AI in Drug Discovery and Development

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

Week-12
Lecture-58

Welcome to the course AI in Drug Discovery and Development. In earlier hands-on sessions, we saw how we can use RDKit to represent the molecular structure, play with the molecular structure, calculate fingerprints, perform similarity searches, and do all sorts of things. And then in another hands-on session, we saw how we could build a regression-based QSAR model to predict the solubility. And in today's session, we will see how we can use classification models to predict bioactivity. For the hands-on session, we will again use the Colab notebook, so we have to visit this link to open that Colab notebook. So, what you have to do is copy this link and open it in any browser.

As I said earlier, we can use any browser. However, we have to log in to the Gmail account or Google account so that we can access the Colab notebook as well as the servers where we can do the computation. Once you open that link, you will end up on this page. This is a Colab notebook named 12.3 bioactivityprediction.ipynb.

And then in this case, in today's session, as I said, we will build an SVM (Support Vector Machine) based classification model to predict bioactivity. And you know predictive modeling is used a lot for activity prediction, as well as what we saw during the lectures on lead optimization and ADMET prediction. So, in today's session, we will see how we can use SVM, which is a machine learning technique, to predict the bioactivity.

So, first things first, we have to, you know, make a local copy of this. So, for that, you will click on File and then click on Save a copy in Drive, okay? Once you click on Save a copy. So, it will say that it is creating a copyright, and then once it has created a copy. So, it will open in a new tab. Once it is copied, you will see that copy of 12.3 bioactivityprediction.ipynb.

So, we will be using RDKit for this tutorial as well. First, we have to install the dependencies. We will install the RDKit, and then we will run this cell so you can see that it will install the RDKit.
For me, it has already been installed. So, it says the requirement is already satisfied, but it will install RDKit first for you, and after the installation of RDKit, what we will do is run the next cell. So, let us run the next cell, and meanwhile, I will explain what this cell is doing because it will take a little bit of time. So, you just run this cell and the second cell. So now, for any QSAR model building, what we need to have is the training data, right?

And there are, you know, multiple ways from which we can get the training data.

So, there are databases like ChemBL, which is one of them, and BiningDB, which is another database. And then you have, you know, you can always curate your own data as well. Like if you have, for example, your own lab generating some bioactivity data. So, that can be used, or you can even curate the data from the literature as well. In this specific case, we will be using the data from ChemBL.

So, let me give you a brief introduction to what ChemBL is and how we are going to use it. When we open ChemBL in a browser. So, this is the ChemBL website, and we have already discussed that ChemBL is a database that consists of bioactivity data. And you can see that it contains around 2.5 million compounds along with their bioactivity data, details about 1.7 million assays, and it has documents on 16k targets, 15k drugs, and all those things are there in this database.

So what we will be going to do in this case is go for the targets; for example, we will go for the compounds that have shown activity against tau. When you click on targets, you will see all the targets that are listed in this database. So, what we have to do now is select the target that we are interested in. For example, we will go for the MAPT, which is microtubule-associated protein tau.

And when we click on this MAPT TAU, you can see here all the MAPT TAU details about their targets. And they are associated with, for example, ChemBL ID, the name, Uniprot accessories, and Uniprot accession. And then the type, what is it, and then the organism, for example, Homo sapiens or Rattus norvegicus or Bos taurus, right? And then you have, for example, the compounds; how many compounds are associated with them, and then activities, like how many activity data points are there in this database? So, we will choose this tau with humans that has around 95,000 compounds, okay. So, let us click on this ChEMBL ID. So, when we click on this ChemBL ID, it will open the page where you will get all the details about the tau protein.

So you have the ChemBL ID, the type, and the name synonyms. These are synonyms of this protein. And then you have the target components like MAPT tau single protein, and the accession is your Uniprot accession ID. And then you have the relationships, for example, approved drugs, clinical candidates, and an activity chart. So we are interested in the activity chart.

And then you can see here, for example, they have around 95,000 compounds deposited in this database, where most of the compounds have potency values associated with them. Potency means they have some sort of activity, right? In this case, if we are interested in building a QSAR model. So, it will be difficult for us if we are using data that is not well

defined. So, for that, we will use the IC_{50}. So, for IC_{50} we click on if we click on the IC_{50}.

So, then you will see that it will go to the page where we will see the structure of the compounds, the compound key, the IC_{50} standard type, and then if we scroll it a little bit down. So, you can see all those compounds you know, and all these activities you know are actually there so far, and we can change that as well, like we can see. For example, if you wanted to see just five compounds per page, you can see that we have those five compounds. And then, if you drag it to the right, what you will see here is all the details. So, the details about the assay, what it is, what kind of assay they used, and then you see the details about, you know, the target ChemBL ID, the target name, and then the target organism and target type.

And then that ChemBL ID document ID, and then the source of description, and then you can get the reference as well from where these compounds have been curated. So, usually, those who know ChemBL say that it uses automatic curation technology. So, which screen you know the literature like research papers, patents, and publications, and then it curates the data, and then it adds the data into this database. So, now we have got, you know, this detail. So, now we have the compounds, the structure with their ChemBL ID, and then activity with only IC_{50} data.

So, now what we can do is download this using the CSV. For example, if we click on the CSV, we can download all this data. And then we can use this data to make a classification-based model using machine learning. So that is one way. So, another way is to automate this in the Colab itself, and that is what I have done here as well in the second cell that we run.

So, in the second cell, what it is doing is installing the ChemBL web resource client and pandas. So, by using this web resource, what we can do is directly download the data associated with that target. So, this ID, which is ChemBL 1293224, is for the MAPT. So, now, if you change this ID to any other protein, you can download the data for any other protein and any other target as well. So, here we select, we said that, okay, I want to download the data associated with this ChemBL ID, which means this protein, and then in the next code.

So, what we are doing is that we are only interested in fetching the IC_{50}, as I showed you on the ChemBL website. So, the tau MAPT has around 95,000 compounds, and potency is one of the major classes of activity there, but that is not, you know, highly reliable. So, for QSAR model building or predictive analytics, we actually need reliable data. So, what we will do is keep the IC_{50} values. We will filter it to keep only the IC_{50} value type data.

And then there is one more issue whenever you download a dataset from ChemBL. So, what you will see is that even if you have now selected only IC_{50}. So, we will see that the IC_{50} value is less than 50 and greater than 100, like that also you will see. So, we are interested in the compounds for which we have the exact IC_{50} values reported. So, we will only keep entries that have, you know, the standard relation that is equal to; means we will only keep the compounds that have exact IC_{50} values not greater than or less than.

Because that is not reliable and is variable, it is not good for building the predictive model. Okay, so in the next line of code, what we are doing here is just dropping the NA type. If NA is present, we are dropping means like missing values. And in the next cell, what we are doing is fetching the smiles because now we have the smiles and the ChemBL ID molecule. and by using that molecule ChemBL ID we can fetch the smiles of all these compounds so that also we download.

And then in the next line of code, we are classifying them into active and negative. Since in this specific tutorial we are building an ML model to classify the compounds as active or inactive, okay? So, for that, we need to make a training dataset where we classify them into active and inactive. So, that is the only difference between, you know, regression and classification, and both are, of course, supervised learning because we are using labeled data, right? So, we have the data and then we label that, okay, this is the IC_{50} in this case; for example, if the IC_{50} value is less than 1000 nanomolar, okay. So, we say that the compound is active, and if the value is greater than or equal to 10,000, we say that the compound is inactive. The compounds in between, we just exclude.

Because we need to have some margin between the active and inactive compounds, we cannot say that it is okay to have less than 10,000 active and more than 10,000 inactive. So, then it will be very difficult for the model to classify the compounds that are always in that borderline. So, we try to keep a freeman zone where we expect that there is no compound lying there. Okay, so after that, we have classified them into active and inactive, and this classification, this labeling, depends on your data. So, for example, if the spread of the bioactivity data is from maybe 10 nanomolar to 10,000 or 100,000 nanomolar.

So, based on that, you actually have to decide. Usually, what we do is try to keep a kind of equal number of compounds in the active and inactive data sets; only then can you get a good model, okay? And that comes with, you know, trial and error. So, you have to run it multiple times, and then you see which classification, which you know, splits between the active and inactive or labeling, works better for your target. So, in this cell and the next cell, we draw the intermediates and convert them to integers. So, where the activity will be either 0 or 1.

And then we select only the relevant columns to keep because when you download it from ChemBL, you get tens of columns. So, we will keep only the molecule ChemBL ID, SMILES, standard type, standard value, standard units, and activity. And then here I just wanted to see how many of the compounds are labeled as active and how many of them are labeled as inactive. And I just view the top 10 molecules, meaning the starting 10 molecules in the table. So, you can see we selected the cell; when you run this, something like this will appear in your Colab notebook.

So, you selected the target; this was your target, and then the count of actives was, you know, 146, and the count of inactives was 36. So, this is a little bit unbalanced. So, we can, you know, drag the value of, you know, inactives a little bit lower as well. So, we can increase the number of compounds in the inactive as well, and that is for you to actually play with it or to create multiple models. And then when you see the compounds, here you can see that these are the original entries because it has removed those compounds from 1 to 5.

Because those were, they might not have the explicit, accurate, or exact values for IC_{50} data. And then we have kept only the ChemBL ID, the smiles, the standard type IC_{50}, and the standard value, which is the IC_{50} value. And then the standard unit that is nanomolar, and then the activity, which is 1. So, these are all labeled as 1 because most of them are active, and if we just run this cell, which we call DF, it will actually show you the top 5 and then the bottom 5, okay. So, you can see that in the top 5, most of them are active, while in the bottom 5, you can see that these are the inactive sections; some of them are inactive.

Okay, so one more thing we could have done here is removed the charge, actually. As you can see, some of those compounds are positively charged. We could have removed the charge, and we could have also removed the salts as well. So and that we discussed during the QSAR when if we wanted to prepare a data set, we have to follow all those rules. So, this is a quick tutorial; I skipped some of those, but whenever you are doing so, you have to follow all those rules and prepare your data curation.

You have to prepare your data set very carefully so that you make sure there is no issue with the data you are using. And as we say in predictive modeling, garbage in, garbage out. So, if your input data is not very good, you cannot expect it to do miracles, actually. Okay, so once we have prepared that, and now that we have downloaded the data, we have labeled it into actives and inactives. Okay, so the next step is to convert them, the smiles, into the RDKit-readable mol list.

Okay, so we use the function ChemMolFromSMILES. What it will do is convert these

smiles into an RDKit mol object, so that RDKit can recognize it. And then if you wanted to have a look at it, you can see that in the mol_list, the first compound is this one. If I, for example, change it to the zeroth compound, I can see that the zeroth compound is this one, and if I change it to, for example, For example, the 25th compound looks like this. So, you have full flexibility to visualize whatever you are doing, and that is the beauty of Colab. That is also very important because if you are not able to verify that whatever you are doing is correct, then you might end up with wrong results or wrong calculations.

So, let us just check how many molecules we have in this data set. We have 229 molecules. And if you remember the total number of molecules, when we go back to the ChemBL, we had about 267 molecules. So, out of 267 molecules, we have 229. So, many of them got filtered because many of them had this, you know, IC_{50} value greater than or less than that.

Okay, so now we have converted them into, you know, this RDKit mol object. So then we convert this into the panda data frame, df_ml = pd.DataFrame(mol_list, columns=('object')), so we got the object. Okay. And then, the next step is to generate the fingerprints, okay? So, from rdkit. Chem import Mol, we import the rdkit fingerprint generator, then we import pandas as pd, and we define the parameters like radius 3 and n bits 1024.

So, we will generate a fingerprint of 1024-bit length using the RDKit fingerprint generator, and we will be generating the ECFP6 eccentric connectivity fingerprint. So, we just create a function to convert calculating fingerprints, and then we generate the fingerprints ECFP6 = fpg.GetFingerprint(mol) for mol in df_ml['object']. And we convert the fingerprints into a data frame for inspection, and then we add the fingerprints into a new column for the original data frame dfml ecfp6, that is, if ECFP6.

And then we display the DataFrame with the ECFP6 fingerprints. So, you can see that this is the fingerprint for all these compounds now. So, in the next cell, what we are going to do here is convert them into a numpy array so that they can be recognized by the ML algorithm. Convert them into an X that PD data frame ECFP6 array column FP for I in range and bits, and then we can visualize that as well. So, you can see that we have 182 rows, and these are the columns for the fingerprint.

So, you have FP0 from FP0 to FP1023, and these are, you know, the bits for the fingerprint, actually. Okay. So, now we have defined the X matrix here, and that is what we require, and then we can define a Y matrix as well. So, we define the Y matrix from the DF, DF dot activity. So we can see that we have 182 rows and one column, so we have to make sure that both the x matrix and y matrix are of the same shape and the same size, actually.

Because if there is any ambiguity, for example, if there are 181 compounds in the x matrix

and 182 in the y matrix, then there will be a mess, and this will not work. So to make sure that we are doing that correctly. Now, we have defined a Y matrix where we have outlined the activity of all these compounds. So, where these compounds are, the 1s are active and the 0s are inactive, okay. Okay, so now that we have defined the X matrix and Y matrix, the next step is to split this data, right? So, we split the data using Sklearn's train-test split.

We split it into X_train, X_test, Y_train, and Y_test. And then we use a size of 0.20, which means we are keeping 20 percent of the compounds in the test set and 80 percent of the compounds in the training set, okay. So, we run this cell. So, after running this cell, what we will get is that we will have split the data into training and test sets. And then in the next cell, we will run a model; we will make a model using the Support Vector Classifier.

And then we will use a linear classifier in this case, and then we will train an SV classifier using the X train and Y train data, okay. And then we can see that we can print the evaluation matrix of the classification model that we have just trained. So, you can see here that the accuracy of the model we have trained is 0.91, which is quite good. Actually, it means that it is able to accurately predict 91 percent of the data out of 100 percent. And then you can see we can also, you know, generate the confusion matrix, and then you can see the precision is 0.82, recall is 0.90 for the, you know, inactive compounds, and the precision for active compounds is 0.96 and recall is 0.93.

So, you can see that it is able to predict the active compounds more reliably than the inactive compounds, and you can also see that the F1 score is good. So, overall, this model looks good, and we can just float the confusion matrix as well, as we discussed earlier. So, what you see here is that the predicted labels 0 and 1 mean 0 is, you know, inactive and 1 is active. And then you have the true labels: you know 0 means active and 1 means inactive. For example, what you see here is that 0 predicted and 0 true means these are true negatives, right? So those compounds that were inactive and predicted as inactive, nine compounds were predicted as inactive, and those were actually inactive, and then you have.

This is, this represents your true positive, means you have 25 compounds which are, you know, active and they were also classified as active. So, this is, you know, a true positive, and then you have this, you know, quadrant where you have, for example, the compounds that were predicted as active; however, they were inactive, okay. So, this is a false positive, which means they are predicted as active; however, they were actually inactive. So, this is a false positive; you have the false negative, which means those are actually active, but they are predicted as inactive, right? So, this is how the confusion matrix represents the quality of your model, actually. And then you can always, you know, run the ROC curve, receiver operating characteristic curve.

So, you can see here, and as we discuss, if it is close to, you know, this corner, the top left

corner. So, it is good, and if it is close to the diagonal line, then it is not good; the diagonal means it is, you know, 50/50; it is able to predict the values and the outcomes 50% of the time. So, that is, you know, not very good, actually. Okay, good, so now we have successfully trained a model, a support vector machine classifier, which can predict the tau activity. Activity of any molecule against tau protein in IC_{50} values or means, like we did, because we converted them into active-inactive.

So, this is just a classification model. The model could classify whether a molecule will be tau active or tau inactive. Once we have made this model, the next step is to save it so that we can use it for predicting new compounds. So, we will run this cell.

Then you can see that we have saved this model as svm_model.pkl. Half of the work is done now. So, the next step is to, you know, use this model to predict the activity of new compounds, whether those compounds have some activity against tau or not, by using this model. So, for that, we will, you know, again download a new data set that is just, you know, a sample data set containing just the smile structure. So, we download the data from, you know, one of my GitHub repos.

So we just call it act_ache. These are actually compounds that have some ACHE activity associated with them. But we wanted to predict whether those compounds could have tau activity as well. Okay. So for that, we will download this from my GitHub repo. And then, if we have a look at it, you can see that it has 229 compounds, a molecule ChemBL ID, SMILES, and bioactivity.

So, you do not have to be confused by this bioactivity because this bioactivity is the ACHE activity, which we are not actually interested in, right? So, once we download this data. So, the next step is that we have to repeat the same things, like converting the smiles into the rdkitmol object, and then we have to calculate the fingerprint. So, let us convert the smiles into the rdkitmol object, okay? Once we do that. So, the next step is to calculate the fingerprints. Right, and then once we calculate the fingerprints using the radius of 3 and n bits of 1024, that is another thing that If we are, you know, we have to make sure that we calculate the similar fingerprints as we have done for training the data because if we are using a different fingerprint, then again it will not work, actually.

So, then we calculate the fingerprint and then we put that into this x_ache, and then what we do is convert this data frame into the numpy array. So that it matches, you know, the labeling to the training data fingerprints as well, right? So, we run this cell and then we import the model that we just saved, which is svm_model.pkl, and then we predict the activity, okay. And then we add this predicted activity to the ACT and ACTCHC data

frame. And we add a new column that predicts tau activity, and then if you look at it, you just issue this command: act_ache.head(25).

So, we can see the top 25 molecules which are seen here, and then you can see that this is the predicted tau activity. Now, what our model has done is predict the bioactivity of these compounds against tau using the model that we have recently created. And then you can see that many of the compounds are, you know, predicted to be active compounds, while many of them are predicted to be inactive compounds. So, those that are showing 0 mean those are inactive against tau, and those that are showing 1 are active against tau. So, they might have activity against tau that might be, you know, tau aggregation or tau binding of whatever we have trained the data on, right? So, it depends on which kind of bioactivity data we have trained on.

So, we can predict that bioactivity using this classification model. Okay, and in the end, we can save this, you know, the data to activity_prediction.csv, and we can download that from the folder here as well. You just click on the folder, then refresh it, and you will see that you have the ache_activity_prediction.csv, which contains all this data.

So, the data we have saved in the ache_activity_prediction.csv is okay. And then we can see that we can issue the ls command as well, so we can. So, now what we have seen here is that we have trained an SVM-based classification model to predict the bioactivity of compounds using fingerprints. So in this case, we just use one model, okay, as we did for the regression modeling, so we can always use the lazy predict here as well. which is again a nice Python library for doing modeling and simultaneously making multiple models Like more than 40 models, you can make it simultaneously without any issue, so for that, we just install the lazy predict and then run the next cell.

So in the next cell, what it is going to do is import the modules like LazyPredict.Supervised and LazyClassifier. And then we use sklearn's train_test_split, and then import accuracy_score from the matrix, and then we initialize the LazyClassifier, and then we fit the model using x_test, y_train, and y_test, which we defined earlier. so that we don't have to change, and then it displays, you know, the models' accuracy, balanced accuracy, ROC, and F1 score. So, then you can see here, for example, it has evaluated at least you know how many models—more than 40 models, at least more than 30 models.

And then it has a perceptron that is performing very well, with an accuracy of 0.95. So, you can see multiple models have similar accuracy: perceptron, nearest centroid, random forest classifier, logistic regression, SGD classifier, rich classifier, CV, extra tree classifier, passive aggressive classifier, AdaBoost classifier, bagging classifier. So, you can see many of those models are, you know, working very well in predicting the tau bioactivity, okay. So, for quickly testing multiple models and multiple machine learning algorithms, it is a

very good idea. So, you can create a model using all these algorithms, and then you can pick some of them that are performing best, and you can retrain the model in detail with even hyper-optimization as well by using hyperparameter optimization. So, you can just use grid CV search or random CV search, and then you can actually make the model better.

Okay, so this is how we can build a model, a classification model, and then this model can be deployed as well. So it can be deployed as a web app; for example, on Streamlit, you can deploy it, or on your own server, you can deploy this as a web app. So that any other user can input the structure, draw the structure, and then predict the activity of their compounds. So, that is, you know, the idea of developing all these predictive models. So, I hope you enjoyed this session, and as always, you will have free access to this Colab notebook.

So, you can just try your hand at it and try to build your own models on your own drug targets, and for that, you should start by using the ChemBL data. So, you just have to change, you know, this ChemBL ID, and then everything will be taken care of by this pipeline. Which we have created; otherwise, you can always, you know, carefully curate the data and then use that data in this Colab and make the model, okay? So, with that, thank you.