International Hands-on Workshop
on
Artificial Intelligence in Drug Discovery

(21st June 2025, 10:00 am to 5:00 pm IST)

User manual

for

Workshop on AI/ML in Drug Discovery

1. Introduction

Welcome to this comprehensive tutorial on molecular docking using AutoDock Vina, enhanced with machine learning (ML) techniques for binding energy prediction. This workflow is designed for researchers and students in computational chemistry and drug discovery, providing a step-by-step guide from environment setup to advanced analysis.
In this tutorial, you will learn how to:

- Set up your computational environment and install all necessary tools.
- Prepare and upload protein and ligand files in the correct formats.
- Perform molecular docking simulations using AutoDock Vina.
- Analyze and interpret docking results.
- Train a Graph Convolutional Network (GCN) model for predicting binding energies.
- Make predictions on new ligands using the trained ML model.
- Integrate docking and ML results for comprehensive compound evaluation.

This notebook is structured to be accessible for both beginners and experienced users, with detailed explanations and practical tips at each step.

1.1 Setup Instructions:

a) Download the project files: Go to https://github.com/SilicoScientia/AI-ML-Workshop/archive/refs/heads/main.zip to download the complete workshop materials as a zip file.
b) Extract and locate the ML files: After downloading, unzip the folder. You'll find two subfolders - "ML" and "AutoDockVina". Navigate to the ML subfolder.
c) Upload files to Colab: Open the provided Colab notebook at https://colab.research.google.com/drive/1UhtkvglAFv1-NmEuX4XwiNIDB1EW2vmk#scrollTo=RXchmRP3Ad38 and upload these three files from the ML folder:

- requirements.txt
- train_gcn.py
- predict_gcn.py

+91-97405 00475
hr@silicoscientia.com
Nagananda Commercial Complex
Jayanagar 9th Block, Bengaluru – 560 041

Once all three files are uploaded to your Colab environment, you're ready to begin the tutorial.

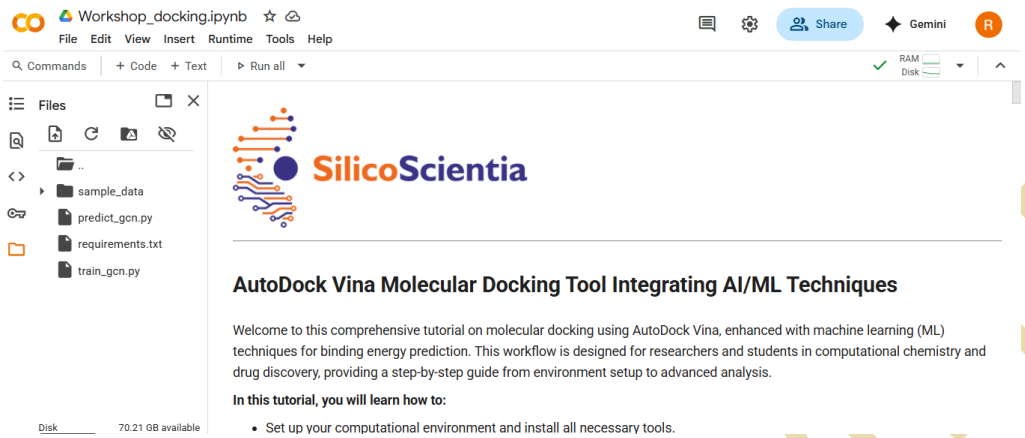

Figure 1. Beginning the tutorial

2. Installing Machine Learning Dependencies

Before starting the molecular docking and ML workflow, we need to install several essential Python packages:

- PyTorch: A powerful deep learning framework used for building and training neural networks.
- RDKit: A cheminformatics toolkit for working with molecular structures and generating molecular graphs.
- NetworkX: A library for the creation, manipulation, and study of complex networks (graphs).
- Other ML-related dependencies: Additional packages required for data processing and model training.

Why are these packages important?

- They provide the computational backbone for implementing and running the GCN model, which predicts binding energies based on molecular structure.

Note: Installation commands may take a few minutes to complete and requires a runtime restart.

3. Importing Essential Libraries

In this section, we import the core Python libraries required for the molecular docking workflow:

- `os`: For file and directory operations.
- `subprocess`: To execute external commands, such as running AutoDock Vina.
- `pandas`: For data manipulation and analysis.
- `pathlib`: For handling file paths in a platform-independent way.
- `glob`: For file pattern matching and retrieval.
- `google.colab.files`: For uploading and downloading files in the Colab environment.
- `shutil`: For high-level file operations.
- `rdkit`: For molecular structure handling and visualization.
- `re`: For regular expression operations.

These libraries form the foundation for file management, data processing, and interaction with external software throughout the workflow.

4. Installing AutoDock Vina

AutoDock Vina is a widely used open-
source software for molecular docking and virtual screening. In this step, we will:

a) Download the AutoDock Vina binary (version 1.2.3) suitable for your operating system.
b) Make the binary executable.
c) Move the binary to a directory included in your system's PATH for easy access.

Why AutoDock Vina?

- It uses a sophisticated scoring function to predict the binding modes and energies of small molecules (ligands) to protein targets, making it a cornerstone tool in computational drug discovery.

Note: Installation may require administrative privileges and internet access.

5. Creating Necessary Directories

To keep your workflow organized, we will create a structured set of directories:

- `protein/`: Stores the target protein file(s).
- `ligands/`: Stores all ligand files to be docked.
- `results/`: Stores the output and results from docking simulations.

Why is this important?

- A well-organized directory structure helps manage multiple files, prevents confusion, and ensures a smooth workflow, especially when handling large datasets or batch processing.

6. Uploading Input Files

This section guides you through uploading the essential input files for molecular docking:

a) Protein File Upload:

- Upload your prepared protein file in PDBQT format.
- The file will be automatically moved to the `protein/` directory.

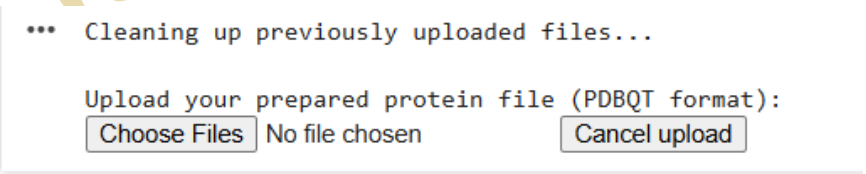

Figure 2. Choose the prepared protein.pdbqt file

b) Ligand Files Upload:

- Upload one or more ligand files in PDBQT format.
- All ligand files will be moved to the `ligands/` directory.

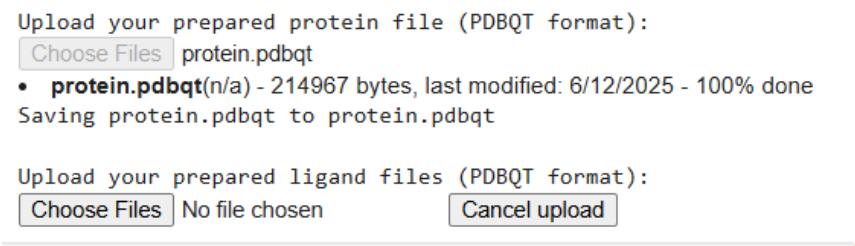

Figure 3. Choose the prepared ligands file in .pdbqt format

Why PDBQT format?

- Both protein and ligand files must be in PDBQT format, which includes information about atom types and partial charges required for docking.

Tip: Ensure your files are properly prepared and named for easy identification later in the workflow.

7. Visualizing 2D Structures of Ligands

In this step, we will:

- Convert ligand files from PDBQT to PDB format using OpenBabel.
- Generate 2D molecular structures for each ligand using RDKit.
- Display all ligands in a grid for visual inspection.

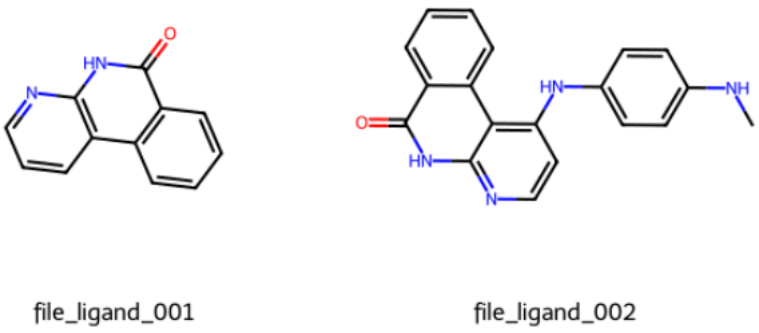

Figure 4. Visualise the 2D structures of the prepared ligands

Why visualize ligands?

- Visual inspection helps verify the integrity and diversity of your ligand set before proceeding with docking.
- It can also reveal potential issues with file preparation or molecular structure.

Note: This step requires OpenBabel and RDKit to be installed.

8. Configuring Docking Parameters

Before running molecular docking, we need to define key parameters:

Binding Site Coordinates:

- `center_x`, `center_y`, `center_z`: Specify the center of the binding site on the protein.
- `size_x`, `size_y`, `size_z`: Define the size of the search space (box) for docking.
- Docking Settings:
- `num_poses`: Number of binding poses to generate for each ligand.

How to choose these values?
- The coordinates and box size should be based on the known or predicted binding site of your protei n target.
- Adjust these parameters to ensure the search space covers the entire binding pocket.

Tip: Use visualization tools or literature data to determine appropriate values for your system.

9. Executing Molecular Docking

This section performs the core docking simulations:
a) File Verification:
- Check the existence of protein and ligand files.
- List all available ligand files for docking.
b) Docking Function:
- Define a function to run AutoDock Vina for each ligand.
- Configure docking parameters and execute the docking command.
- Capture and process the output for each ligand.
c) Batch Processing:
- Iterate through all ligand files and run docking.
- Store results for further analysis.

Why automate docking?
- Batch processing enables efficient screening of large ligand libraries, saving time and reducing man ual errors.

Note: Docking may take several minutes per ligand, depending on system resources and box size.

10. Analyzing Docking Results

After docking, we need to process and analyze the results:
a) Output Organization:
- Create directories for best poses and all poses.
- Extract binding energies from Vina output.
- Separate the best pose from all generated poses.
b) Data Processing:
- Parse Vina output to extract binding energies.
- Create a summary of results in CSV format.

- Organize output files for easy access and downstream analysis.

c) Results Summary:

- Generate a comprehensive CSV file with all docking results.
- Include binding energies and pose information for each ligand.

docking_results.csv

1 to 2 of 2 entries Filter

ligand	output	binding_energies	best_energy
file_ligand_001.pdbqt	results/file_ligand_001_out.pdbqt	[-7.367, -7.168, -6.964, -6.882, -6.725, -6.635, -6.435, -6.289, -6.272, -6.252]	-7.367
file_ligand_002.pdbqt	results/file_ligand_002_out.pdbqt	[-9.147, -8.558, -8.543, -8.522, -8.431, -8.182, -8.084, -8.029, -7.995, -7.965]	-9.147

Figure 5. Summary of the docking results along with best binding energy

Why analyze results?

- Identifying ligands with the best binding energies is crucial for prioritizing compounds for further study.

Tip: Use the CSV summary for quick comparison and visualization of docking results.

11. Downloading Results

To facilitate further analysis and sharing, this section prepares and downloads the docking results:

a) Results Packaging:

- Create a zip archive containing all output files, including best poses, all poses, and analysis results.

b) File Downloads:

- Provide the complete results package as a downloadable zip file.
- Offer direct download of the CSV summary for quick access.

Why package results?

- Packaging results ensures all relevant files are easily accessible and transferable for downstream analysis or collaboration.

Tip: Keep a backup of your results for reproducibility and future reference.

12. PLIP Docking Analysis

PLIP (Protein-Ligand Interaction Profiler) is a powerful tool for analyzing protein-ligand interactions. In this section, we will:

a) Install PyMOL:

- PyMOL is required for visualizing and analyzing protein-ligand complexes.
- The installation process may take some time and requires a runtime restart.

b) Create Complexes:

- Convert docking results into protein-ligand complexes.
- Prepare files for PLIP analysis.

Why use PLIP?

- PLIP provides detailed insights into protein-ligand interactions, including hydrogen bonds, hydrophobic contacts, and other interaction types.
- This analysis helps validate docking results and understand binding mechanisms.

Note: PyMOL installation is a one-time setup that may require runtime restart.

13. PLIP Installation and Setup

This section covers the installation and configuration of PLIP:

a) Install OpenBabel:

- Required for file format conversions.
- Ensures compatibility with various molecular file formats.

b) Clone and Install PLIP:

- Clone the PLIP repository from GitHub.
- Modify installation requirements for compatibility.
- Install PLIP for command-line usage.

Why OpenBabel?

- OpenBabel is essential for converting between different molecular file formats, ensuring compatibility throughout the workflow.

Tip: Ensure all dependencies are correctly installed before proceeding.

14. Processing PDB Files with PLIP

In this section, we will:

a) Check PDB Files:

- Verify the existence of PDB files in the specified directory.
- List all available PDB files for analysis.

b) Process PDB Files:

- Run PLIP analysis on each PDB file.
- Generate detailed reports and visualizations.
- Organize results in a structured directory.

```
Binding sites for file_ligand_001:
```

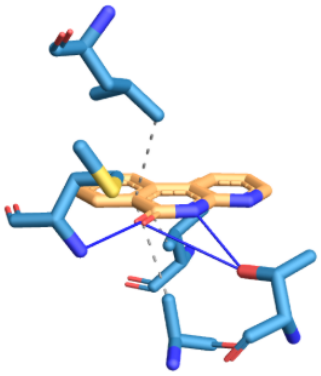

Figure 6. Structural representation of protein-ligand binding interactions using PLIP tool

What to expect from PLIP analysis?

- Detailed reports on protein-ligand interactions.
- Visualizations of binding sites and interaction types.
- Comprehensive summaries for each complex.

Note: Processing time depends on the number and size of PDB files.

15. Developing a GCN-based Binding Energy Prediction Model

This section introduces the machine learning component of the workflow:

a) Model Overview:

- Graph Convolutional Network (GCN) for binding energy prediction.
- Uses molecular graph representation of ligands.
- Predicts binding energies based on molecular structure.

b) Implementation:

- Training the GCN model on docking results.
- Making predictions for new ligands.
- Integrating ML predictions with docking results.

Why use GCN?

- GCNs are well-suited for molecular data, capturing structural information effectively.
- They enhance docking results by providing additional insights into binding energies.

Tip: Ensure sufficient training data for robust model performance.

16. Training the GCN Model

This section covers the training process of the GCN model:

a) Data Preparation:

- Convert docking results to molecular graphs.
- Prepare training and validation datasets.
- Extract features from molecular structures.

b) Model Training:

- Set up the GCN architecture.
- Train the model on docking results.
- Validate model performance.

What to monitor during training?

- Loss and accuracy metrics.
- Overfitting and underfitting signs.
- Model convergence.

```
Epoch 170/200, Train Loss: 0.1017, Val Loss: 3.1043
Epoch 180/200, Train Loss: 0.0154, Val Loss: 2.4477
Epoch 190/200, Train Loss: 0.1910, Val Loss: 2.8067
Epoch 200/200, Train Loss: 0.0852, Val Loss: 2.8707

Loading best model for evaluation...

Evaluating model performance...

Model Evaluation Results:
------------------------
Mean Squared Error (MSE): 1.8119
Root Mean Squared Error (RMSE): 1.3461
Mean Absolute Error (MAE): 1.0285
R-squared (R²): -1.2874

Results have been saved to the 'model_output' directory

Training and evaluation completed successfully!
<Figure size 640x480 with 0 Axes>
```

Figure 7. Evaluation results of the GCN model

Note: Training time depends on dataset size and model complexity.

17. Making Predictions with the GCN Model

This section executes the prediction process for new ligands:

a) Prediction Pipeline:

- Load the trained GCN model.
- Process new ligand files.
- Generate binding energy predictions.

b) Results Analysis:

- Compare ML predictions with docking results.
- Provide confidence scores.

- Generate comprehensive reports.

```
Predictions saved to predicted_binding_energies.csv

Prediction Summary:
Total ligands processed: 11
Successful predictions: 11
Failed predictions: 0

Top 5 predicted binders (lowest energy):
                ligand  predicted_energy
file_ligand_006.pdbqt         -8.118565
file_ligand_008.pdbqt         -7.893367
file_ligand_010.pdbqt         -7.718667
file_ligand_011.pdbqt         -7.718667
file_ligand_001.pdbqt         -7.527202
```

Figure 8. Summary of predictions results of new ligands

Why combine ML with docking?

- ML predictions can provide additional insights and validation for docking results.
- They help prioritize compounds for further study.

Tip: Use confidence scores to assess prediction reliability.

18. Filtering Compounds Based on Threshold

This section guides you through filtering compounds based on binding energy thresholds:

a) Set Energy Threshold:

- Use a slider to set the binding energy threshold.
- Filter compounds with binding energies below the threshold.

b) Convert to SMILES:

- Convert filtered compounds to SMILES format.
- Prepare for ADMET analysis.

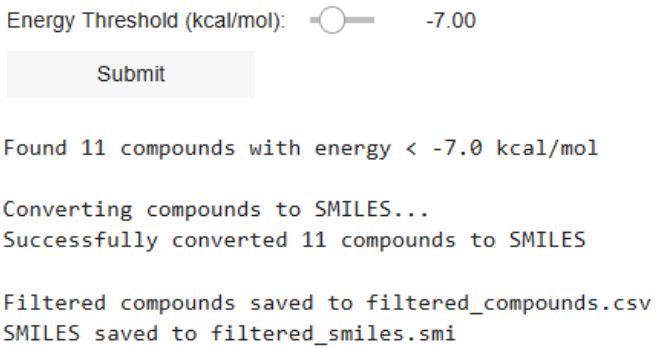

Figure 9. Adjust the binding energy threshold to filter the potential compounds

Why filter compounds?
- Filtering helps focus on the most promising compounds.
- It reduces the number of compounds for further analysis.

Tip: Adjust the threshold based on your specific research goals.

19. ADMET-AI Analysis

This section introduces ADMET-AI for predicting drug-like properties:
a) Install and Initialize ADMET Model:
- Install the ADMET-AI package.
- Initialize the model for predictions.

b) Run Predictions:
- Predict ADMET properties for filtered compounds.
- Save results for further analysis.

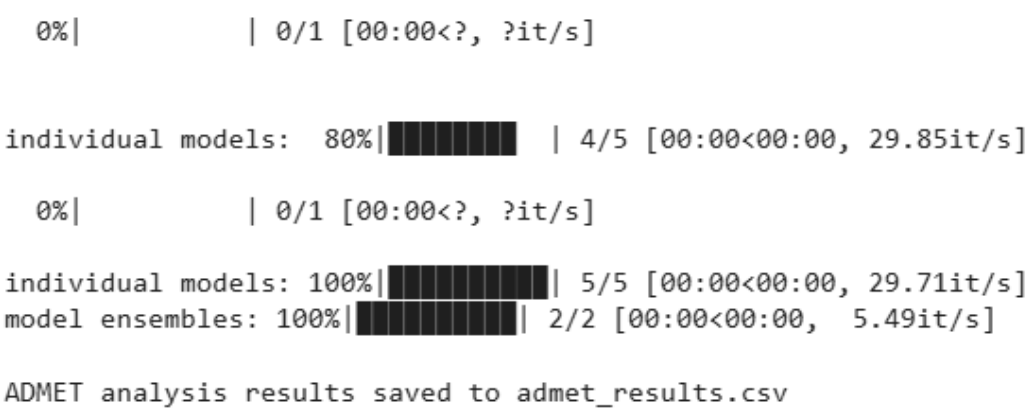

Figure 10. ADMET properties prediction using ADMET-AI tool on the filtered compounds

Why use ADMET-AI?
- ADMET properties are crucial for drug development.
- They help identify compounds with favorable pharmacokinetic profiles.

Note: Ensure the ADMET-AI package is correctly installed and configured, need a runtime restart.

20. Filtering Compounds After ADMET Predictions

This section guides you through filtering compounds based on ADMET properties:

a) Set ADMET Thresholds:

- Define thresholds for key ADMET properties.
- Filter compounds based on these thresholds.

```
Total compounds in ADMET results: 11

ADMET Property Thresholds Configuration
========================================
Enter new values for each property (press Enter to keep default value)

molecular_weight (default: 500 <):

logP (default: 5.0 <):

hydrogen_bond_acceptors (default: 10 <):

hydrogen_bond_donors (default: 5 <):

tpsa (default: 140 <):

Lipinski (default: 0.5 >):

QED (default: 0.5 >):
```

Figure 11. Configuration of the ADMET parameters to filter compounds based on these thresholds

b) Visualize Filtered Compounds:

- Generate 2D structure visualizations.
- Display key properties for each compound.

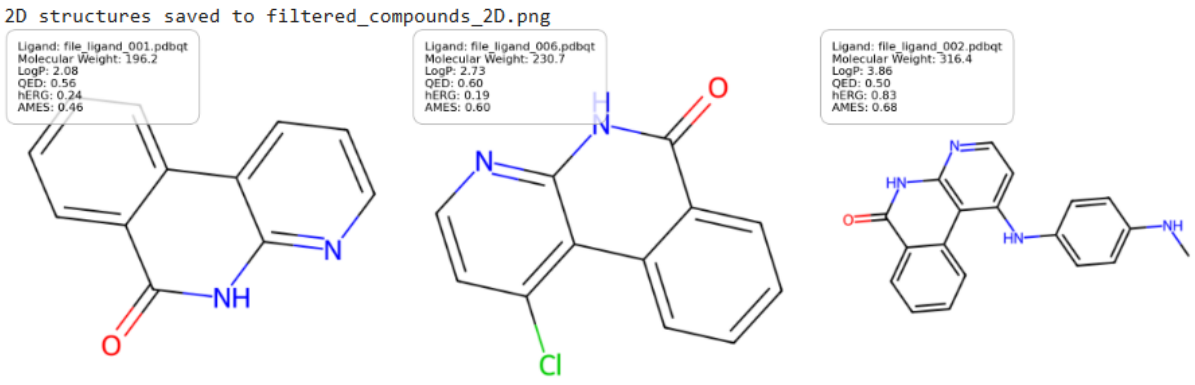

Figure 12. 2D structure visualizations of the final filtered compounds

Why filter based on ADMET properties?

- Ensures compounds have favorable drug-like properties.
- Reduces the risk of failure in later stages of drug development.

Tip: Use visualizations to quickly assess compound properties.

21. Docking for Filtered Compounds

This section performs docking for the filtered compounds:

a) Run Docking:
- Use AutoDock Vina to dock filtered compounds.
- Generate binding poses and energies.

b) Analyze Results:
- Extract best poses and binding energies.
- Compare with previous docking results.

```
Docking results saved to filtered_docking_outputs/docking_results.csv
Best poses saved to filtered_docking_outputs/best_poses
All poses saved to filtered_docking_outputs/all_poses

Summary of best binding energies:
                  ligand  best_energy  predicted_energy
3  file_ligand_004.pdbqt       -8.961         -7.392422
2  file_ligand_002.pdbqt       -8.690         -7.250137
1  file_ligand_006.pdbqt       -7.557         -8.118565
0  file_ligand_001.pdbqt       -7.355         -7.527202
4  file_ligand_005.pdbqt       -7.302         -7.478648
```

Figure 13. Docking of the compounds filtered from the ADMET analysis

Why dock filtered compounds?
- Validates the filtering process.
- Ensures compounds have favorable binding modes.

Note: Docking may take some time, depending on the number of compounds.

22. PLIP for Filtered Compounds

This section analyzes protein-ligand interactions for filtered compounds:

a) Create Complexes:
- Convert docking results into protein-ligand complexes.
- Prepare files for PLIP analysis.

b) Run PLIP Analysis:
- Generate detailed reports and visualizations.
- Organize results in a structured directory.

```
Binding sites for file_ligand_002:
```

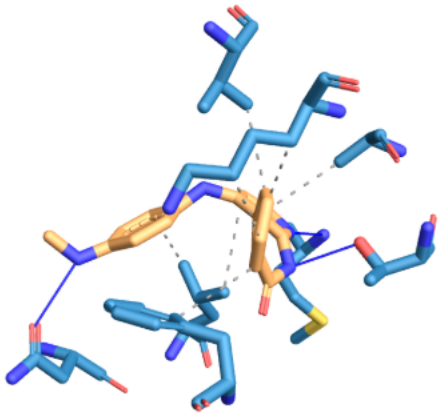

Figure 14. Structural representation of protein-ligand binding interactions of the final selected compounds using PLIP tool

Why use PLIP for filtered compounds?

- Provides detailed insights into binding interactions.
- Helps validate docking results and understand binding mechanisms.

Tip: Use PLIP visualizations to identify key interaction types.

Conclusion

Congratulations on completing this tutorial! You've learned how to perform molecular docking, analyze results, and integrate machine learning for binding energy predictions.

Key Takeaways:

- Perform docking with AutoDock Vina and analyze interactions with PLIP.
- Use machine learning to predict binding energies and filter compounds based on ADMET properties.

Next Steps:

- Experiment with parameters and models to optimize results.
- Explore additional tools to enhance your drug discovery pipeline.

Thank you for following along! If you have any questions, feel free to reach out.

Happy Docking and Discovering!