drug repurposing efforts. Moreover, the computed descriptors provide valuable insights into the physicochemical properties and pharmacological profiles of candidate compounds, aiding in the identification and

prioritization of potential drug candidates for further evaluation and experimental validation.

4.4.2 MOLECULAR FINGERPRINT GENERATION

Molecular Fingerprint Generation phase, RDKit's versatile cheminformatics tools are utilized to generate molecular fingerprints for the compiled dataset of drugs, facilitating the identification of structurally similar compounds and enabling virtual screening approaches.

Molecular fingerprints encode structural information in a compact binary or bit-string format, representing the presence or absence of specific structural features within a molecule. RDKit offers various fingerprinting algorithms, such as Morgan fingerprints (also known as circular fingerprints) and MACCS keys, each capturing different aspects of molecular structure and diversity.

These fingerprints are calculated for each drug compound in the dataset, capturing its unique structural characteristics and encoding them into numerical representations suitable for similarity analysis and substructure searching. By employing molecular fingerprints, researchers can efficiently compare the structural similarity between candidate compounds and known drugs used in the treatment of neurological disorders, aiding in the identification of promising repurposing candidates.

4.4.3 SIMILARITY SEARCHING

Similarity Searching phase, RDKit algorithms are employed to perform similarity searching, enabling the identification of potential drug candidates with structural profiles similar to existing drugs but with different pharmacological targets. Using molecular fingerprints generated for the compiled dataset of drugs, similarity searching involves comparing the structural similarity between candidate compounds and known drugs used in the treatment of neurological disorders.

RDKit offers efficient and scalable algorithms for similarity analysis, including Tanimoto coefficient-based methods, which quantify the similarity between two molecular fingerprints by measuring the overlap of their binary features. By applying similarity searching techniques, researchers can systematically explore the chemical space occupied by existing drugs and identify structurally related compounds that may exhibit similar biological activities.

The results of similarity searching provide valuable insights into potential repurposing candidates, prioritizing compounds with structural resemblance to known drugs but distinct pharmacological profiles.

CHAPTER 5

RESULT AND DISCUSSION

5.1 RESULTS

The results of the project align closely with its goal of utilizing computational techniques, particularly employing RDKit cheminformatics tools, to explore drug repurposing opportunities for neurological disorders.

Through comprehensive data compilation, molecular descriptor calculation, molecular fingerprint generation, and similarity searching, a curated dataset of existing drugs used in neurological disorders was systematically analyzed.

This approach enabled the identification of structurally similar compounds with distinct pharmacological targets, laying the groundwork for potential repurposing candidates. By prioritizing compounds based on their predicted pharmacological properties and potential for repurposing.

These results highlight the effectiveness of computational methods in rapidly screening and identifying repurposable drugs, thereby advancing therapeutic interventions in neurology.

5.2 DISCUSSION

File Home Insert Page Layout Formulas Data Review View Help	
A2 X f x	
A	B
1 Neurological drugs	SMILES
2	
3 Abilify (aripiprazole)	<chem>C1CC(=O)NC2=C1C=CC(=C2)OCCCCN3CCN(CC3)C4=C(C(=CC=C4)Cl)Cl</chem>
4 Ampyra (dalfampridine)	<chem>C1=CN=CC=C1N</chem>
5 Amrix (cyclobenzaprine hydrochloride)	<chem>CN(C)CCC=C1C2=CC=CC=C2C=CC=C=C31</chem> Close
6 Apokyn (apomorphine hydrochloride)	<chem>CN1CCC2=C3C1CC4=C(C3=CC=C2)C(=C(C=C4)O)O</chem>
7 Aptiom (eslicarbazepine acetate)	<chem>CC(=O)OC1CC2=CC=CC=C2N(C3=CC=CC=C3)C(=O)N</chem>
8 Aristada (aripiprazole lauroxil)	<chem>C1CC(=O)NC2=C1C=CC(=C2)OCCCCN3CCN(CC3)C4=C(C(=CC=C4)Cl)Cl</chem>
9 Banzel (rufinamide)	<chem>C1=CC(=C(C(=C1)F)CN2C=C(N=N2)C(=O)N)F</chem>
10 Belsomra (suvorexant)	<chem>CC1CCN(CCN1C(=O)C2=C(C(=C2)C)N3N=CC=N3)C4=NC5=C(O4)C=CC(=C5)Cl</chem>
11 Briviact (brivaracetam)	<chem>CCCC1CC(=O)N(C1)C(C)C(=O)N</chem>
12 Carbaglu (carglumic acid)	<chem>C(C(=O)O)C(C(=O)O)NC(=O)N</chem>
13 Cialis (tadalafil)	<chem>CN1CC(=O)N2C(C1=O)CC3=C(C2C4=CC5=C(C(=C4)OCOC5)NC6=CC=CC=C36</chem>
14 Comtan (entacapone)	<chem>CCN(C)C(C(=O)C(=CC1=CC(=C(C(=C1)O)O)[N+](=O)[O-])C#N</chem>
15 Cymbalta (duloxetine)	<chem>CNCCC(C1=CC=CS1)OC2=C(C=CC3=CC=CC=C32.Cl</chem>
16 Diacomit (stiripentol)	<chem>CC(C)C(C(=CC1=CC2=C(C(=C1)OC2)O</chem>
17 Duvia (sufentanil)	<chem>CC(C=O)N(C1=CC=CC=C1)C2(CCN(C2)CCC3=CC=CS3)COC</chem>
18 Epidiolex (cannabidiol)	<chem>CCCCCCC1=CC(=C(C(=C1)O)C2C=C(C(C2)C(=C)C)O</chem>
19 Exelon (rivastigmine)	<chem>CCN(C)C(=O)OC1=CC=CC(=C1)C(C)N(C)C</chem>
20 Fetzima (levomilnacipran)	<chem>CCN(C)C(C(=O)C1(C1)CC1)C2=CC=CC=C2.Cl</chem>
21 Fintepla (fenfluramine)	<chem>CCNC(C)CC1=CC(=CC=C1)C(F)F</chem>
22 Firdapse (amifampridine)	<chem>C1=CN=CC(=C1)N.N.OP(=O)(O)O</chem>
23 Focalin, Focalin XR (dexamethylphenidate HCl)	<chem>COC(=O)C(C1CCCC1)C2=C(C=CC=C2.Cl</chem>
24 Frova (frovatriptan succinate)	<chem>CN1CCC2=C(C1)C3=C(N2)C=CC(=C3)C(=O)N.C(C(=O)O)C(=O)O.O</chem>
25 Fycompa (perampanel)	<chem>C1=CC(=C(C(=C1)N2C=C(C(=C2)O)C3=CC=CC=C3C#N)C4=CC=CC=N4</chem>
26 Gocovri (amantadine)	<chem>C1C2CC3CC1CC(C2)C3N</chem>
27 Gralise (gabapentin)	<chem>C1CCC(C(C1)C(=O)O)CN</chem>
28 Hetlioz (tasimelteon)	<chem>CCC(=O)NCC1CC1C2=C3CCOC3=CC=C2</chem>
29 Imitrex (sumatriptan)	<chem>CNS(=O)(=O)CC1=CC2=C(C(=C1)NC=C2CCN(C)C</chem>
30 Ingrezza (valbenazine)	<chem>CC(C)C1CN2CCC3=CC(=C(C(=C3)C2C1OC(=O)C(C(C)C)N)OC)OC</chem>

Figure 5.1 Existing drugs of neurological disorder (1-30)

A	B
31 Intuniv (guanfacine)	<chem>C1=CC(=C(C(=C1)Cl)CC(=O)N=C(N)N)Cl.Cl</chem>
32 Invega (paliperidone)	<chem>CC1=C(C(=O)N2CCCC(C2=N1)O)CN3CCC(C3)C4=NC5=C4C=CC(=C5)F</chem>
33 Keppra (levetiracetam)	<chem>CCC(C(=O)N)N1CCCC1=O</chem>
34 Lamictal (lamotrigine)	<chem>C1=CC(=C(C(=C1)Cl)C)C2=C(N=C(N2)N)N</chem>
35 Lunesta (eszopiclone)	<chem>CN1CCN(C1)C(=O)OC2C3=NC=CC=C3C(=O)N2C4=NC(=C(C4)Cl</chem>
36 Lyrica (pregabalin)	<chem>CC(C)CC(C(=O)O)CN</chem>
37 Maxalt (rizatriptan benzoate)	<chem>CN(C)CCC1=CC2=C1C=C(C(=C2)CN3C=NC=N3.C1=CC=C(C(=C1)C(=O)O</chem>
38 Migranal (dihydroergotamine mesylate)	<chem>C1(C(=O)N2C(C(=O)N3CCCC3C2(O1)O)CC4=CC=CC=C4)NC(=O)C5CC6C(C7=C7CNC8=CC=CC6=C78)N(C5)C.CS(=O)(=O)O</chem>
39 Mirapex (pramipexole dihydrochloride)	<chem>CCCN1CC2=C(C1)SC(=N2)N.O.Cl.Cl</chem>
40 Movantik (naloxegol)	<chem>COCOCOCOCOCOCOCOCOCOC1CCC2(C3CC4=C5C2C1OC5=C(C(=C4)O)CCN3CC=C)O</chem>
41 Namenda (memantine HCl)	<chem>CC12CC3CC(C1)CC(C3)C2)N)Cl</chem>
42 Nayzilam (midazolam)	<chem>CC1=NC=C2N1C3=C(C(=C(C(=C3)C)C(=NC2)C4=CC=CC=C4F</chem>
43 Neupro (Rotigotine)	<chem>CCCN(CCC1=CC=CS1)C2CCC3=C(C2)C=CC=C3O</chem>
44 Neurontin (gabapentin)	<chem>C1CCC(C(C1)CC(=O)O)CN</chem>
45 Northera (droxidopa)	<chem>C1=CC(=C(C(=C1)C(C(=O)O)N)O)O</chem>
46 Nourianz (istradefylline)	<chem>CN1C2=C(C(=O)N(C1=O)CC)N(C(=N2)C=CC3=CC(=C(C(=C3)OC)OC)C</chem>
47 Nucynta (tapentadol)	<chem>CCC(C1=CC(=CC=C1)O)C(C)N(C)C.Cl</chem>
48 Nuedexta (dextromethorphan hydrobromide and quinidine sulfate)	<chem>COC1=CC2=C(C(=CN=C2C1)C(C3CC4CCN3CC4=C)O.COC1=CC2=C(C(=CN=C2C=C1)C(C3CC4CCN3CC4=C)O.O.O.OS(=O)(=O)O</chem>
49 Nuplazid (pimavanserin)	<chem>CC(C)COC1=CC=C(C(=C1)CNC(=O)N(C2=CC=C(C(=C2)F)C3CCN(C3)C</chem>
50 Nuvigil (armodafinil)	<chem>C1=CC=C(C(=C1)C(C2=CC=CC=C2)S(=O)(CC(=O)N</chem>
51 Onfi (clobazam)	<chem>CN1C(=O)CC(=O)N(C2=C1C=CC(=C2)Cl)C3=CC=CC=C3</chem>
52 Oxecta (oxycodone HCl)	<chem>CN1CCC23C4(=O)CCC2(C1CC5=C3C(=C(C(=C5)OC)O4)O.Cl</chem>
53 Qelbree (viloxazine)	<chem>CCOC1=CC=CC=C1OCC2CNCCO2.Cl</chem>
54 Qutenza (capsaicin)	<chem>CC(C)C=CCCCC(=O)NCC1=CC(=C(C(=C1)O)OC</chem>
55 Radicava (edaravone)	<chem>CC1=NN(C(=O)C1)C2=CC=CC=C2</chem>
56 Relpax (eletriptan hydrobromide)	<chem>CN1CCCC1CC2=CC3=C2C=C(C(=C3)CS(=O)(=O)C4=CC=CC=C4.Br</chem>
57 Reminyl (galantamine hydrobromide)	<chem>CN1CCC23C=CC(C2OC4=C(C(=CC(=C34)C1)OC)O.Br</chem>
58 Reyvow (lasmiditan)	<chem>CN1CCC(C1)C(=O)C2=NC(=CC=C2)NC(=O)C3=C(C(=C(C3)F)F</chem>
59 Rilutek (riluzole)	<chem>C1=CC2=C(C(=C1OC(F)F)F)SC(=N2)N</chem>
60 Rozerem (ramelteon)	<chem>CCC(=O)NCCC1CCC2=C1C3=C(C(=C2)OCC3</chem>

Figure 5.2 Existing drugs of neurological disorder (31-60)

61	Sabril (vigabatrin)	<chem>C=CC(CCC(=O)O)N</chem>
62	Silenor (doxepin)	<chem>CN(C)CC=C1C2=CC=CC=C2COC3=CC=CC=C31.Cl</chem>
63	Sonata (zaleplon)	<chem>CCN(C1=CC=CC(=C1)C2=CC=CC=NC3=C(C(=NN23)C#N)C(=O)C</chem>
64	Spravato (esketamine)	<chem>CNC1(CCCCC1=O)C2=CC=CC=C2Cl</chem>
65	Sprix (ketorolac tromethamine)	<chem>C1CN2C(=CC=C2C(=O)C3=CC=CC=C3)C1C(=O)O.C(C(CO))(CO)N)O</chem>
66	Stavzor (valproic acid)	<chem>CCCC(CCC)C(=O)O</chem>
67	Strattera (atomoxetine HCl)	<chem>CC1=CC=CC=C1OC(CCN(C)C2=CC=CC=C2.Cl</chem>
68	Subsys (fentanyl sublingual spray)	<chem>CCC(=O)N(C1CCN(C1)CCC2=CC=CC=C2)C3=CC=CC=C3</chem>
69	Tegretol, Tegretol XR (carbamazepine)	<chem>C1=CC=C2C(=C1)C=CC3=CC=CC=C3N2C(=O)N</chem>
70	Tivorbex (indomethacin)	<chem>CC1=C(C2=C(N1C(=O)C3=CC=C(C=C3)C)C=CC(=C2)OC)CC(=O)O</chem>
71	Topamax (topiramate)	<chem>CC1(OC2COC3(C(C2O1)OC(O3)C(C)COC(=O)C(=O)N)C</chem>
72	Trileptal (oxcarbazepine)	<chem>C1C2=CC=CC=C2N(C3=CC=CC=C3C1=O)C(=O)N</chem>
73	Trudhesa (dihydroergotamine mesylate)	<chem>CC1(C(=O)N2C(C(=O)N3CCCC3C2(O1)O)CC4=CC=CC=C4)NC(=O)C5CC6C(C7=CNC8=CC=CC=C8)N(C5)C.CS(=O)C(=O)O</chem>
74	Ubreelvy (ubrogepant)	<chem>CC1(C(C(C(=O)N1CC(F)(F)F)NC(=O)C2=CC3=C(C(C4(C3)C5=C(NC4=O)N=CC=C5)N=C2)C6=CC=CC=C6</chem>
75	Ultracet (acetaminophen and tramadol HCl)	<chem>CC(=O)NC1=C(C(=C1)O</chem>
76	Viibryd (vilazodone hydrochloride)	<chem>C1CN(CCN1CCCCC2=CNC3=C2C=C(C(=C3)C#N)C4=CC5=C(C(=C4)OC(=C5)C(=O)N).Cl</chem>
77	Vimpat (lacosamide)	<chem>CC(=O)NC(COC)C(=O)NCC1=CC=CC=C1</chem>
78	Vivlodex (meloxicam)	<chem>CC1=CN=C(C(S1)NC(=O)C2=C(C3=CC=CC=C3S(=O)(=O)N2)CO</chem>
79	Vraylar (cariprazine)	<chem>CN(C)C(=O)NC1CCC(C1)CCN2CCN(C2)C3=C(C(=CC=C3)C)Cl.Cl</chem>
80	Vyvanse (Lisdexamfetamine Dimesylate)	<chem>CC(CCC1=CC=CC=C1)NC(=O)C(CCCCN)N.CS(=O)C(=O)O.CS(=O)C(=O)O</chem>
81	Xadago (safinamide)	<chem>CC(C(=O)N)NCC1=CC=C(C(=C1)OCC2=CC(=CC=C2)F</chem>
82	Zanaflex (tizanidine hydrochloride)	<chem>C1CN=C(N1)NC2=C(C(=CC3=NSN=C32)Cl.Cl</chem>
83	Zipsor (diclofenac potassium)	<chem>C1=CC=C(C(=C1)CC(=O)O)[O-])NC2=C(C(=CC=C2)Cl)Cl.[K+]</chem>
84	Zomig (zolmitriptan)	<chem>CN(C)CCC1=CNC2=C1C=C(C(=C2)CC3COC(=O)N3</chem>
85		

FIGURE 5.3 Existing drugs of neurological disorder (61-82)

- The results include the compilation of information, such as drug names and SMILES notation, for FDA-approved existing drugs used in the treatment of neurological disorders, organized and stored in a .csv file for further analysis and exploration.

```

In [1]: pip install rdkit

Requirement already satisfied: rdkit in c:\users\mukesh\anaconda3\lib\site-packages (2023.9.5)
Requirement already satisfied: numpy in c:\users\mukesh\anaconda3\lib\site-packages (from rdkit) (1.21.5)
Requirement already satisfied: Pillow in c:\users\mukesh\anaconda3\lib\site-packages (from rdkit) (9.0.1)
Note: you may need to restart the kernel to use updated packages.

```

Figure 5.4 Installing RDKit in a Jupyter Notebook environment

5.3 RESULTS OF MOLECULAR DESCRIPTOR

```
In [1]: import pandas as pd
        from rdkit import Chem
        from rdkit.Chem import Descriptors

        # Function to calculate descriptors for a molecule
        def calculate_descriptors(molecule):
            """Calculate molecular descriptors for a given molecule."""
            descriptors = {
                'MolecularWeight': Descriptors.MolWt(molecule),
                'NumValenceElectrons': Descriptors.NumValenceElectrons(molecule),
                'LogP': Descriptors.MolLogP(molecule),
                'TPSA': Descriptors.TPSA(molecule),
                # Add more descriptors as needed
            }
            return descriptors
```

Figure 5.5 Function to calculate descriptors for a molecule

- This code begins by importing the necessary libraries, including Pandas for data manipulation and RDKit for molecular descriptor calculation. A function named `calculate_descriptors` is then defined to compute molecular descriptors for a given molecule.
- Within this function, RDKit's `Descriptors` module is utilized to calculate descriptors such as Molecular Weight, Number of Valence Electrons, LogP (Partition Coefficient), and Topological Polar Surface Area (TPSA).
- These descriptors are stored in a dictionary format and returned by the function.

```
# Load data from CSV file with appropriate encoding
neuro_drugs_file_path = r"C:\Users\Mukesh\OneDrive\Documents\molecular fingerprint.csv"
df = pd.read_csv(neuro_drugs_file_path, encoding='latin1')

# Drop rows with 'nan' values in the 'SMILES' column
df = df.dropna(subset=['SMILES'])

# Convert 'SMILES' column to strings
df['SMILES'] = df['SMILES'].astype(str)
```

Figure 5.6 Loading data from CSV file (1)

- Data is loaded from a CSV file with appropriate encoding, and then rows containing 'nan' values in the 'SMILES' column are dropped, followed by converting the 'SMILES' column to strings.

```
# Add columns for descriptors
descriptor_columns = ['MolecularWeight', 'NumValenceElectrons', 'LogP', 'TPSA']
for column in descriptor_columns:
    df[column] = None
```

Figure 5.7 Adding columns for descriptors

- In this code segment, new columns for molecular descriptors are added to a DataFrame (**df**) using a loop over a list of descriptor column names.
- The list **descriptor_columns** specify the names of the descriptors to be added, including Molecular Weight, Number of Valence Electrons, LogP (Partition Coefficient), and Topological Polar Surface Area (TPSA).
- Within the loop, each descriptor column is initialized with 'None' values, effectively creating placeholders for the descriptor data. This step prepares the DataFrame for subsequent population with calculated descriptor values, facilitating the integration of molecular property information into the dataset.

```
# Calculate descriptors for each compound
for index, row in df.iterrows():
    smiles = row['SMILES']
    molecule = Chem.MolFromSmiles(smiles)
    if molecule is not None:
        descriptors = calculate_descriptors(molecule)
        for column, value in descriptors.items():
            df.at[index, column] = value
```

Figure 5.8 Calculating descriptors for each compound

- In this code, descriptors are calculated for each compound in the DataFrame (**df**) by iterating over its rows using the **iterrows()** function.
- For each row, the SMILES notation is extracted from the 'SMILES' column, and RDKit's **Chem.MolFromSmiles()** function is utilized to convert the SMILES notation into a molecular representation (RDKit **Mol** object).
- If a valid molecule is generated, the **calculate_descriptors()** function is called to compute molecular descriptors for the molecule. The descriptors, stored in a dictionary format, are then assigned to the corresponding columns in the DataFrame using the **at** method, ensuring that each compound's descriptor values are appropriately populated.

```
# Save the updated DataFrame to CSV
output_csv_path = "output_with_descriptors.csv"
df.to_csv(output_csv_path, index=False)

print("Descriptors calculated and saved successfully!")
```

Descriptors calculated and saved successfully!

Figure 5.9 Saving the updated DataFrame to CSV (1)

- In this code snippet, the updated DataFrame, which now includes calculated molecular descriptors for each compound, is saved to a CSV file using Pandas' `to_csv()` function.
- The output CSV file path is specified as "output_with_descriptors.csv", and the parameter `index=False` is used to prevent the DataFrame index from being included as a separate column in the CSV file.
- Once the DataFrame is successfully saved, a confirmation message is printed to the console, indicating that the descriptors have been calculated and saved successfully.

jupyter output_with_descriptors.csv 18 hours ago Logout

File	Edit	View	Language	current mode
1	Neurological drugs, SMILES, MolecularWeight, NumValenceElectrons, LogP, TPSA			
2	Abilify (aripiprazole), C1CC(=O)NC2=C1C=CC(=C2)OCCCN3CCN(CC3)C4=C(C(=CC=C4)C1)C1, 448.3940000000001, 160, 4.859300000000005, 44.81			
3	Ampyra (dalfampridine), C1=CN=CC=C1N, 94.117, 36, 0.6638000000000001, 38.91			
4	Amrix (cyclobenzaprine hydrochloride), CN(C)CCC=C1C2=CC=CC=C2C3=CC=CC=C3C1, 275.39500000000004, 106, 4.553800000000004, 3.24			
5	Apokyn (apomorphine hydrochloride), CN1CCC2=C3C1CC4=C(C3=CC=C2)C(=C(C=C4)O)O, 267.328, 102, 2.8499000000000017, 43.7			
6	Aptiom (eslicarbazepine acetate), CC(=O)OC1C2=C(C=CC=C2N(C3=CC=CC=C13)C(=O)N, 296.32600000000001, 112, 3.0637000000000008, 72.63			
7	Aristada (aripiprazole lauroxil), C1CC(=O)NC2=C1C=CC(=C2)OCCCN3CCN(CC3)C4=C(C(=CC=C4)C1)C1, 448.3940000000001, 160, 4.859300000000005, 44.81			
8	Banzel (rufinamide), C1=CC(=C(C(=C1)F)CN2C=C(N=N2)C(=O)N)F, 238.19699999999997, 88, 0.7035, 73.8			
9	Belsomra (suvorexant), CC1CCN(CCN1C(=O)C2=C(C(=CC(=C2)C)N3N=CC=N3)C4=NC5=C(O4)C=CC(=C5)C1, 450.93000000000002, 164, 4.111320000000004, 80.29			
10	Briviact (brivaracetam), CCCC1CC(=O)N(C1)C(C)C(=O)N, 212.29299999999998, 86, 0.8989, 63.400000000000006			
11	Carbaglu (carglumic acid), C(CC(=O)O)C(C(=O)O)NC(=O)N, 190.155, 74, -1.0272999999999997, 129.72			
12	Cialis (tadalafil), CN1CC(=O)N2C(C1=O)CC3=C(C2C4=CC5=C(C=C4)OC5)NC6=CC=CC=C36, 389.41100000000001, 146, 2.211300000000005, 74.87			
13	Comtan (entacapone), CCN(CC)C(=O)C(=CC1=CC(=C(C(=C1)O)O)[N+](=O)[O-])C#N, 305.29000000000001, 116, 1.7813799999999995, 127.70000000000002			
14	Cymbalta (duloxetine), CCCC(C1=CC=CC1)OC2=CC=CC=C2C3=CC=CC=C32, 313.88399999999996, 116, 5.052700000000004, 21.259999999999998			
15	Diacomit (stiripentol), CC(C)(C)C(C=CC1=CC2=C(C=C1)OC2)O, 234.29499999999996, 92, 2.835500000000002, 38.69			
16	Dsuvia (sufentanil), CCC(=O)N(C1=CC=CC=C1)C2(CCN(C2)CCC=CC=C3)COC, 386.5610000000001, 146, 4.214800000000004, 32.78			
17	Epidiolex (cannabidiol), CCCCC1=CC(=C(C(=C1)O)C2C=C(C(C2C(=C)C)C)O), 314.46900000000005, 126, 5.846500000000007, 40.46			
18	Exelon (rivastigmine), CCN(C)C(=O)OC1=CC=CC(=C1)C(C)N(C)C, 250.34199999999998, 100, 2.7597000000000014, 32.78			
19	Fetzima (levomilnacipran), CCN(CC)C(=O)C1(C1CN)C2=CC=CC=C2, 282.815, 106, 2.193200000000005, 46.33			
20	Fintepla (fenfluramine), CCNC(C)CC1=CC(=CC=C1)C(F)(F)F, 231.26099999999997, 90, 3.2459000000000024, 12.03			
21	Firdapse (amifampridine), C1=CN=CC(=CN)N.OP(=O)(O)O, 207.126, 74, -0.6826, 142.69			
22	"Focalin, Focalin XR (dexamethylphenidate HCl)", COC(=O)C(C1CCCCN1)C2=CC=CC=C2, 269.772, 100, 2.5071000000000008, 38.33			
23	Frova (frovatriptan succinate), CN1CCC2=C(C1)C3=C(N2)C=CC(=C3)O.N.C(C(=O)O)C(=O)O, 379.4130000000002, 148, 0.454600000000006, 177.01			
24	Fycompa (perampanel), C1=CC=C(C=C1)N2C=C(C(=C(C2=O)C3=CC=CC=C3C#N)C4=CC=CC=N4, 349.39300000000003, 128, 4.438180000000004, 58.68			
25	Gocovri (amantadine), C1C2CC3CC1CC(C2)(C3)N, 151.253, 62, 1.9139000000000002, 26.02			
26	Gralise (gabapentin), C1CCC(C1)(CC(=O)O)CN, 171.23999999999998, 70, 1.3703, 63.319999999999999			
27	Hetlioz (tasimelteon), CCC(=O)NCC1CC1C2=C3CCOC3=CC=C2, 245.32199999999995, 96, 2.2512, 38.33			
28	Imitrex (sumatriptan), CNS(=O)(=O)CC1=CC2=C(C=C1)NC=C2CN(C)C, 295.408, 110, 1.3212, 65.2			
29	Ingrezza (valbenazine), CC(C)CC1CN2CCC=CC(=C(C=C3C2CC1OC(=O)C(C(C)C)N)OC)OC, 418.578000000000026, 168, 3.5641000000000016, 74.02000000000001			
30	Intuniv (guanfacine), C1=CC(=C(C(=C1)C1)CC(=O)N(C)N)C1, 282.55799999999994, 88, 1.757699999999998, 81.47			
31	Invega (paliperidone), CC1=C(C(=O)N2CCCC(C2=N1)O)CN3CCC(C3)C4=NC5=C4C=CC(=C5)F, 426.49200000000002, 164, 3.0813200000000016, 84.39			
32	Keppra (levetiracetam), CCC(C(=O)N)N1CCCC1=O, 170.212, 68, -0.1272999999999997, 63.400000000000006			
33	Lamictal (lamotrigine), C1=CC(=C(C(=C1)C1)C2=C(N=C(N=N2)N)N, 256.096, 82, 2.0098, 90.71			

FIGURE 5.10 Descriptors output (1)

5.4 RESULTS OF MOLECULAR FINGERPRINTING

```
In [2]: import pandas as pd
        from rdkit import Chem
        from rdkit.Chem import AllChem
        from rdkit.Chem import DataStructs

        # Function to calculate Morgan fingerprints for a molecule
        def calculate_morgan_fingerprint(molecule):
            """Calculate Morgan fingerprints for a given molecule."""
            radius = 2 # Morgan fingerprint radius
            n_bits = 2048 # Number of bits in the fingerprint
            fp = AllChem.GetMorganFingerprintAsBitVect(molecule, radius, nBits=n_bits)
            return fp
```

Figure 5.13 Function to calculate Morgan fingerprints for a molecule (1)

- This code begins by importing the necessary libraries, a function named `calculate_morgan_fingerprint` is defined to compute Morgan fingerprints for a given molecule using RDKit's `AllChem.GetMorganFingerprintAsBitVect()` function.
- Morgan fingerprints are a type of circular fingerprint that encode molecular structure information based on atom environments within a specified radius. Within the function, the desired fingerprint radius is set to 2, and the number of bits in the fingerprint is specified as 2048.
- The `GetMorganFingerprintAsBitVect()` function generates the Morgan fingerprint as a binary vector (BitVect), representing the presence or absence of specific substructures within the molecule.
- Once computed, the Morgan fingerprint is returned by the function, ready for further analysis and comparison with other fingerprints to assess structural similarity between molecules.

```
# Load data from CSV file with appropriate encoding
neuro_drugs_file_path = r"C:\Users\Mukesh\OneDrive\Documents\molecular fingerprint.csv"
df = pd.read_csv(neuro_drugs_file_path, encoding='latin1')

# Drop rows with 'nan' values in the 'SMILES' column
df = df.dropna(subset=['SMILES'])

# Convert 'SMILES' column to strings
df['SMILES'] = df['SMILES'].astype(str)
```

Figure 5.14 Loading data from CSV file (2)

- Data is loaded from a CSV file with appropriate encoding, and then rows containing 'nan' values in the 'SMILES' column are dropped, followed by converting the 'SMILES' column to strings.

```
# Add a column for Morgan fingerprints
df['Morgan_Fingerprint'] = None
```

Figure 5.15 Adding column for Morgan fingerprints (1)

- In this code, a new column named 'Morgan_Fingerprint' is added to the DataFrame `df`, initialized with 'None' values. This column will be populated with Morgan fingerprints calculated for each molecule in the dataset.

```
# Calculate Morgan fingerprints for each compound
for index, row in df.iterrows():
    smiles = row['SMILES']
    molecule = Chem.MolFromSmiles(smiles)
    if molecule is not None:
        fingerprint = calculate_morgan_fingerprint(molecule)
        df.at[index, 'Morgan_Fingerprint'] = fingerprint
```

Figure 5.16 Calculating Morgan fingerprints for each compound (1)

- In this code, Morgan fingerprints are computed for each compound in the DataFrame `df` by iterating over its rows using the `iterrows()` function.
- For each row, the SMILES notation is extracted from the 'SMILES' column, and RDKit's `Chem.MolFromSmiles()` function is employed to convert the SMILES notation into a molecular representation (RDKit `Mol` object).
- If a valid molecule is generated, the `calculate_morgan_fingerprint()` function is invoked to compute the Morgan fingerprint for the molecule.
- The resulting fingerprint is then assigned to the 'Morgan_Fingerprint' column in the DataFrame using the `at` method, ensuring that each compound's fingerprint is appropriately stored.
- This process enables the systematic calculation of Morgan fingerprints for all compounds in the dataset.

```
# Save the updated DataFrame to CSV
output_csv_path = "output_with_morgan_fingerprints.csv"
df.to_csv(output_csv_path, index=False)

print("Morgan fingerprints calculated and saved successfully!")
```

Morgan fingerprints calculated and saved successfully!

Figure 5.17 Saving the updated DataFrame to CSV (2)

- In this code, the updated DataFrame containing Morgan fingerprints for each compound is saved to a CSV file using Pandas' `to_csv()` function. The output CSV file path is specified as "output_with_morgan_fingerprints.csv", and the parameter `index=False` is included to prevent the DataFrame index from being written as a separate column in the CSV file.
- Following the successful saving of the DataFrame, a confirmation message is printed to the console, indicating that the Morgan fingerprints have been calculated and saved successfully.

The screenshot shows a Jupyter Notebook window titled "output_with_morgan_fingerprints.csv". The interface includes a top bar with "File", "Edit", "View", and "Language" menus, and a "Logout" button. Below the menu bar, a code cell displays a DataFrame with 22 rows and 3 columns: "Neurological drugs", "SMILES", and "Morgan_Fingerprint". The first few rows are visible, showing drug names like Abilify, Ampyra, Amrix, Apokyn, and Aptiom, along with their SMILES strings and corresponding Morgan fingerprints. The output is rendered as a text table within the notebook cell.

Figure 5.18 Molecular fingerprinting output (1)

```

23 Frova (frovatriptan succinate),CNC1CCC2=C(C1)C3=C(N2)C=CC(=C3)C(=O)N.C(CC(=O)O)C(=O)O.O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect
object at 0x0000023ADD29D9E0>
24 Fycompa (perampanel),C1=CC=C(C=C1)N2C=C(C=C(C2=O)C3=CC=CC=C3#N)C4=CC=CC=N4,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C2660>
25 Gocovri (amantadine),C1C2CC3CC1CC(C2)(C3)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C26D0>
26 Gralise (gabapentin),C1CCC(CC1)(CC(=O)O)CN,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C27B0>
27 Hetlioz (tasimelteon),CCC(=O)NCC1CC1C2=C3CCOC3=CC=C2,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2820>
28 Imitrex (sumatriptan),CNS(=O)(=O)CC1=CC2=C(C=C1)NC=C2CCN(C)C,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2890>
29 Ingrezza (valbenazine),CC(C)CC1CN2CCC=CC(=C(C=C3C2CC1OC(=O)C(C(C)C)N)OC)OC,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C2900>
30 Intuniv (guanfacine),C1=CC(=C(C(=C1)C1)CC(=O)N=C(N)N)C1.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2970>
31 Invega (paliperidone),CC1=C(C(=O)N2CCCC(C2=N1)O)CCN3CCC(CC3)C4=NOC5=C4C=CC(=C5)F,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C29E0>
32 Keppra (levetiracetam),CCC(C(=O)N)N1CCCC1=O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2A50>
33 Lamictal (lamotrigine),C1=CC(=C(C(=C1)C1)C1)C2=C(N=C(N=N2)N)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2AC0>
34 Lunesta (eszopiclone),CN1CCN(CC1)C(=O)OC2C3=NC=CNC3C(=O)N2C4=NC=C(C=C4)C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C2B30>
35 Lyrica (pregabalin),CC(C)CC(CC(=O)O)CN,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2BA0>
36 Maxalt (rizatriptan benzoate),CN(C)CCC1=CNC2=C1C=C(C=C2)CN3C=NC=N3.C1=CC=C(C=C1)C(=O)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect
object at 0x0000023ADD3C2C10>
37 Migranal (dihydroergotamine mesylate),CC1(C(=O)N2C(C(=O)N3CCCC3C2(O1)O)CC4=CC=CC=C4)NC(=O)C5CC6C(CC7=CN8=CC=CC6=C78)N(C5)C.CS(=O)(=O)O,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2C80>
38 Mirapex (pramipexole dihydrochloride),CCCN1CCC2=C(C1)SC(=N2)N.O.C1.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C2CF0>
39 Movantik (naloxegol),COCOCOCOCOCOCOCOCOCOC1CCC2(C3CC4=C5C2(C1OC5=C(C=C4)O)CCN3CC=C)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect
object at 0x0000023ADD3C2D60>
40 Namenda (memantine HCl),CC12CC3CC(C1)(CC(C3)(C2)N)C.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2DD0>
41 Nayzilam (midazolam),CC1=NC=C2N1C3=C(C=C(C3)C1)C(=NC2)C4=CC=CC=C4F,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C2E40>
42 Neupro (Rotigotine),CCCN(CCC1=CC=CS1)C2CCC3=C(C2)C=CC=C3O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2EB0>
43 Neurontin (gabapentin),C1CCC(CC1)(CC(=O)O)CN,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2F20>
44 Northera (droxidopa),C1=CC(=C(C=C1C(C(=O)O)N)O)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C2F90>

```

Figure 5.19 Molecular fingerprinting output (2)

```

45 Nourianz (istradefylline),CCN1C2=C(C(=O)N(C1=O)CC)N(C(=N2)C=CC3=CC(=C(C=C3)OC)OC)<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0040>
46 Nucynta (tapentadol),CCC(C1=CC(=CC=C1)O)C(C)CN(C)C.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C00B0>
47 Nuedexta (dextromethorphan hydrobromide and quinidine
sulfate),COC1=CC2=C(C=CN=C2C=C1)C(C3CC4CCN3CC4C=C)O.COC1=CC2=C(C=CN=C2C=C1)C(C3CC4CCN3CC4C=C)O.O.O.OS(=O)(=O)O,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0120>
48 Nuplazid (pimavanserin),CC(C)COC1=CC=C(C=C1)CNC(=O)N(CC2=CC=C(C=C2)F)C3CCN(CC3)C,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0190>
49 Nuvigil (armodafinil),C1=CC=C(C=C1)C(C2=CC=CC=C2)S(=O)CC(=O)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0200>
50 Onfi (clobazam),CN1C(=O)CC(=O)N(C2=C1C=CC(=C2)C1)C3=CC=CC=C3,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0270>
51 Oxecta (oxycodone HCl),CN1CCC23C4C(=O)CCC2(C1CC5=C3C(=C(C=C5)OC)O4)O.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C02E0>
52 Qelbree (viloxazine),CCOC1=CC=CC=C1OCC2CNCCO2.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0350>
53 Qutenza (capsaicin),CC(C)C=CCCCC(=O)NCC1=CC(=C(C=C1)O)OC,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C03C0>
54 Radicava (edaravone),CC1=NN(C(=O)C1)C2=CC=CC=C2,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0430>
55 Relpax (eletriptan hydrobromide),CN1CCCC1CC2=CNC3=C2C=C(C=C3)CCS(=O)(=O)C4=CC=CC=C4.Br,<rdkit.DataStructs.cDataStructs.ExplicitBitVect
object at 0x0000023ADD3C04A0>
56 Reminyl (galantamine hydrobromide),CN1CCC23C=CC(CC2OC4=C(C=CC(=C34)C1)OC)O.Br,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0510>
57 Reyvow (lasmiditan),CN1CCC(CC1)C(=O)C2=NC(=CC=C2)NC(=O)C3=C(C=C(C=C3F)F)F,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0580>
58 Rilutek (riluzole),C1=CC2=C(C=C1OC(F)(F)F)SC(=N2)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C05F0>
59 Rozerem (ramelteon),CCC(=O)NCCC1CCC2=C1C3=C(C=C2)OCC3,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0660>
60 Sabril (vigabatrin),C=CC(CCC(=O)O)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C06D0>
61 Silenor (doxepin),CN(C)CCC=C1C2=CC=CC=C2OC3=CC=CC=C31.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0740>
62 Sonata (zaleplon),CCN(C1=CC=CC(=C1)C2=CC=NC3=C(C=NN23)C#N)C(=O)C,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C07B0>
63 Spravato (esketamine),CNC1(CCCCC1=O)C2=CC=CC=C2C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0820>
64 Sprix (ketorolac tromethamine),C1CN2C(=CC=C2C(=O)C3=CC=CC=C3)C1C(=O)O.C(C(CO)(CO)N)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object
at 0x0000023ADD3C0890>
65 Stavzor (valproic acid),CCCC(CCC)C(=O)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0900>
66 Strattera (atomoxetine HCl),CC1=CC=CC=C1OC(CNC)C2=CC=CC=C2.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0970>

```

Figure 5.20 Molecular fingerprinting output (3)

```

66 Strattera (atomoxetine HCl),CC1=CC=CC=C1OC(CCN)C2=CC=CC=C2.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0970>
67 Subsys (fentanyl sublingual spray),CCC(=O)N(C1CN(C(C1)CCC2=CC=CC=C2)C3=CC=CC=C3,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C09E0>
68 "Tegretol, Tegretol XR (carbamazepine)",C1=CC=C2C(=C1)C=CC3=CC=CC=C3N2C(=O)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0A50>
69 Tivorbex (indomethacin),CC1=C(C2=C(N1C(=O)C3=CC=C(C=C3)C1)C=CC(=C2)OC)CC(=O)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0AC0>
70 Topamax (topiramate),CC1(OC2COC3(C(C2O1)OC(O3)(C)C)COS(=O)(=O)N)C,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0B30>
71 Trileptal (oxcarbazepine),C1C2=CC=CC=C2N(C3=CC=CC=C3C1=O)C(=O)N,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0BA0>
72 Trudhesa (dihydroergotamine mesylate),CC1(C(=O)N2C(C(=O)N3CCCC3C2(O1)O)CC4=CC=CC=C4)NC(=O)C5CC6(C(C7=CNC8=CC=CC=C78)N(C5)C.CS(=O)(=O)O,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0C10>
73 Ubrelvy (ubrogepant),CC1C(CC(C(=O)N1CC(F)(F)F)NC(=O)C2=CC3=C(CC4(C3)C5=C(NC4=O)N=CC=C5)N=C2)C6=CC=CC=C6,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0C80>
74 Ultracet (acetaminophen and tramadol HCl),CC(=O)NC1=CC=C(C=C1)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0CF0>
75 Viibryd (vilazodone hydrochloride),C1CN(CCN1CCCC2=CNC3=C2C=C(C=C3)C#N)C4=CC5=C(C=C4)OC(=C5)C(=O)N.C1,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0D60>
76 Vimpat (lacosamide),CC(=O)NC(COC)C(=O)NCC1=CC=CC=C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0DD0>
77 Vivlodex (meloxicam),CC1=CN=C(S1)NC(=O)C2=C(C3=CC=CC=C3S(=O)(=O)N2C)O,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0E40>
78 Vraylar (cariprazine),CN(C)C(=O)NC1CCC(CC1)CCN2CCN(CC2)C3=C(C(=CC=C3)C1)C1.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0EB0>
79 Vyvanse (Lisdexamfetamine Dimesylate),CC(CC1=CC=CC=C1)NC(=O)C(CCCN)N.CS(=O)(=O)O.CS(=O)(=O)O,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0F20>
80 Xadago (safinamide),CC(C(=O)N)NCC1=CC=C(C=C1)OCC2=CC(=CC=C2)F,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C0F90>
81 Zanaflex (tizanidine hydrochloride),C1CN=C(N1)NC2=C(C=CC3=NSN=C32)C1.C1,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C0540>
82 Zipsor (diclofenac potassium),C1=CC=C(C(=C1)CC(=O)[O-])NC2=C(C=CC=C2)C1.[K+],<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at
0x0000023ADD3C05B0>
83 Zomig (zolmitriptan),CN(C)CCC1=CNC2=C1C=C(C=C2)CC3COC(=O)N3,<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x0000023ADD3C5120>

```

Figure 5.21 Molecular fingerprinting output (4)

Molecular fingerprints, such as Morgan fingerprints, were generated for all existing drugs listed in the provided CSV file.

5.5 RESULTS OF SIMILARITY SEARCHING

```

In [3]: import pandas as pd
        from rdkit import Chem
        from rdkit.Chem import AllChem
        from rdkit.Chem import DataStructs

        # Function to calculate Morgan fingerprints for a molecule
        def calculate_morgan_fingerprint(molecule):
            """Calculate Morgan fingerprints for a given molecule."""
            radius = 2 # Morgan fingerprint radius
            n_bits = 2048 # Number of bits in the fingerprint
            fp = AllChem.GetMorganFingerprintAsBitVect(molecule, radius, nBits=n_bits)
            return fp

```

Figure 5.22 Function to calculate Morgan fingerprints for a molecule (2)

- This code begins by importing the necessary libraries, a function named `calculate_morgan_fingerprint` is defined to compute Morgan fingerprints for a given molecule using RDKit's `AllChem.GetMorganFingerprintAsBitVect()` function.

- Morgan fingerprints are a type of circular fingerprint that encode molecular structure information based on atom environments within a specified radius. Within the function, the desired fingerprint radius is set to 2, and the number of bits in the fingerprint is specified as 2048.
- The `GetMorganFingerprintAsBitVect()` function generates the Morgan fingerprint as a binary vector (`BitVect`), representing the presence or absence of specific substructures within the molecule.
- Once computed, the Morgan fingerprint is returned by the function, ready for further analysis and comparison with other fingerprints to assess structural similarity between molecules.

```
# Load data from CSV file with appropriate encoding
neuro_drugs_file_path = r"C:\Users\Mukesh\OneDrive\Documents\molecular_fingerprint.csv"
df = pd.read_csv(neuro_drugs_file_path, encoding='latin1')

# Drop rows with 'nan' values in the 'SMILES' column
df = df.dropna(subset=['SMILES'])

# Convert 'SMILES' column to strings
df['SMILES'] = df['SMILES'].astype(str)
```

Figure 5.23 Loading data from CSV (3)

- Data is loaded from a CSV file with appropriate encoding, and then rows containing 'nan' values in the 'SMILES' column are dropped, followed by converting the 'SMILES' column to strings.

```
# Add a column for Morgan fingerprints
df['Morgan_Fingerprint'] = None
```

Figure 5.24 Adding column for Morgan fingerprints (2)

- In this code, a new column named 'Morgan_Fingerprint' is added to the DataFrame `df`, initialized with 'None' values. This column will be populated with Morgan fingerprints calculated for each molecule in the dataset.

```
# Calculate Morgan fingerprints for each compound
for index, row in df.iterrows():
    smiles = row['SMILES']
    molecule = Chem.MolFromSmiles(smiles)
    if molecule is not None:
        fingerprint = calculate_morgan_fingerprint(molecule)
        df.at[index, 'Morgan_Fingerprint'] = fingerprint
```

Figure 5.25 Calculating Morgan fingerprints for each compound (2)

- In this code, Morgan fingerprints are computed for each compound in the DataFrame **df** by iterating over its rows using the **iterrows()** function.
- For each row, the SMILES notation is extracted from the 'SMILES' column, and RDKit's **Chem.MolFromSmiles()** function is employed to convert the SMILES notation into a molecular representation (RDKit **Mol** object).
- If a valid molecule is generated, the **calculate_morgan_fingerprint()** function is invoked to compute the Morgan fingerprint for the molecule.
- The resulting fingerprint is then assigned to the 'Morgan_Fingerprint' column in the DataFrame using the **at** method, ensuring that each compound's fingerprint is appropriately stored.
- This process enables the systematic calculation of Morgan fingerprints for all compounds in the dataset.

```
# Function to calculate Tanimoto similarity
def calculate_tanimoto_similarity(fp1, fp2):
    """Calculate Tanimoto similarity between two fingerprints."""
    return DataStructs.TanimotoSimilarity(fp1, fp2)
```

Figure 5.26 Function to calculate Tanimoto similarity

The function **calculate_tanimoto_similarity** computes the Tanimoto similarity between two fingerprints, **fp1** and **fp2**, using RDKit's **DataStructs.TanimotoSimilarity()** function. This similarity measure quantifies the overlap between the binary representations of the fingerprints, providing a numerical value indicating the degree of similarity between the two compounds.

```

# Define a function for similarity searching with threshold
def similarity_search(query_smiles, threshold):
    """Perform similarity search with a specified threshold."""
    query_morgan_fp = calculate_morgan_fingerprint(Chem.MolFromSmiles(query_smiles))
    similarities = []
    for fp in df['Morgan_Fingerprint']:
        sim = calculate_tanimoto_similarity(query_morgan_fp, fp)
        similarities.append(sim)
    df['Similarity'] = similarities
    result = df[df['Similarity'] >= threshold]
    return result

```

Figure 5.27 Defining a function for similarity searching with threshold

- The function `similarity_search` conducts a similarity search based on a specified threshold, using Morgan fingerprints to compare compounds.
- The Tanimoto similarity between this query fingerprint and the fingerprints of all molecules in the DataFrame (`df`) is calculated using the `calculate_tanimoto_similarity` function.
- The resulting similarities are stored in a list, and a new column named 'Similarity' is added to the DataFrame to capture these values.

```

# Query SMILES string and similarity threshold -METOPROLOL
query_smiles = 'CC(C)NCC(COC1=CC=C(C=C1)CCOC)O'
threshold = 0.27
# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.28 Performing Similarity searching - Metoprolol

- In this code, a similarity search is performed using the SMILES string of the compound "Metoprolol" and a specified similarity threshold of 0.27.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

Compounds similar to the query compound with a similarity above the threshold:

```
Neurological drugs \
47 Nuplazid (pimavanserin)
79 Xadago (safinamide)

SMILES \
47 CC(C)COC1=CC=C(C=C1)CNC(=O)N(CC2=CC=C(C=C2)F)C...
79 CC(C(=O)N)NCC1=CC=C(C=C1)OCC2=CC(=CC=C2)F

Morgan_Fingerprint Similarity
47 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.271429
79 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.274194
```

Figure 5.29 Potential Drug candidate – Metoprolol

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
 - This process enables the identification of structurally similar compounds to "METOPROLOL" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.
1. The process involves first importing the necessary libraries such as pandas, RDKit's Chem module, and DataStructs module.
 2. Then, defining a function to calculate Morgan fingerprints for a molecule. Following that, loading data from a CSV file with appropriate encoding, dropping rows with 'nan' values in the 'SMILES' column, and converting the 'SMILES' column to strings are performed.
 3. Subsequently, a column for Morgan fingerprints is added to the DataFrame. Morgan fingerprints are then calculated for each compound in the dataset. Next, a function to calculate Tanimoto similarity is defined.
 4. Finally, a function for similarity searching with a specified threshold is defined, which utilizes the calculated Morgan fingerprints and the Tanimoto similarity calculation function.
 5. These steps form a standardized and applicable for all upcoming potential drug candidates for similarity searching and analysis.

```

# Query SMILES string and similarity threshold - Fexofenadine
query_smiles = 'CC(C)(C1=CC=C(C=C1)C(CCCN2CCC(CC2)C(C3=CC=CC=C3)(C4=CC=CC=C4)O)O)C(=O)O'
threshold = 0.30

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.30 Performing Similarity searching – Fexofenadine

- In this code, a similarity search is performed using the SMILES string of the compound "Fexofenadine" and a specified similarity threshold of 0.30.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```

Compounds similar to the query compound with a similarity above the threshold:
      Neurological drugs \
66 Subsys (fentanyl sublingual spray)

      SMILES \
66 CCC(=O)N(C1CCN(CC1)CCC2=CC=CC=C2)C3=CC=CC=C3

      Morgan_Fingerprint  Similarity
66 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  0.313433

```

Figure 5.31 Potential Drug candidate – Fexofenadine

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Fexofenadine" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Loratadine
query_smiles = 'CCOC(=O)N1CCC(=C2C3=C(CCC4=C2N=CC=C4)C=C(C=C3)C1)CC1'
threshold = 0.30
# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.32 Performing Similarity searching – Loratadine

- In this code, a similarity search is performed using the SMILES string of the compound "Loratadine" and a specified similarity threshold of 0.30.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```

Compounds similar to the query compound with a similarity above the threshold:
Neurological drugs                                     SMILES \
33 Lunesta (eszopiclone) CN1CCN(CC1)C(=O)OC2C3=NC=CN=C3C(=O)N2C4=NC=C(C...

                                     Morgan_Fingerprint  Similarity
33 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...  0.324675

```

Figure 5.33 Potential Drug candidate – Loratadine

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Loratadine" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Loperamide
query_smiles = 'CN(C)C(=O)C(CCN1CCC(CC1)(C2=CC=C(C=C2)Cl)O)(C3=CC=CC=C3)C4=CC=CC=C4'
threshold = 0.27

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.34 Performing Similarity searching – Loperamide

- In this code, a similarity search is performed using the SMILES string of the compound "Loperamide" and a specified similarity threshold of 0.27.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```

Compounds similar to the query compound with a similarity above the threshold:
Neurological drugs      SMILES \
77 Vraylar (cariprazine) CN(C)C(=O)NC1CCC(CC1)CCN2CCN(CC2)C3=C(C(=CC=C3...

                                Morgan_Fingerprint  Similarity
77 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ...    0.272727

```

Figure 5.35 Potential Drug candidate – Loperamide

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Loperamide" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Clopidogrel
query_smiles = 'COC(=O)C(C1=CC=CC=C1Cl)N2CCC3=C(C2)C=CS3'
threshold = 0.25

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.36 Performing Similarity searching – Clopidogrel

- In this code, a similarity search is performed using the SMILES string of the compound "Clopidogrel" and a specified similarity threshold of 0.25.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```

Compounds similar to the query compound with a similarity above the threshold:
      Neurological drugs \
21  Focalin, Focalin XR (dexmethylphenidate HCl)

      SMILES \
21  COC(=O)C(C1CCCCN1)C2=CC=CC=C2.Cl

      Morgan_Fingerprint  Similarity
21  [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  0.261538

```

Figure 5.37 Potential Drug candidate – Clopidogrel

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Clopidogrel" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Lorcaserin
query_smiles = 'CC1CNCCC2=C1C=C(C=C2)Cl'
threshold = 0.20

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.38 Performing Similarity searching – Lorcaserin

- In this code, a similarity search is performed using the SMILES string of the compound "Lorcaserin" and a specified similarity threshold of 0.20.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```

Compounds similar to the query compound with a similarity above the threshold:
Neurological drugs
49 Onfi (clobazam) CN1C(=O)CC(=O)N(C2=C1C=CC(=C2)Cl)C3=CC=CC=C3
51 Qelbree (viloxazine) CCOC1=CC=CC=C1OCC2CNCCC2.Cl

Morgan_Fingerprint Similarity
49 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.207547
51 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.203704

```

Figure 5.39 Potential Drug candidate – Lorcaserin

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Lorcaserin" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Tamsulosin
query_smiles = 'CCOC1=CC=CC=C1OCCNC(C)CC2=CC(=C(C=C2)OC)S(=O)(=O)N'
threshold = 0.27

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.40 Performing Similarity searching – Tamsulosin

- In this code, a similarity search is performed using the SMILES string of the compound "Tamsulosin" and a specified similarity threshold of 0.27.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

Compounds similar to the query compound with a similarity above the threshold:

	Neurological drugs	SMILES	\
19	Fintepla (fenfluramine)	CCNC(C)CC1=CC(=CC=C1)C(F)(F)F	

	Morgan_Fingerprint	Similarity
19	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.285714

Figure 5.41 Potential Drug candidate – Tamsulosin

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Tamsulosin" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```
# Query SMILES string and similarity threshold - Nebivolol
query_smiles = 'C1CC2=C(C=CC(=C2)F)OC1C(CNCC(C3CCC4=C(O3)C=CC(=C4)F)O)O'
threshold = 0.22

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)
```

Figure 5.42 Performing Similarity searching – Nebivolol

- In this code, a similarity search is performed using the SMILES string of the compound "Nebivolol" and a specified similarity threshold of 0.22.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```
Compounds similar to the query compound with a similarity above the threshold:
Neurological drugs \
47 Nuplazid (pimavanserin)
58 Rozerem (ramelteon)

SMILES \
47 CC(C)COC1=CC=C(C=C1)CNC(=O)N(CC2=CC=C(C=C2)F)C...
58 CCC(=O)NCCC1CCC2=C1C3=C(C=C2)OCC3

Morgan_Fingerprint Similarity
47 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.200000
58 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.227273
```

Figure 5.43 Potential Drug candidate – Nebivolol

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Nebivolol" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Salmeterol
query_smiles = 'C1=CC=C(C=C1)CCCCOCCCCCNCC(C2=CC(=C(C=C2)O)CO)O'
threshold = 0.20

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.44 Performing Similarity searching – Salmeterol

- In this code, a similarity search is performed using the SMILES string of the compound "Salmeterol" and a specified similarity threshold of 0.20.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

Compounds similar to the query compound with a similarity above the threshold:

	Neurological drugs	SMILES \	Morgan_Fingerprint	Similarity
43	Northera (droxidopa)	<chem>C1=CC(=C(C=C1C(C(C(=O)O)N)O)O)O</chem>	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]	0.209677
75	Vimpat (lacosamide)	<chem>CC(=O)NC(COC)C(=O)NCC1=CC=CC=C1</chem>	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...]	0.223881

Figure 5.45 Potential Drug candidate – Salmeterol

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Salmeterol" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```
# Query SMILES string and similarity threshold - Cyclizine
query_smiles = 'CN1CCN(CC1)C(C2=CC=CC=C2)C3=CC=CC=C3'
threshold = 0.25

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)
```

Figure 5.46 Performing Similarity searching – Cyclizine

- In this code, a similarity search is performed using the SMILES string of the compound "Cyclizine" and a specified similarity threshold of 0.25.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```
Compounds similar to the query compound with a similarity above the threshold:
Neurological drugs \
21 Focalin, Focalin XR (dexamethylphenidate HCl)
48 Nuvigil (armodafinil)

SMILES \
21 COC(=O)C(C1CCCCN1)C2=CC=CC=C2.Cl
48 C1=CC=C(C=C1)C(C2=CC=CC=C2)S(=O)CC(=O)N

Morgan_Fingerprint Similarity
21 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.25000
48 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0.25641
```

Figure 5.47 Potential Drug candidate – Cyclizine

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Cyclizine" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```
# Query SMILES string and similarity threshold - Cromolyn
query_smiles = 'C1=CC2=C(C(=C1)OCC(COC3=CC=CC4=C3C(=O)C=C(O4)C(=O)O)O)C(=O)C=C(O2)C(=O)O'
threshold = 0.25

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)
```

Figure 5.48 Performing Similarity searching – Cromolyn

- In this code, a similarity search is performed using the SMILES string of the compound "Cromolyn" and a specified similarity threshold of 0.25.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```
Compounds similar to the query compound with a similarity above the threshold:
Neurological drugs      SMILES \
79 Xadago (safinamide)  CC(C(=O)N)NCC1=CC=C(C=C1)OCC2=CC(=CC=C2)F

                                Morgan_Fingerprint  Similarity
79 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...    0.265625
```

Figure 5.49 Potential Drug candidate – Cromolyn

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Cromolyn" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```

# Query SMILES string and similarity threshold - Dimenhydrinate
query_smiles = 'CN1C2=C(C(=O)N(C1=O)C)NC(=N2)C1.CN(C)CCOC(C1=CC=CC=C1)C2=CC=CC=C2'
threshold = 0.20

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)

```

Figure 5.50 Performing Similarity searching – Dimenhydrinate

- In this code, a similarity search is performed using the SMILES string of the compound "Dimenhydrinate" and a specified similarity threshold of 0.20.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

Compounds similar to the query compound with a similarity above the threshold:

	Neurological drugs \	SMILES \	Morgan_Fingerprint	Similarity
17	Exelon (rivastigmine)	CCN(C)C(=O)OC1=CC=CC(=C1)C(C)N(C)C	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.211268
35	Maxalt (rizatriptan benzoate)	CN(C)CCC1=CNC2=C1C=C(C=C2)CN3C=NC=N3.C1=CC=C(C...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.206897

Figure 5.51 Potential Drug candidate – Dimenhydrinate

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Dimenhydrinate" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```
# Query SMILES string and similarity threshold - Meclizine
query_smiles = 'CC1=CC(=CC=C1)CN2CCN(CC2)C(C3=CC=CC=C3)C4=CC=C(C=C4)C1'
threshold = 0.20

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)
```

Figure 5.52 Performing Similarity searching – Meclizine

- In this code, a similarity search is performed using the SMILES string of the compound "Meclizine" and a specified similarity threshold of 0.20.
- The `similarity_search` function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable `result`.

```
Compounds similar to the query compound with a similarity above the threshold:
      Neurological drugs \
49      Onfi (clobazam)
66 Subsys (fentanyl sublingual spray)

      SMILES \
49 CN1C(=O)CC(=O)N(C2=C1C=CC(=C2)C1)C3=CC=CC=C3
66 CCC(=O)N(C1CCN(CC1)CCC2=CC=CC=C2)C3=CC=CC=C3

      Morgan_Fingerprint  Similarity
49 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  0.225806
66 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  0.208955
```

Figure 5.53 Potential Drug candidate – Meclizine

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.

- This process enables the identification of structurally similar compounds to " Meclizine " within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```
# Query SMILES string and similarity threshold - Scopolamine
query_smiles = 'CN1C2CC(CC1C3C2O3)OC(=O)C(CO)C4=CC=CC=C4'
threshold = 0.25

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)
```

Figure 5.54 Performing Similarity searching – Scopolamine

- In this code, a similarity search is performed using the SMILES string of the compound " Scopolamine" and a specified similarity threshold of 0.25.
- The similarity_search function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable result.

```
Compounds similar to the query compound with a similarity above the threshold:
      Neurological drugs \
21  Focalin, Focalin XR (dexmethylphenidate HCl)
      SMILES \
21  COC(=O)C(C1CCCN1)C2=CC=CC=C2.Cl
      Morgan_Fingerprint  Similarity
21  [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  0.278689
```

Figure 5.55 Potential Drug candidate – Scopolamine

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.

- This process enables the identification of structurally similar compounds to "Scopolamine" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

```
# Query SMILES string and similarity threshold - Nifedipine
query_smiles = 'CC1=C(C(C(=C(N1)C)C(=O)OC)C2=CC=CC=C2[N+](=O)[O-])C(=O)OC'
threshold = 0.22

# Perform similarity search
result = similarity_search(query_smiles, threshold)

# Display the results
print("Compounds similar to the query compound with a similarity above the threshold:")
print(result)
```

Figure 5.56 Performing Similarity searching – Nifedipine

- In this code, a similarity search is performed using the SMILES string of the compound "Nifedipine" and a specified similarity threshold of 0.22.
- The similarity_search function is called with the query SMILES string and threshold parameters, which internally computes the Morgan fingerprint for the query molecule and compares it with the fingerprints of all compounds in the dataset.
- Compounds with similarity scores above the threshold are filtered, and the resulting DataFrame containing the matching compounds is stored in the variable result.

Compounds similar to the query compound with a similarity above the threshold:

	Neurological drugs \	SMILES \	Morgan_Fingerprint	Similarity
5	Aptiom (eslicarbazepine acetate)			
21	Focalin, Focalin XR (dexmethylphenidate HCl)			
75	Vimpat (lacosamide)			
5		CC(=O)OC1CC2=CC=CC=C2N(C3=CC=CC=C13)C(=O)N		
21		COC(=O)C(C1CCCCN1)C2=CC=CC=C2.Cl		
75		CC(=O)NC(COC)C(=O)NCC1=CC=CC=C1		
5			[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.229508
21			[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.233333
75			[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...	0.224138

Figure 5.57 Potential Drug candidate – Nifedipine

- Finally, a message is printed to the console, displaying the compounds similar to the query compound with a similarity above the specified threshold.
- This process enables the identification of structurally similar compounds to "Nifedipine" within the dataset, aiding in the exploration of potential drug candidates with similar chemical structures.

The selection of 15 potential drug candidates, including Metoprolol, Fexofenadine, Loratadine, Loperamide, Clopidogrel, Lorcaserin, Tamsulosin, Nebivolol, Salmeterol, Cyclizine, Cromolyn, Dimenhydrinate, Meclizine, Scopolamine, and Nifedipine, represents a diverse range of pharmaceutical compounds with varying therapeutic indications and chemical structures. These candidates have been identified based on their potential for repurposing using RDKit features, such as molecular descriptors, molecular fingerprinting, and similarity searching.

TABLE 5.1 Potential drug candidates with indication

NO.	Drug	Indication
1.	Metoprolol	Heart failure, myocardial infarction prevention
2.	Fexofenadine	Allergic rhinitis, chronic idiopathic urticaria
3.	Loratadine	Allergic rhinitis, chronic idiopathic urticaria
4.	Loperamide	Acute and chronic diarrhea
5.	Clopidogrel	Prevention of heart attack, stroke, and other cardiovascular events
6.	Lorcaserin	Obesity management
7.	Tamsulosin	Benign prostatic hyperplasia (BPH), urinary symptoms
8.	Nebivolol	Hypertension, heart failure
9.	Salmeterol	Asthma, chronic obstructive pulmonary disease (COPD)
10.	Cyclizine	Motion sickness, nausea, vomiting
11.	Cromolyn	Allergic rhinitis, asthma prevention
12.	Dimenhydrinate	Motion sickness, nausea, vomiting
13.	Meclizine	Motion sickness, vertigo
14.	Scopolamine	Motion sickness, nausea, vomiting
15.	Nifedipine	Hypertension, angina

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

This project focuses on employing computational techniques, specifically utilizing RDKit cheminformatics tools, to facilitate drug repurposing in neurological disorders. The methodology involves several pivotal stages aimed at identifying potential drug candidates with repurposing potential.

Initially, a comprehensive dataset of existing drugs utilized in neurological disorder treatments is compiled from reputable sources such as DrugBank and PubChem. Subsequently, RDKit tools are deployed to analyze the chemical properties and structural features of these compounds. Molecular descriptors, including Molecular Weight, Number of Valence Electrons, LogP, and Topological Polar Surface Area (TPSA), are computed to characterize the compounds. Additionally, Morgan fingerprints are generated to capture molecular structural information.

The project then proceeds to perform similarity searching, utilizing RDKit algorithms, to identify compounds with analogous structural profiles to existing drugs, yet with distinct pharmacological targets. This approach presents promising opportunities for drug repurposing. From this analysis, we identified several compounds with promising repurposing potential for neurological disorders. Notable identified compounds include Metoprolol, Fexofenadine, Loratadine, Loperamide, Clopidogrel, Lorcaserin, Tamsulosin, Nebivolol, Salmeterol, Cyclizine, Cromolyn, Dimenhydrinate, Meclizine, Scopolamine, and Nifedipine. These compounds exhibit molecular properties and structural similarities indicative of latent therapeutic potential, thereby laying the groundwork for further investigation and validation.

The project aims to prioritize candidate compounds based on their predicted pharmacological properties and potential for repurposing as treatments for neurological disorders. By seamlessly integrating computational methods with

cheminformatics tools, this endeavor seeks to expedite the drug discovery process and facilitate the identification of innovative therapeutic interventions for neurological disorders.

6.2 CONCLUSION

In conclusion, this project represents a significant advancement in drug repurposing for neurological disorders by harnessing computational methodologies and RDKit cheminformatics tools to identify potential therapeutic candidates. Through the compilation and analysis of a comprehensive dataset of existing drugs, we have effectively utilized molecular descriptors, molecular fingerprinting, and similarity searching algorithms to characterize the chemical properties and structural features of these compounds. This systematic approach has led to the identification of promising candidates with repurposing potential, thereby opening new avenues for therapeutic intervention in neurological conditions.

Looking ahead, further validation through experimental studies will be crucial to confirm the efficacy and safety of repurposed drugs. This will ensure that the identified candidates can translate into tangible clinical benefits for patients. Ultimately, the outcomes of this project have the potential to revolutionize neurological care by expanding treatment options, improving patient outcomes, and advancing precision medicine approaches in neurology. Through continued research and collaboration, we can strive towards realizing these transformative goals and making meaningful contributions to the field of neurology.

REFERENCES

1. WHO. (2020). Neurological disorders. World Health Organization. <https://www.who.int/news-room/fact-sheets/detail/neurological-disorders>
2. GBD 2019 Diseases and Injuries Collaborators. (2020). Global burden of 369 diseases and injuries in 204 countries and territories, 1990-2019: a systematic analysis for the Global Burden of Disease Study 2019. *Lancet* (London, England), 396(10258), 1204–1222. [https://doi.org/10.1016/S0140-6736\(20\)30925-9](https://doi.org/10.1016/S0140-6736(20)30925-9)
3. NINDS. (2020). National Institute of Neurological Disorders and Stroke. <https://www.ninds.nih.gov/Disorders>
4. Feigin, V. L., Nichols, E., Alam, T., Bannick, M. S., Beghi, E., Blake, N., & Chavan, P. P. (2019). Global, regional, and national burden of neurological disorders, 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016. *The Lancet Neurology*, 18(5), 459-480. [https://doi.org/10.1016/S1474-4422\(18\)30499-X](https://doi.org/10.1016/S1474-4422(18)30499-X)
5. GBD 2016 Neurology Collaborators. (2019). Global, regional, and national burden of neurological disorders, 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016. *The Lancet Neurology*, 18(5), 459-480. [https://doi.org/10.1016/S1474-4422\(18\)30499-X](https://doi.org/10.1016/S1474-4422(18)30499-X)
6. Ashburn, T. T., & Thor, K. B. (2004). Drug repositioning: identifying and developing new uses for existing drugs. *Nature Reviews Drug Discovery*, 3(8), 673-683. <https://doi.org/10.1038/nrd1468>

7. Paul, S. M., Mytelka, D. S., Dunwiddie, C. T., Persinger, C. C., Munos, B. H., Lindborg, S. R., & Schacht, A. L. (2010). How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature Reviews Drug Discovery*, 9(3), 203-214. <https://doi.org/10.1038/nrd3078>
8. Kola, I., & Landis, J. (2004). Can the pharmaceutical industry reduce attrition rates? *Nature Reviews Drug Discovery*, 3(8), 711-715. <https://doi.org/10.1038/nrd1470>
9. DiMasi, J. A., Grabowski, H. G., & Hansen, R. W. (2016). Innovation in the pharmaceutical industry: New estimates of R&D costs. *Journal of Health Economics*, 47, 20-33. <https://doi.org/10.1016/j.jhealeco.2016.01.012>
10. Pushpakom, S., Iorio, F., Eyers, P. A., Escott, K. J., Hopper, S., Wells, A., ... & Pirmohamed, M. (2019). Drug repurposing: progress, challenges and recommendations. *Nature Reviews Drug Discovery*, 18(1), 41-58. <https://doi.org/10.1038/nrd.2018.168>
11. Ashburn, T. T., & Thor, K. B. (2004). Drug repositioning: identifying and developing new uses for existing drugs. *Nature Reviews Drug Discovery*, 3(8), 673-683. <https://doi.org/10.1038/nrd1468>
12. Nosengo, N. (2016). Can you teach old drugs new tricks? *Nature*, 534(7607), 314-316. <https://doi.org/10.1038/534314a>
13. Corsello, S. M., Bittker, J. A., Liu, Z., Gould, J., McCarren, P., Hirschman, J. E., & Roth, F. P. (2017). The Drug Repurposing Hub: a next-generation drug library and information resource. *Nature Medicine*, 23(4), 405-408. <https://doi.org/10.1038/nm.4306>

14. Dudley, J. T., Deshpande, T., & Butte, A. J. (2011). Exploiting drug-disease relationships for computational drug repositioning. *Briefings in Bioinformatics*, 12(4), 303-311. <https://doi.org/10.1093/bib/bbr013>
15. Hurle, M. R., Yang, L., Xie, Q., Rajpal, D. K., Sanseau, P., & Agarwal, P. (2013). Computational drug repositioning: from data to therapeutics. *Clinical Pharmacology & Therapeutics*, 93(4), 335-341. <https://doi.org/10.1038/clpt.2013.1>
16. Langedijk, J., Mantel-Teeuwisse, A. K., Slijberman, D. S., Schutjens, M. H., & Drug Rediscovery Working Group. (2015). Drug repositioning and repurposing: terminology and definitions in literature. *Drug Discovery Today*, 20(8), 1027-1034. <https://doi.org/10.1016/j.drudis.2015.05.001>
17. Kaltenbach, H. M., Kauffmann, H. M., & Berres, M. (2007). Successful drug repositioning: A win-win opportunity for drug development. *Drug Discovery Today*, 12(7-8), 333-336. <https://doi.org/10.3390/v12091058>.
18. Proudfoot, J. R. (2002). Drugs, leads, and drug-likeness: an analysis of some recently launched drugs. *Bioorganic & Medicinal Chemistry Letters*, 12(12), 1647-1650. [https://doi.org/10.1016/s0960-894x\(02\)00244-5](https://doi.org/10.1016/s0960-894x(02)00244-5)
19. Schneider, G., & Fechner, U. (2005). Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8), 649-663. <https://doi.org/10.1038/nrd1799>
20. Kitchen, D. B., Decornez, H., Furr, J. R., & Bajorath, J. (2004). Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature Reviews Drug Discovery*, 3(11), 935-949. <https://doi.org/10.1038/nrd1549>

21. Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., & Blaschke, T. (2018). The rise of deep learning in drug discovery. *Drug discovery today*, 23(6), 1241–1250. <https://doi.org/10.1016/j.drudis.2018.01.039>
22. Hartenfeller, M., Schneider, G., & Reutlinger, M. (2012). Computational target prediction. In *Chemoinformatics and Computational Chemical Biology* (pp. 547-579). Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-1-60761-839-3>
23. Oprea, T. I., & Gottfries, J. (2001). Chemography: the art of navigating in chemical space. *Journal of Combinatorial Chemistry*, 3(2), 157-166. <https://doi.org/10.1021/cc0000388>
24. Fourches, D., Muratov, E., Tropsha, A. (2010). Trust, but verify: on the importance of chemical structure curation in cheminformatics and QSAR modeling research. *Journal of Chemical Information and Modeling*, 50(7), 1189-1204. <https://doi.org/10.1021/ci100176x>
25. Tetko, I. V., Gasteiger, J., Todeschini, R., Mauri, A., Livingstone, D., Ertl, P., & Virtual Computational Chemistry Laboratory (VCCLab). (2005). Virtual computational chemistry laboratory--design and description. *Journal of Computer-Aided Molecular Design*, 19(6), 453-463. <https://doi.org/10.1007/s10822-005-8694-y>
26. Bajorath, J. (2002). Integration of virtual and high-throughput screening. *Nature Reviews Drug Discovery*, 1(11), 882-894. <https://doi.org/10.1038/nrd941>

27. Koutsoukas, A., Paricharak, S., Galloway, W. R., Spring, D. R., & Ijzerman, A. P. (2013). Identification and application of multidimensional quantitative structure–activity relationship models for predicting off-target activities for drug discovery. *Journal of Chemical Information and Modeling*, 53(6), 1307-1322.
28. Dudley, J. T., Deshpande, T., & Butte, A. J. (2011). Exploiting drug-disease relationships for computational drug repositioning. *Briefings in Bioinformatics*, 12(4), 303-311. <https://doi.org/10.1093/bib/bbr013>
29. Xie, L., Evangelidis, T., Xie, L., & Bourne, P. E. (2018). Drug discovery using chemical systems biology: identification of the protein–ligand binding network to explain the side effects of CETP inhibitors. *PLoS Computational Biology*, 14(1), e1005939.
30. Corsello, S. M., Bittker, J. A., Liu, Z., Gould, J., McCarren, P., Hirschman, J. E., & Roth, F. P. (2017). The Drug Repurposing Hub: a next-generation drug library and information resource. *Nature Medicine*, 23(4), 405-408. <https://doi.org/10.1038/nm.4306>
31. Landrum, G. (2006). RDKit: Open-source cheminformatics. <http://www.rdkit.org>
32. Wang, Y., Bryant, S. H., Cheng, T., Wang, J., Gindulyte, A., Shoemaker, B. A., ... & Roth, B. L. (2019). PubChem BioAssay: 2017 update. *Nucleic Acids Research*, 45(D1), D955-D963. <https://doi.org/10.1093/nar/gky1033>
33. Olanow, C. W., & Schapira, A. H. (2013). Therapeutic prospects for Parkinson disease. *Annals of Neurology*, 74(3), 337-347. <https://doi.org/10.1002/ana.24011>

34. Brodie, M. J., Perucca, E., & Ryvlin, P. (2007). Comparison of antiepileptic drugs in patients with epilepsy. *JAMA*, 298(6), 626-638.
<https://doi.org/10.1212/01.wnl.0000252941.50833.4a>
35. Birks, J. (2006). Cholinesterase inhibitors for Alzheimer's disease. *Cochrane Database of Systematic Reviews*, (1), CD005593.
<https://doi.org/10.1002/14651858.CD005593>
36. Compston, A., & Coles, A. (2008). Multiple sclerosis. *The Lancet*, 372(9648), 1502-1517. [https://doi.org/10.1016/S0140-6736\(08\)61620-7](https://doi.org/10.1016/S0140-6736(08)61620-7)
37. Giovannoni, G. (2016). Disease-modifying treatments for early and advanced multiple sclerosis: a new treatment paradigm. *Current Opinion in Neurology*, 29(3), 1-8. <https://doi.org/10.1097/WCO.0000000000000561>
38. Miller, R. G., Mitchell, J. D., & Lyon, M. (2012). Riluzole for amyotrophic lateral sclerosis (ALS)/motor neuron disease (MND). *Cochrane Database of Systematic Reviews*, (3), CD001447.
<https://doi.org/10.1002/14651858.CD001447.pub3>

