AI in Drug Discovery and Development

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

Week-12
Lecture-57

Welcome to the course "AI in Drug Discovery and Development." In this hands-on session, we will see how we can make a machine learning-based QSAR model for predicting the solubility of small molecules. So, for that, we need to follow this link. So, we open, we copy this link, and we will open it in a browser once you open that link in the browser. So you will see something like this: you will end up with a Colab notebook with the title "Solubility Prediction Using QSAR Modeling." Okay? For the ease of use, what we will do is make a copy of it; we will just click on File and save a copy in Drive.

And then, once we do that, what it will do is make a copy of it and open it in your browser while keeping a copy of it in your Google Drive. So, for example, we will see something like this: a copy of 12.2 solubility prediction using QSAR modeling. So, in this notebook, we will use QSAR modeling to predict the solubility of small molecules and why solubility is important when we are working with drug discovery.

So, solubility refers to the ability of a substance, such as a drug compound, to dissolve in a solvent, and typically, in drug discovery, we dissolve the compounds in either water or buffers. So, in the context of pharmaceuticals, excess solubility is a critical property that influences a drug's absorption, distribution, and overall effectiveness. Because a drug must be soluble in water or in body fluids to be absorbed, this was also discussed in pharmacokinetics when we were talking about it. So, it will not be absorbed in the body, and if it is not absorbed, then it will not give any effect in the body, actually any therapeutic effect. So, if we quickly look at the importance of solubility in drug discovery and development, it is responsible, you know, for one of the major contributors to bioavailability.

Because the poorly soluble drugs often exhibit low bioavailability, meaning less of the drug reaches the systemic circulation. And then another big challenge is the formulation challenge if there is low solubility of the compounds that can complicate drug formulations and delivery. For example, how we will make them like oral tablets, or you know, the ointments, or you know. So, if those are not soluble, it is very hard to make formulations of them, actually. And then, you know, they affect the ADMET profile as well.

So, solubility is closely tied to absorption, distribution, metabolism, excretion, and toxicity

properties. And then it has a very important role during lead optimization because, during drug design, solubility is optimized alongside potency and selectivity to develop a balanced drug candidate. And this we talked about earlier as well; lead optimization is one of the challenges, you know, steps where we have to optimize those molecules at multiple levels. While retaining the potential of the molecule, we have to, you know, optimize other properties like solubility, toxicity, and efficacy.

Okay. So, what we will do here is see if we can predict the solubility with QSAR. So, what it will do is help us make a decision about whether we should go further with this molecule or not. Taking an example where you have 1000 molecules and those molecules are, you know, having good activity against your target, you want to see which of those molecules you should take further. So, if you can predict their solubilities beforehand, you can make the decision, and that can solve a lot of problems. QSAR models help predict solubility using molecular descriptors, enabling faster screening of compounds in silico even before synthesis or testing.

Like you are optimizing one molecule, and then you have designed maybe 500 molecules from that molecule in order to increase their efficacy and potency. And you wanted to see which of those molecules might have good solubility, so that you could proceed with the synthesis of that molecule. So, those kinds of decisions can also be made. So, you can see that solubility is playing an important role. So, if we can make a QSAR model by using, you know, the molecular descriptors, that will help us a lot.

So, for that, we need to install the RDKIT; we will be using RDKIT here. We can install the RDKIT by running this cell, okay? Once we have installed the RDKIT, you can see that it is installed, and now we will use the RDKIT to build a regression-based QSAR model. To predict the solubility of compounds by using descriptors like LogP, molecular weight, rotatable bonds, and aromatic proportion. So, we will make this model. So, we are taking the dataset from a paper titled "ESOL: Estimating Aqueous Solubility Directory from the Molecular Structure" by Joan S.

Delaney. So, it was published in the Journal of Cheminformatics and Computer Science in 2004. Here is the link to that paper; you can go through it if you want. So, what Mr. Delaney did in that paper is that he introduced the ESOL model, which is a straightforward method for predicting the aqueous solubility of compounds directly from their molecular structure, utilizing linear regression on a data set of 2,874 compounds. The model incorporates nine molecular descriptors.

With calculated LogP, the octanol-water partition coefficient is the most significant, followed by the molecular weight, aromatic atom proportion, and the number of rotatable

bonds. So, ESOL demonstrated consistent performance across three validation sets, predicting solubility within a factor of 5 to 8 of measured values, making it competitive with the established general solubility equation for drug-like molecules. So, we will download the dataset. So, we run this cell, and we will download the dataset again from one of my GitHub repos where I have hosted all these datasets, actually. So, once we download the data set, we will get this Delanay. csv data set, okay.

So, then we will just read it as, you know, as a data frame, okay. We will open the data set and then have a look at the dataset file SOL. So, you can see here, for example, it has the index, and then it has the compound IDs, and it has the measured log solubility, which is in the unit of log solubility in moles per liter. And then it has also given the ESOL predicted solubility, which is the ESOL predicted log solubility in moles per liter, and it has the SMILE structure as well.

So, in the next cell, what we do is take a look at the first structure. For example, this is the structure here, this is the structure indexed at 0, and we can just change it to, you know, see any structure like 1, 1, 2; what will be the structure indexed at 1, 1, 2, 2? So, you can see that this is the structure of the compound indexed at 1, 1, 2, 2. Likewise, we can see any other molecule as well, or we can just plot them using moles to grid, which we saw in the previous session, showing how we can use RDKit moles to grid to visualize the structures. So, once we have done that, the next step is to convert those SMILES to a molecule list and RDKit list files. So, we can just run this cell, and then what it will do is take those SMILES and put them into the, you know, the RDKit molecular object, okay.

And then we can see that the length of the molecule list is 1144. So, we have 1144 molecules here that we will use to make the QSAR model. So, now the next step is, since this data set is already prepared, we did not go through the preparation steps. However, as we discussed earlier, whenever we are making, you know, we are curating the data ourselves. So, it is very, very important that we prepare that data set very carefully.

So that we can get rid of all those issues. And some of those issues which we discussed earlier were, for example, when you have salts associated with the molecules. So, we have to remove them if there is a water molecule, water of crystallization that we have to remove. If there are problems with the bond orders that we have to fix, and then if there are problems with whether a molecule can exist in tautomers, okay. So, based on the pH at which the data are collected, fix that as well.

So, all sorts of those kinds of things we have to do whenever we curate a dataset. So, we have to prepare that dataset carefully to make the QSAR models okay, but since this dataset is already prepared. So, we will just skip those steps and directly jump to the model building

by calculating the descriptors, okay. So, now we will calculate the descriptors. So, as we discussed earlier, descriptors are nothing but the features of molecules that represent them.

So, what we will do here is extract the descriptors from the structure, and here we will calculate the LogP, molecular weight, number of rotatable bonds, and aromatic proportion, which is calculated using RDKit. So, there are many other packages also which we discussed earlier as well which can be used for calculating molecular descriptors like you have the MordRed which can calculate more than 1000 descriptors, then you have the Paddle, which can also calculate more than 1,000 descriptors. So, all these tools can also be used to calculate the descriptors of your choice. Since, in this case, we already know which of those descriptors provide good results or a good QSAR model. So, we will only calculate those, and we will make a model with them.

However, the ideal thing is that you calculate as many descriptors as you can. And then you choose descriptors based on different methods; for example, you can use dimensionality reduction methods such as principal component analysis. where you can identify which descriptors are working for your target or which of them are not working correctly. And then you can use feature selection methods like recursive feature elimination. So, what it does is make a model, and then it will eliminate one feature and see if the model quality is better or worse.

And then, based on that, it will keep dropping and adding features, and in the end, it will give you the list of features that are good for providing a better model for your data. So let us calculate these descriptors. So, when you run this cell, what it will do is create a function to calculate the aromatic proportion, and the aromatic proportion is the amount of aromatic atoms in your molecule, okay. So, for example, if you have a benzene molecule. So, it has all these carbon atoms that it contains, or all the atoms it contains are aromatic in nature, right? So, the aromatic proportion is the number of aromatic atoms divided by the number of heavy atoms.

In the case of benzene, it has 6 carbon atoms and 6 hydrogen atoms. So, all those carbon atoms are aromatic. The aromatic index, which is the aromatic proportion value for benzene, is 1. So, likewise, it calculates the aromatic proportion for each of these molecules. And then, by using the descriptor function in RDKIT, we calculate the molecular weight, number of rotatable bonds, number of hydrogen-bond donors, number of hydrogen-bond acceptors, and aromatic proportion, which we have already calculated using this function, as well as TPSA.

So, we will calculate the TPSA as well, okay. So, here you can see this is, you know, the descriptor function which calculates all these parameters and then adds them into the

column which we have created in the sol file, okay. So, we have this sol file where it will add all these descriptor values, and then in the end, we can just display the updated data frame named df as sol. And you can see here, for example, we have had the index compound ID measured, low solubility, ESOL predicted solubility, SMILES, and molecular weight.

So, these were already here. So, we have added LogP molecular weight, LogP rotatable bonds, number of H donors, number of H acceptors, aromatic proportions, and TPSA. So, you can see the values for some of the first molecules and then the values for the last tail molecules, actually. So, you can see that TPSA, the number of acceptors, and the aromatic proportions—all these values are given. So, now we have calculated the descriptor values, and we have stored those descriptor values in the data frame named "sol ok." So, the next step is what we can do is run this cell; what this cell will do is create a pairwise relationship between the descriptors.

This is another important thing: for example, whenever we are selecting the features for making a QSAR model. So, we see that whether those features are correlating with them or not. So, if those features are correlating with themselves, then we always have to drop those features that are correlated and have high collinearity; this is called high collinearity. We can simply calculate the correlation between each of those descriptors, and then we can drop whichever have high collinearity because that is important. It is important because if we do not do that, we will see a model that is overfitting, right? The correlation between the features will also contribute to the model as well, okay? So, now by running this cell, what we are doing here is making a pairwise scatter upload using Seaborn between all these descriptors.

So, now let us see. So, here you can see that what you see here is like having the molecular weight, for example, and then comparing molecular weight to molecular weight. Here you will just see the distribution plot, and then otherwise you will see a pairwise relation plot. So, here you can see how molecular weight compares with LogP and how molecular weight compares with TPSA. How is molecular weight compared with the number of H-bond donors, the number of acceptors, and the aromatic proportion? So, all these correlations you know, you will see.

Not correlation, sorry. All those you are plotting one by one, actually, like molecular weight on this side and then the low p, low p on this side and molecular weight on this side. So, how does it look? So, you can quickly see some kind of, you know, relation between this data as well, okay, but for properly looking at the relation. So, what you can do is upload this correlation matrix, okay? So, when you run this cell. So, what it will do is we have all these, you know, variables or those descriptors, and then we calculate the

correlation matrix and plot this correlation matrix. When we look at this correlation matrix, what you see here is, for example, the aromatic proportion.

So, the correlation of aromatic proportion with the number of hydrogen bond acceptors, donors, TPSA, LogP, and molecular weight is okay. So, what you will immediately see here is that the TPSA is correlating very strongly with the number of hydrogen bond acceptors. So, TPSA is correlating very strongly with the number of hydrogen bond acceptors that you see here (0.90), and there is also a correlation between TPSA and the number of hydrogen bond donors as well but that is not as strong as the acceptors.

So, on the basis of this correlation matrix or the collinearity, what we can do is drop, you know, the TPSA or number of hydrogen bond acceptors from making this model. So, now what we will do is drop the TPSA, and then we will try to make the model for predicting the solubility. So, for making a model, what we need are two things. One is the x matrix, which will contain the features.

These are also known as independent variables. So, in QSAR or in prediction modeling, whenever we are correlating the features with some activity, we have one variable that is called the independent variable, and those are the features. So, here we define x as equal to sol from the sol file columns: molecular weight, LogP, rotatable bonds, donors, acceptors, and aromatic proportions. We have dropped the TPSA because it was correlating very strongly with the TPSA, H bond acceptors, okay. The Y matrix is the target variable, which is our dependent variable. This means that our dependent variable is the property we want to correlate with the features.

So, in this case, it is our solubility. So, in this case, it is measured as log solubility in moles per liter. So, now that we have defined what x and y are, the x matrix and y matrix, the next step is to scale the features as well. Like you might have seen, the molecular weight is ranging up to 500, while the logP is going only up to 5. The number of rotatable bonds or hydrogen bond donors and acceptors is also up to 5 or 10, and the aromatic proportion ranges only up to 1. So, there is a scaling for all these descriptors that is very different; for example, the molecular weight is almost 500 times larger in scale than the aromatic proportion.

So, this scale needs to be, you know, normalized so that it can come on a similar scale and its contribution does not affect the model quality, okay. So, for that we do the scaling. So, we use a standardized scaler; what it does is calculate the mean, median, etc., and then, on the basis of that, it scales down the descriptor values.

So, we will just run this cell. So, it will scale that down, and then another step we have to perform is to divide the data into a training set and a test set, okay. So, whenever we are

using, we are developing a model. We have a training data set, and then we have a test data set. So, we train the model on the training dataset and evaluate the quality of the model by using the test set.

So, that is what we do. In this case, we are defining the x_train, x_test, y_train, and y_test, which will divide the data of these X matrix and Y matrix into training and test sets. Okay, and then we are using the, you know, X scaled method. Okay, and then we are using a test size of 0.2, which means 20 percent of the molecules will be in the test set, and 80 percent of them will be in the training set.

And that can be changed as well; like, you can have 0.3, and then you can compare how well you know your model works when you are using different splits, actually. So, let us run this one. So, when we run this, it has scaled the data and then it has specified the x matrix and y matrix and split them into the training and test sets as well. So, in this case, we will use the machine learning model. So, in this case, we are using the random forest regressor, and there are, you know, hundreds of them; actually, you can choose whichever is more suitable for you.

So we will use the random forest regressor here. When we run this cell, what this cell is going to do is take the training and test set data and then fit them using the random forest regressor. And then report the evaluation statistics, which are the mean absolute error and the R-squared score. Okay, so it will train the model. model.

fit(x_train, y_train); make the prediction y_pred = model.predict(x_test) and then evaluate the model, where it will report the MAE, mean absolute error, and the R-squared value. So, you can see that the mean absolute error is 0.47, which is very good because, you know, the smaller it is, the better the model is, okay? And then the R-squared score is 0.89. So, that is also quite good because the closer it is to 1, the better model we actually have.

So, now we have talked about this mean square error and R-squared score, and we see that our model is performing well. So, that that can be now saved actually. So, we can just save this model so that we can use it for prediction because the purpose of the modeling is not just to make a model, but to use this model to predict the solubility of new compounds. So, we just import the joblib module, and we save it as joblib.dump(model, 'random_forest_model.pkl'). And then joblib dumps the scaler as well as "scaler.pkl" because of the way we scaled the data in this case. So, the same scaling needs to be done on the, you know, the test data as well as the external test set data, okay? Then we can just quickly list whatever files we have.

So, we have got the random_forest_model.pkl, sample_data, scaler.pkl, and the

updated_sol1.csv. Okay, now that we have made the model, the thing is how we also saw from the statistics that the model is quite good. So, now the next thing is, can we see the feature importance, and that adds to the explainability, actually. So, we are talking about explainable, you know, AI or explainable methods.

So, this is one of the methods by which we can add explainability to models. So, now when we run this cell, what it will do is note the importance of the features from the random forest. So, it will create a bar plot of feature importance. And here you can see that this is a feature on the x-axis; you have feature importance, on the y-axis you have all those descriptors of the features, and you can see what is more important for predicting solubility.

The molecular weight, LogP, is ok. These are the major contributors to the solubility, and then you have, you know, aromatic proportion and the number of H bond acceptors. The number of rotatable bonds is also important, but the major role is played by the LogP. So, if the LogP is, you know, higher, then the molecule is not soluble. If the LogP is low, then the molecule is soluble, and it is universally accepted that LogP is a major contributor to predicting solubility because LogP indicates the hydrophobicity or lipophilicity. If a molecule is lipophilic, then LogP will be high, and that is how we use it.

So, after that, what we can do is plot the predicted versus the actual for the best models you can see here. So, this is the true solubility, and this is the predicted solubility, and you see a nice, you know, correlation between them, okay. So, you can see that it has predicted the solubility of the models quite well. And then we can do the cross-validation as well.

So, that is another important feature or thing to consider in QSAR modeling. So, whenever we are evaluating our models, we have used a training set for training the model and a test set to evaluate the model. But somehow we have not used, you know, an external test set, actually. So, to do that, there are actually two ways. One is either you split your data into training, test, and validation. So, in that case, it will have, for example, 60 percent of your data in the training set, 20 percent of the data in the test set, and 20 percent of the data in the validation set.

So, it will use 60 percent for training, 20 percent will be used for testing, and those 20 percent will be used for evaluating the model because those 20 percent have not been used during the model building or testing. So, in that case, we use that method; the problem is that we are already reducing the size of the training data set, actually. So, we will lose some information and lose some molecules, actually. Another way is to do the k-fold cross-validation, okay? So, in k-fold cross-validation, what we do is make a model by using an 80-20 training and test set. And then we run it again by just randomly selecting the training and test sets again.

So, in this case, what we are doing is making the model multiple times, and each time the split between the training and test sets is different. So, it means that every time the molecules that were in the training set and test set were actually different. So, then we evaluate how well this actually worked. When we run this cell, what it will do is import the sklearn model selection and cross-validation score k-fold. And then on the x, we have all these features; on the y, we have the target variables.

We initialize the random first regressor, make the model, and then set up the cross-validation; we are using 5-fold validation. So, CV k-fold n split is 5, the random state is 42, and shuffle is true, which means that every time we shuffle and perform the cross-validation, we get the R-squared scores for the regression. So, here we get the model XY CV and scoring R-squared. So, here you can see that when we run this, the cross-validation R-squared score is 0.89, 0.88, 0.88, 0.87, 0.87; it did 45 runs, and then the mean R-squared is 0.88 and the SD of R-squared is 0.0088. So, you can see that it has confidently shown a very high confidence in the model that every time it is done, we are getting a good score. Good R-squared score, which is close to 1.

Okay, so this is one way to evaluate the model. Another way is to note the learning curve as well. So, this is how fast the model is able to learn, you know, the data and learn the trend in the data. So, when we run this cell, what you will have here is that it will import the module from sklearn, and then it will plot the learning curve for both the training and test sets. So, here you can see that on the x-axis we have the training size, and then on the y-axis we have the score.

So, you can see that the learning curve for the training set is very high. This means that it has learned very quickly even with a small amount of data. So, it has learned, you know, very quickly; it has got a very high score close to one. However, for the cross-validation score.

So, it was a little bit slow learning and did not reach, you know, more than 0.85. So, sometimes there is a large difference between the learning curve for the training and cross-validation scores. So, it can represent that there is some overfitting in your model, actually, because your model is very good at predicting the solubility of the training set. While it is not very good at predicting, you know, the solubility of the cross-validation set or the test set. So, in that case, we have to, you know, look at whether there is collinearity or whether there are, you know, some features that are affecting the performance of the model. or whether there is this training test split can affect the prediction this you know these statistics.

So, we have to go back and check all these things. Okay, so that was, you know, how we build a model and then how we validate it and how we look at whether the model is good or not. The next thing is, as I said, once we have made a model, the question now is how we can use that model to predict the properties of new molecules. And this is the real application, as it is called, for deploying the model or the deployment of the model. So, you might have seen this DruMap as we talked about it, actually. So, DruMap is nothing, but they have made these kinds of models for different properties, and then they have made a web server and deployed those models there.

So that other people can predict properties of their molecules. So, now let us see how we can predict the solubility of external unknown molecules. For that, what we will do is download this dataset. So, we will go to you when we run this cell. Then, we will download this SOL1 file, and if you look at the SOL1 file.

So, you can see that it has just the ambit key score activity in this file. So, there is nothing about the solubility. So, these molecules have no solubility data associated with them. So, what we have here is just the SMILE structures, their SMILES, their activity score, and the ambit key. So, these are irrelevant; actually, we are active, and the score is not relevant here.

Therefore, we will not take care of them. We just need these SMILES. So, now we have these SMILES, and we want to calculate, you know, the solubility of these molecules from these SMILES. So, we have opened it, and then the next step is. So, the idea is that once we have made that model, we now know which of those features are important. So, now if we can calculate the features for these new molecules and just put those features into that model, okay.

So, we can calculate the solubility, okay, that is, it is that simple, actually. When we run this cell, what this will do is define a function to calculate the aromatic proportion again. Another important thing is that whenever we are using this model, we have to make sure that we can only use the features that were implemented in that model. So, we have to calculate those features for the external molecules, then put those features into the model, and only then will we get the prediction. So, here we define the function to calculate the aromatic proportion; we initialize the empty list for each descriptor, then calculate the descriptors and put them into the empty list we have created.

And then we save it as a data frame to a new CSV file, sol1.csv. And then we can add this into the sol1, and then you can see that the sol1 contains what we have calculated, so these are there. So, we have the molecular weight, LogP, rotatable bonds, donors, acceptors, aromatic proportion, as well as the TPSA, okay. So, we have calculated all these properties,

like we did for the training dataset: molecular weight, LogP, rotatable bonds, donors, acceptors, aromatic proportion, and TPSA. So, once we calculate, the next step is to load the model and scaler.

So, the model that we saved, random_forest_model.pkl, and the scaler.pkl, which we also saved to make sure that we do the similar scaling; otherwise, it will not, you know, give us good results. So, we run this cell, and then after scaling, we define x_sol1 and x_sol_scaled = scaler.transform(x_sol1). And then we predict the solubility using modal.predict(x_sol1_scaled), and then we add the predicted solubility to the sol1 file. So, when we run this cell, what you can see here is that you will get the predicted solubility. We have added the predicted solubility here. So, now we can see that we have the predictive solubility values for the molecules in our data frame file.

So, where we had these smiles. So, this is, you know, the application or deployment of these models. And now we know that our model is working well because it has an R-squared value of about 0.8. So, the error was also less. Now we can confidently say that the predicted solubility is close to their observed values. However, we need to confirm it because we do not know at least for these molecules, but as per the model statistics.

So, these are very confident. Okay, this is how you can predict the solubility of molecules. And this is not only about the solubility. So, like this workflow you can use for any sort of property prediction. So, whether it is solubility, toxicity, or potency against any target. So, all sorts of activities, all sorts of, you know, properties like ADMET properties or toxicities can be predicted by using this method, actually. So, in the end, I just wanted to show you how you can even make multiple machine learning models and choose the best among them by using this library called Lazypredict.

Okay. So, in this case, we have just made a model with a random forest regressor, which is just one machine learning technique or algorithm that we use to create a model. So, first we have to install LazyPredict. When we run this cell, we will install LazyPredict. So, LazyPredict is a library that can build more than 30 models altogether in a very short time.

Since we are going to make a regression model. So, we will import lazy predict, and from lazy predict.supervise, we will import LazyRegressor. And then the regressor is a lazy regressor, and the models are X_train, X_test, Y_train, Y_test that we have already defined, which you know we did by using that, and then you can just display the different models. So, when we run this, what it will do is it will run.

So, you can see 67, 80. So, it is running 42 different models and then comparing the differences between them. So, now you see that it has built these 42 different models and

the best model is the random forest regressor, which has an adjusted R-squared value of 0.

90, an R-squared value of 0.90, and an RMSD of 0.67, and it took just 0.46 seconds to make this model. And then you have the extra tree regressor, bagging regressor, hist gradient boosting regressor, gradient boost regressor, LGBM regressor, SVR, XGBoost regressor, and no SVR. So, you can see that most of these models are giving very good statistics. So, this means they are very good at making this model and predicting the property, specifically predicting the solubility from these descriptors. However, there are some bad models as well; for example, you can see that some of them have poor statistics as well. However, what you can do now is the best use of this lazy predict is to make some models, and after that, you can pick those that are giving you the best results.

And then, in detail, you can individually make models and evaluate those models by plotting the learning curve or by plotting, you know, how to say, the feature importance map. So, all that you do, and then you can compare whichever is, you know, whichever is giving you the best results, okay. So, yeah. This is how it is done, and then I just quickly let us go to this, yeah. Yeah, so here what we can do now is we can just say that, okay, we want to make a test split of, quite frankly, thirty percent of the molecules.

We wanted to have it in the test split, so by doing that, what we can do is select this, click on runtime, and then run. cell and below. So, because we just started from this cell and then built the models, let us see if we get some difference in that. So, you can see here that the mean absolute error has increased a little bit, and the R-squared score has decreased a little bit when we are using 30 percent of them in the test set. Okay, and then if you look at the importance, the feature importance is, as it is, there is not much change in the feature importance.

And this one also looks good: the predicted and the true solubility for the best model. And then the learning curve, you can see, is also going to be the same; actually, it is not much different. So, all those sorts of changes you can make in this model, and then you can create a machine learning-based model for predicting any sort of property. I hope you will use this pipeline. This Colab is freely accessible, so you will have access to it forever. So, you can use it to make your machine learning model for your research problems and your datasets, and I hope you find it useful. And with that, thank you.