# AI in Drug Discovery and Development

**Prof. Rajnish Kumar**
**Dept. of Pharmaceutical Engineering and Technology**
**IIT-(BHU), Varanasi**

**Week-03**
**Lecture-15**

Welcome to the course "AI in Drug Discovery and Development." In today's session, we will talk about Python libraries for AI in drug discovery. You know, Python has become quite a popular tool that has been used extensively for every task. And drug discovery is not untouched either. So, in drug discovery and development, there are hundreds of Python libraries that are being used. So, in this session, we will cover a couple of them and see how they can be used and what they can do.

So, at the end of this lecture, you will be able to understand the importance of Python and its ecosystem in advancing AI-driven drug discovery. Identify key categories of Python libraries used in cheminformatics, bioinformatics, molecular modeling, docking, machine learning, and visualization. And describe the core functionalities and applications of commonly used libraries, as well as apply Python libraries to typical drug discovery workflows. Okay, so why Python for drug discovery? Because it's kind of open source, powerful, and flexible; it is free to use and customize.

One can use it for free, and we can even customize it well. And it has extensive support for scientific computing and data manipulation, and it is adaptable to different drug discovery tasks as well, like cheminformatics, bioinformatics, or ML. And then another important thing is that it has a large and active scientific community where you receive continuous updates, improvements, and support. It has a rich pool of tutorials, forums, and collaborative projects. And it is also easy to find solutions and best practices.

And then we have a wide range of specialized libraries. Like we have libraries for molecular modeling, simulation, machine learning, and visualization. And some of the examples are like RDKit, Scikit-learn, DeepChem, and Py3Dmol. And then it has been a very seamless integration with data science, AI, and simulation. So, it combines chemical data with ML models easily, and we have smooth workflows for simulations, molecular modeling, docking, and prediction tasks.

Also, these are interoperable with high-performance computing tools like Gromacs, Autodock, and TensorFlow. If you are talking about Python libraries for drug discovery and development, we can broadly classify them based on the tasks we use. Like we can have cheminformatics libraries which like RDKit, OpenBabel, MolVS and then we can have the bioinformatics library like BioPy thon and then we have molecular modeling libraries like MDTraj and ParmED. And then we have molecular docking libraries such as PyRx and Meeko. And then we have, you know, ML and AI libraries like Scikit-learn, TensorFlow, and PyTorch.

And then we have visualization libraries for visualizing that molecular data. We can use py3dmol, matplotlib, seaborn, or nglview. And then some of the miscellaneous libraries handling the numerical data, like NumPy and Pandas, actually. So let us start with the RDKit, which is becoming one of the most popular cheminformatics libraries. So it's a core toolkit for modern cheminformatics, and it is a widely used open-source collection of cheminformatics and machine learning tools designed to support the representation, manipulation, and analysis of chemical information.

So the key capabilities of RDKit are that you can use it for parsing and handling molecular structures from various formats; it can handle SMILES, SDF, MOL2, PDB, and all those sorts of data files. And then you can use it for substructure searching, pattern matching, and molecular similarity assessment. You can, you know, do the 2D and 3D molecular structure generation, depiction, and optimization. Even you can calculate several features, such as molecular descriptors and fingerprints, like Morgan fingerprints, for data modeling and machine learning tasks. And then it has, you know, and it can handle chemical reactions, including reaction smiles parsing and reaction-based enumeration as well.

So some of the applications where we can use it are virtual screening and chemical library filtration. If you want to prepare molecules from large libraries, we can use it to prepare those molecules, and we will see that in hands-on sessions as well. We will use RDKit to prepare molecules for, you know, performing all those tasks. And then you can use it for feature generation for machine learning models, such as developing QSAR and QSPR models. And then it can be used for hit-to-lead optimization, property prediction, and chemical informatics analysis.

And then the development of custom cheminformatics pipelines and workflows as well. And this is just a quick example where you can see that you can create a MolFromSmiles, and then you can get the number of atom counts; you can calculate all those properties and everything. Another cheminformatics library is OpenBabel. So, it is a free, open-source chemical toolbox designed to facilitate the interconversion of chemical data between various file formats. So, it serves as a cross-platform program and library supporting molecular modeling, computational chemistry, and related areas.

So, whenever we are talking about virtual screening, we need to prepare the molecules, the libraries containing molecular structures, and Open Babel can be very helpful in that case. So, what it can do is that it has the capability to convert the formats. So, it supports conversion between over 100 chemical file formats including smiles, pdb, mol2, inchi and then you can use it for geometry optimization. It provides geometry optimization using built-in force fields. As well as that, we can calculate some basic descriptors, such as molecular weight and logP.

And then you can use it for substructure search, fingerprinting, and even filtration, as well as filtering molecules based on properties like Lipinski's rule of five if you want to filter your library of molecules, so you can use Open Babel as well. So, it enables substructure searching and fingerprint generation for molecular similarity analysis as well. So the typical applications of OpenBabel are preparing the libraries for docking and simulations. You can convert and prepare molecular files for use in docking studies and molecular simulations. And then you can use it for pre-processing the files, molecular data for machine learning as well, where it can process molecular information to generate descriptors and fingerprints suitable for machine learning pipelines.

Okay, coming to another library, which is called MolVS or Molecule Validation and Standardization. Again, it is an open-source Python library built on top of RDKit and designed to clean, standardize, and validate chemical structures. And it ensures chemical data consistency, which is critical for applications such as virtual screening, cheminformatics, analysis, and machine learning. So, the core capabilities are tautomer standardization and standardizing the tautomers of a common form for consistent representation. It can perform formal charge normalization, adjusting the formal charge to chemically reasonable defaults.

And you can, you know, remove the salts and counter ions, which removes unwanted small fragments such as salts and solvents. And then molecule canonicalization, where it produces a consistent canonical molecular representation for reliable comparison and indexing. And all of these are very important steps, you know, for preparing a ligand library. Because we have actually discussed those issues, those molecules might have some problems with the tautomers, stereochemistry, or charge, right. So, we have to process molecules properly, and these libraries can be quite helpful in that.

And the typical application could be chemical database cleaning, where it can prepare chemical libraries by standardizing molecules before virtual screening or database mining. And then, input standardization for ML and descriptor calculation can ensure consistent molecular structures for accurate descriptive generation and machine learning modeling. So, this is just standardizing these files. So, where do you get the standardized canonical smile? And then we have BioPython, which is a core toolkit for bioinformatics in Python. It is a freely available open-source collection of Python tools designed to facilitate computational biology and bioinformatics.

It provides extensive functionality for working with biological data, making it one of the foundational libraries for bioinformatics programming. So, what it can do is parse different biological file formats, as it supports a wide range of biological formats such as FASTA, GenBank, and PDB. We can do the sequence analysis; it enables sequence alignment, translation, transcription, and manipulation of DNA, RNA, and protein sequences. We can access the databases as well. It provides tools to query and retrieve biological information from online databases such as NCBI, ExPASy, and others.

And we can perform analyses like BLAST, which is, of course, done with the help of those databases. And then it can handle the molecular structures as well, where it supports reading, writing, and manipulating the 3D structures of biomolecules. Some of the typical applications are that you can use it for DNA, RNA, and protein sequence manipulation, where we can analyze and modify biological sequences for research and application. It can be used for data retrieval from biological databases, allowing us to fetch gene, protein, and nucleotide data programmatically. And then we can use it to prepare the data for docking and structure prediction.

We can process biological sequences and structures for modeling, docking, and computational studies. And here is just a snapshot of how we can do the BLAST analysis for a nucleotide sequence obtained from your experiments, actually. And then we have the ParmEd, which is a powerful and flexible Python library designed to facilitate the editing, manipulation, and conversion of molecular structures, topology, and force field parameters. So it acts as a bridge between different molecular dynamics simulation packages, enabling seamless system preparation modification. So whenever we are running a molecular dynamics simulation, we need to generate the topologies of the molecules and the force field parameters.

And ParmED can be quite helpful in that regard. So the core capabilities of ParMED are that it can handle various file formats, as it can read, edit, and write topology and parameter files from major MD engines like Amber, Charmm, Gromacs, and OpenMM. We can also do the force field modification, where it can adjust force field parameters such as atomic charges, bond lengths, angles, and torsions. We can use it for system preparation for MD, where it can build and modify systems for MD simulation, including adding restraints, modifying solvent environments, and combining molecular systems. And we can use it for advanced system editing, where it can strip solvents or ions, merge or split systems, and fine-tune the force field parameters as needed.

And some of the typical applications are editing and customizing simulation input. We can modify a prepared input file for compatibility or optimization across different MD engines and automate system preparation, allowing us to streamline the setup process for large batches of systems in high-throughput simulation. And we can, you know, do the force field customization where we can tailor the force fields, such as adjusting the ligand parameters to meet the specific simulation needs. And this is a snapshot of how we can actually use the ParmED. And then MD-TRAJ is again related to molecular dynamics.

So MD-TRAJECTORY. So it's a fast open source Python library designed for analyzing molecular dynamics simulation trajectories. So it provides a simple yet powerful interface for loading, processing, and analyzing the large MD datasets, making it a key tool for computational chemistry, structure, biology, and biophysics. So what it can do is handle the trajectory. It can read and write a variety of trajectory formats, including DCD, XTC, TRR, PDB, and others. And then you can do the structural analysis; you can compute structural properties such as RMSD, radius of gyration, and hydrogen bond analysis.

You can perform frame extraction visualization, extract specific frames, and visualize molecular movements across a trajectory. And you can do the advanced analysis as well, where you can perform tasks like clustering confirmations and generating contact maps to study the molecular interactions. And the typical applications could be post-processing of MD simulation, where you can analyze and interpret molecular dynamics output. And you can use it for snapshot extraction for docking or visualization, where you can select the representative structures for downstream studies such as docking or structural visualization. And then for the quantification of structural changes, you can track and measure molecular conformation changes over the simulation time.

And this is an example snapshot to calculate the hydrogen bonds between the molecules in the molecular dynamics' trajectory. And then you have Meeko. So, Meeko is another beautiful library that is extensively powerful in helping with molecular docking studies. So Meeko is an open-source Python library specifically designed to automate and simplify the preparation of molecules for docking with the AutoDock suite. Built on top of RDKit, it enables the efficient generation of AutoDock-compatible input files, making it ideal for both manual and high-throughput docking workflows.

So, where the core compatibilities are, we can prepare the molecules, add missing hydrogen atoms, calculate the Gasteiger partial charges, and generate the 3D structure from SMILES strings. And then we can generate AutoDock-specific files as well, where we can convert the molecules to PDBQT format, assign proper atom types, identify and assign rotatable bonds for flexible docking, and prepare both ligands and receptors for docking. And it has a user-friendly API that provides simple high-level functions to automate molecular preparation, reducing the manual steps. And then the typical application could be ligand preparation, where it can convert files to 3D structures and prepare fully dockable PDBQT files. And then we can use it for receptor preparation as well, where we can process and adjust protein structures for compatibility with the AutoDock tools.

And then we can use it for automation in the docking pipeline as well, where it can seamlessly integrate into batch processing workflows for virtual screening and drug discovery projects. And then we have one of the most famous libraries, which is scikit-learn. It's one of the most popular and trusted open source libraries for classical machine learning in Python. It provides a simple, efficient tool for data mining, data analysis, and machine learning, emphasizing performance, scalability, and ease of use. So what it can do is be used for classification and regression modeling.

It can build predictive models such as QSAR and QSPR to estimate biological activities and chemical properties based on molecular descriptors or fingerprints. We can use it for clustering, where we can apply clustering algorithms like k-means or DBSCAN. To identify patterns in chemical datasets or to group structurally or functionally similar compounds. We can do the dimensionality reduction as well, where we can use techniques like PCA or t-SNE to reduce the complexity of high-dimensional molecular feature sets, adding visualization and model performance. Or we can use it for model evaluation optimization, where it has comprehensive tools for cross-validation, hyperparameter tuning, and evaluation metrics like ROC curve, precision, recall, confusion matrices, and scoring functions.

So, there is no doubt that it is one of the most powerful ML libraries and is extensively used as well. So, the typical applications could include developing predictive models for drug discovery and repurposing. As well as exploring chemical space, clustering compounds, reducing feature space for more interpretable and efficient models, and fine-tuning machine learning models to maximize predictive performance. Okay, and then we have deep learning libraries like TensorFlow and PyTorch. So, these are two of the leading deep learning frameworks widely used in drug discovery for building and deploying advanced neural network models, both of which are powerful tools for molecular modeling, drug design, and predictive analytics.

So the core capabilities, you know, can be used for making deep learning models. Both frameworks support training neural networks for molecular property prediction, protein-ligand binding, affinity, and drug design. We can use them for generative models as well. We can use models like generative adversarial networks and variational autoencoders for molecule generation and property optimization. And then we can do the transfer learning as well, where we can fine-tune pre-trained models for drug discovery tasks, leveraging existing knowledge to accelerate the research.

And we can build an end-to-end workflow, build, train, and deploy deep learning models seamlessly in both TensorFlow and PyTorch, supporting diverse applications. So, in drug discovery and development, we can use molecule generation optimization, molecule property prediction, and protein-ligand interaction prediction as well. So, here is a kind of simple neural net for property prediction; the code for that is shown. Okay, then coming to the visualization library. So, py3dmol is a Python wrapper for the 3D Ah Mole dot JS library.

So, it is designed to provide interactive molecular visualization, allowing researchers to visualize and analyze molecular structures directly within the Jupyter notebook and Python environments. We can use it for molecular visualization, such as displaying the 3D structure of molecules using different formats like PDB, SMILES, and others. We can use it for an interactive display where it can interactively visualize a 3D structure, enabling zooming, rotating, and manipulating the molecular structures in real time. And then we can have customizable visuals as well. We can customize molecular views with color schemes such as styles like sticks or spheres and annotations.

And then integration with the Python notebook can be seamlessly integrated with the Jupyter notebook, making it easy to visualize molecular data alongside the computational analysis. And the typical applications could be visualizing protein-ligand interactions, displaying molecular dynamics simulation results, and enhancing molecular docking analysis with interactive 3D views. Another visualization library is NGLView. So, it is used for interactive visualization of molecular structures and is especially suited for use in Jupyter notebooks, and it integrates with popular libraries like PyMol and Chimera, allowing for efficient visualization of proteins, nucleic acids, and other molecular structures. So, the core capabilities of NGL view are molecular visualization, as it supports a variety of file formats including PDB, SDF, and others for visualizing 3D structures.

And then we can have a customizable representation that allows for the customization of visuals with different styles like sticks, spheres, or cartoons and color schemes. And we can have an interactive display as well, where it provides real-time interaction with 3D molecular structures, such as zooming, rotating, and translating. And then it is highly compatible with Jupyter Notebook as well. And we can use it for visualizing protein-ligand interactions, displaying molecular dynamics simulation data, and analyzing large molecular systems in research as well.

Okay, and then we have Matplotlib and Seaborn. So these are also used for visualization. So these are two most widely used Python libraries for creating high quality data visualization, particularly suited for scientific research. So, Matplotlib offers flexibility in creating a wide range of plots, while Seaborn simplifies complex visualization with enhanced features. So what we can do with Matplotlib is create flexible 2D plots where we can create static, animated, and interactive plots.

And then there is the possibility of customization as well. It has a high degree of control over plot appearance, such as color, style, axis, and more. You can even make interactive plots. And then, with Seaborn, we can do the statistical plotting built on top of Matplotlib; it creates advanced plots like heat maps, violin plots, pair plots, and regression plots. And it simplifies complex visualizations by automating the plotting of statistical relationships with fewer lines of code. And then we can use it to make activity plots, such as visualizing the biological activity of compounds like dose-response curves.

Or we can make property distribution plots where we can display the distribution of molecular properties, and then we can do the correlation analysis as well, where we can visualize the relationship between molecular descriptors or activity. And then we have pandas. So Pandas is a powerful Python library designed for data manipulation and analysis, and it is used for data preparation, data wrangling, and data filtration. So it provides a flexible and efficient data structure such as data frames to handle large data sets, and it is widely used in data science, including drug discovery, for organizing, cleaning, and analyzing data. So, the core capability of pandas is to organize, clean, and analyze structured tabular data.

So, easy data manipulation using data frames—filter, group, merge, and aggregate—we can do all sorts of actions on the data, and it can handle various data formats as well; it can handle CSV, Excel files, JSON, SQL, etc. And how we are using it in drug discovery is that we are using it to manage the large chemical data sets, such as in the form of SMILES, activity values, or ADMET properties. We can filter molecules based on properties like logP, molecular weights, etc. And then we can also merge molecular descriptors with biological data for QSAR modeling, and Pandas is quite powerful in doing all sorts of actions for that. And then we have NumPy; NumPy is a foundational library in Python for numerical computing.

So it offers high-performance operations on large arrays and matrices, making it essential for performing mathematical operations on chemical data. So it is an optimized array object that enables efficient handling of large data sets, especially in computational chemistry and drug discovery. So the core capabilities of numpy include efficient operation on large multidimensional arrays, performing mathematical, statistical, and linear algebra operations, and support for vectorized operations to speed up the calculations. And how we can use it in drug discovery is by using it for handling molecular descriptors and matrices for QSAR modeling. Performing mathematical operations on chemical structures, like normalization and scaling, accelerates the model training process, especially for large datasets.

So, now we have seen some of those important libraries because, as I just said, there are hundreds of them that are regularly or frequently used in drug discovery and development. So, let us see how we can use the one that we have discussed; now we can use it for some kind of drug discovery workflow. So, by using RDKit, we can collect the molecular structures. And then we can import the molecule data, build the chemical library, and by using MolVS, we can standardize the structure by normalizing, cleaning, and standardizing the molecules by removing salts, counter ions, and adjusting the formal charges. And then, by using these visualization libraries like Py3Dmol or NGL View, we can visualize the molecular structures in 3D to understand geometry and interaction potentials.

And then we can use RDKit, Mordred, and Paddle, and there are other libraries as well that we can use to calculate the features or molecular descriptors. We can generate the descriptors and fingerprints, like the molecular Morgan fingerprint, for feature extraction once we have obtained the features. So we can use the pandas you know to handle the data, as we can manage, clean, and manipulate data, like merging molecular descriptors with the bioactivity labels. And then finally, we can use scikit-learn to train and evaluate the model, where we can build and train a machine learning model, for example, a random forest or SVM, to predict the biological activity. We can also use metrics like accuracy, ROC curve, and MAE to evaluate the model's performance.

So, once we have developed the model, we can optimize it; we can identify molecules, and then we can further optimize those molecules using Keras or TensorFlow by optimizing the molecular properties with deep learning models. Molecular docking tools can be used to perform docking of the top hits and to predict the binding affinity to the target protein. And then we can visualize those results with PyMOL or Py3Dmol. And finally, we can generate the predictions using Keras or TensorFlow, where we can use deep learning models for more accurate property prediction and lead optimization. And finally, we can do the statistical analysis using Seaborn and Matplotlib, where we can conduct statistical analysis and generate visualizations like histograms and heat maps to analyze model performance and property distribution.

So this is just a kind of basic workflow where I have shown you how we can use all these libraries together to make a pipeline for drug discovery and development. Okay, let's move on to the summary. So, the Python libraries are crucial for AI driven drug discovery supporting tasks like molecule standardization, descriptor calculation and predictive modeling. There are libraries like RDKit, MolVS, and MordRed that assist in cleaning and generating molecular features, while scikit-learn enables model development. And with that, thank you.