## Machine Learning Techniques

* 1. **Machine Learning Algorithms Employed**

Several machine learning algorithms are applied to the dataset to optimize the result. Some of the techniques give results close to the output's actual values. On our dataset, 6 different algorithms namely, Support Vector Regressor (SVR), Decision Tree Regressor (DTR), Random Forest Regressor (RFR), Nearest Neighbor Regressor (NNR), Ridge, and Lasso are applied to obtain the desired output. Our dataset contains both numerical and categorical parameters and the continuous output parameter, so the One Hot Encoding method is applied to convert all categorical values into numeric values and then machine learning models are applied to the dataset. The following section discusses the various machine learning algorithms that are applied to the dataset.

## Decision Tree Regressor (DTR)

A most common, widely used, and efficient machine learning algorithm to solve the regressor tasks. It is imported from sklearn.tree library. DTR works by partitioning the data into subsequent parts until the leaf node is achieved or there is no data left for partitioning. DTR partitions the data based on some condition. The top node in the DTR is called as root node and nodes with children as left and right are called internal nodes. The nodes which have zero children are called leaf nodes. Data points that meet the condition go to the leaf child while others go to the right child. In this way, a whole decision tree is made. To make predictions for the output parameter, data points are taken and compared with the conditions present at all the nodes and based on that condition, the data point goes towards the left or right child. When the data point reaches the leaf node, average values of all the data points present at the leaf node are found as the predicted value of that particular data point. To obtain better results, the condition should be based on a variance reduction factor. That condition, which lowers the impurity more should be taken and variance reduction has to be found out at all levels of the tree. DTR also has different parameters to optimize the results such as criterion, splitter, max\_depth, min\_samples\_leaf, min\_samples\_split, etc. It is an intuitive algorithm suitable for various regression problems. **Fig. 1** shows the actual insights of DTR.

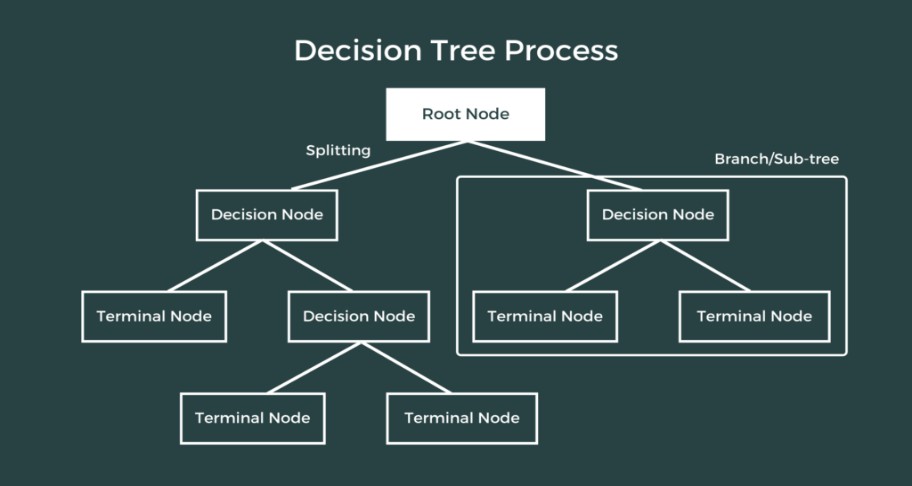


Fig. 1 DTR

## Random Forest Regressor (RFR)

Random Forest Regressor is a type of bagging technique. It is imported from sklearn.ensemble library. The base learner in a random forest technique is a decision tree. It makes all the decision trees parallelly. In the random forest technique, the dataset is divided into samples with some rows and some features taken and given to the decision tree. For each iteration, samples with replacements are given to the decision trees. After that, decision trees are trained on the given sample. For regressor problems, the mean of all the output predicted from different decision trees is taken as the final output as clearly shown in **Fig. 2**. Decision trees have two problems that are low bias and high variance, but random forest regressor solves the problem of decision trees by taking all decision trees parallelly and controlling the depth of each decision tree. Different parameters are available in the sklearn library to optimize the results of RFR such as n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf, max\_features, etc. Random Forest Regressor is a robust model that gives accurate and suitable predictions for a wide range of regression problems.

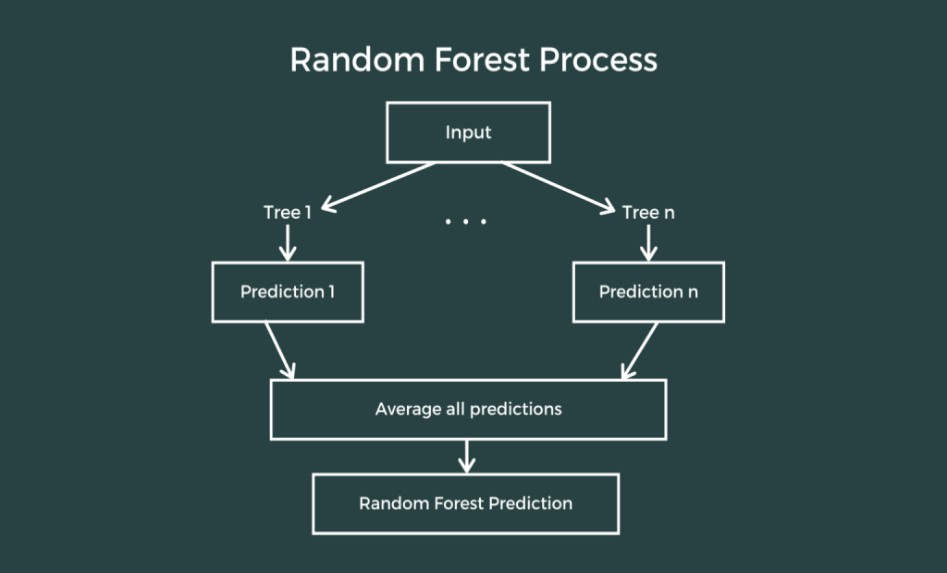


Fig.2 RFR

## Support Vector Regressor (SVR)

Support Vector Regressor is an efficient machine-learning technique for regression problems. It is imported from sklearn.svm library. Best-fit line and 2 marginal planes are created in SVR. The marginal planes are equidistant from the best-fit line. The distance between the marginal plane and the best-fit line is epsilon as shown in **Fig. 3**. To obtain better results, epsilon should be minimized. In SVR, the distance between the actual value and predicted value should be less than or equal to epsilon. If the distance between them is more than epsilon, then the predicted data point is not considered to be accurately predicted. SVR can handle both linear and non-linear relationships between the features. It can handle complex datasets by transforming the lower-dimension feature space into a higher-dimension feature. SVR also supports different types of kernels such as linear, polynomial, and rbf kernel. To optimize the results of prediction, the sklearn library provides many hyperparameters for SVR such as kernel, gamma, C, tol, epsilon, etc. C and epsilon are the free parameters.

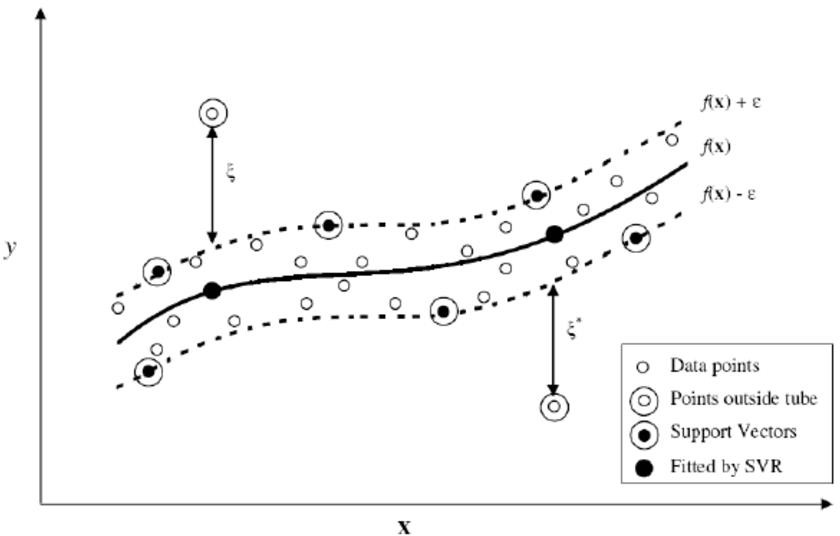


Fig. 3 SVR

## Nearest Neighbor Regressor (NNR)

It is a supervised machine-learning algorithm used for regression tasks. It is imported from sklearn.neighbors library. The value of the number of nearest neighbors can be any value ranging from 0 to infinity. Generally, it is derived using hyperparameters. Grid search is used and the value of several nearest neighbors is selected among many values which gives the best results in the prediction. After finalizing the value of the number of nearest neighbors, the distance between the

test data point and the nearest neighbors is found as depicted in **Fig. 4**. Distance is calculated with the help of either the Euclidean formula or the Manhattan distance, as shown in **equations 1 and 2.** Then, the average of all the distances is calculated to find the predicted value. NNR is mostly used with smaller datasets as computational complexity increases with an increase in the data points. Also, NNR is sensitive to outliers. Outliers have a huge impact on the performance of the NNR model as the distance between the test data point and the outlier point is huge as compared to other distances which will impact the final results. It is also sensitive to missing points. Sklearn library also provides many hyperparameters to optimize the results such as n\_neighbors, weights, algorithm, leaf\_size, etc. SVR is a simple yet powerful machine-learning algorithm.

**Manhattan Distance** -**|x1 – x2| + |y1 – y2| --(1)**

**Euclidean Distance - √[ (x22 – x11)2 + (y22 – y11)2]** --(**2**)

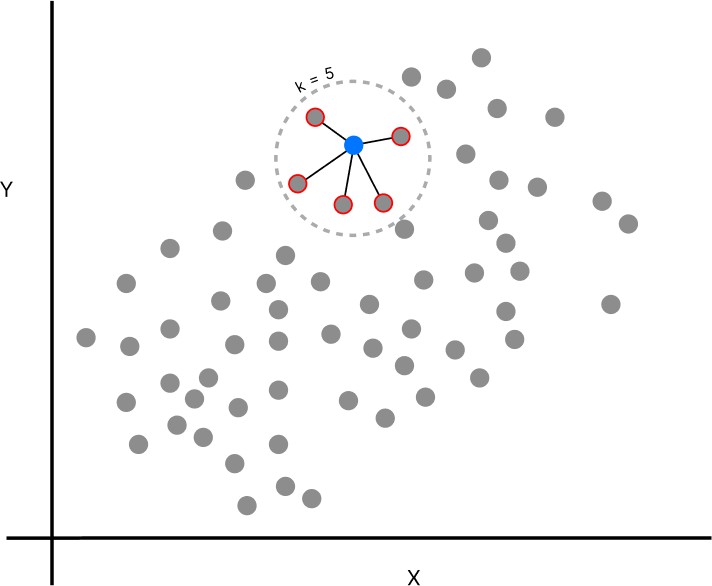


Fig. 4 NNR

## Ridge Regressor

Ridge regression is used when there is a high variance in the training data. The main idea behind ridge regression is to add a small amount of bias in the training data which lowers the variance up to a great extent. It is imported from sklearn.linear\_model library. Predictions made with ridge regression lines are less sensitive to weights. In ridge regression, a penalty term is added in terms of lambda. Lambda is also known as the regularization parameter. It controls the trade-off between fitting the training data and keeping the model coefficients small. Lambda is directly proportional to the ridge regression penalty. The lambda value is chosen wisely, it is chosen with the help of either grid search or cross-validation method. Ridge regression works on both continuous and discrete

values. It is used because datasets having smaller sizes can lead to poor least-square estimates, so a penalty term is included in the equation. Penalty is sensitive to feature scale, so it is important to scale the features before applying ridge regression. Several hyperparameters are used to optimize the results of ridge regression such as tol, fit\_intercept, alpha, max\_iter, etc. Ridge can also be applied to logistic regression. It also prevents overfitting and provides a robust solution. **Fig. 5** shows the actual representation of Ridge Regression.

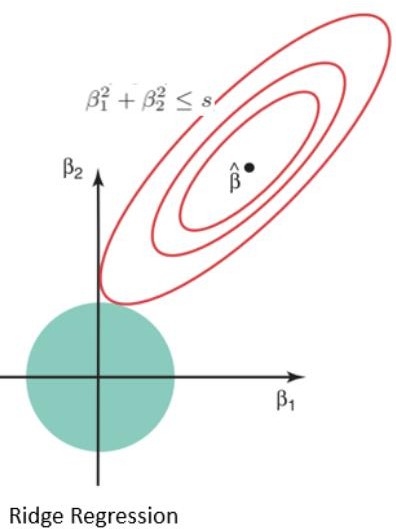


Fig. 5 Ridge Regression

## Lasso Regressor

Lasso regression is also called L1 regression. It is similar to ridge regression but there are some differences. It is imported from sklearn.linear\_model library. It also has a regularization parameter lambda, which is similar to ridge but it adds lambda \* absolute(slope) to the equation as a penalty term as shown in **Fig.6** . Lambda ranges from zero to infinity. The Lasso regression line has more bias but less variance compared to the simple model. Lasso regression can be applied to datasets that are small in size or datasets having complicated data. It can shrink the parameters, but not equally every time. The main difference between ridge and lasso regression is that when the value of lambda increases, ridge will shrink the value of slope asymptotically close to zero and lasso will shrink lambda to zero. Lasso is a little bit better than Ridge in reducing the variance of a dataset having useless variables. Ridge is better when most of the variables are useful. Sklearn library provides hyperparameters also to optimize the results of lasso regression such as alpha, tol, fit\_intercept, max\_iter, etc. It is a powerful tool for regression tasks.

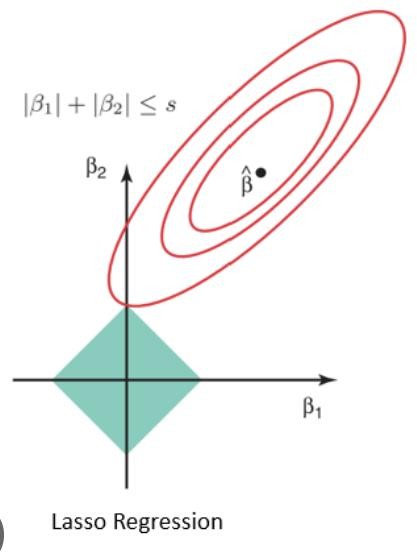


Fig. 6 Lasso Regression

## Results and Discussion

This section discusses the performances of machine learning models such as DTR, SVR, NNR, RFR, Ridge, and Lasso. Different errors are calculated such as R^2, Root Mean Square Error (RMSE), Mean Square Error (MSE), Variance Accounted For (VAF), and Index of Scatter (IOS). Error metric and Taylor diagram are made for comparison of different models. Best hyperparameters are found to optimize the prediction results.

## Performance of Machine Learning Technique

82 data points are present in the dataset. The dataset is divided into training and testing data. 33% of total data points are taken as testing data while others are taken as training data. The dataset consists of 3 input features namely, Techniques, Composition, and Passes, and 1 output feature which is R1. Input features are a mixture of both categorical as well as numeric values. Techniques is a categorical feature having buried, sandwiched, and grooved as categories, Composition is also a categorical feature having composition A, B, and C as categories while Passes have numeric values. Output feature, R1 has continuous values. To convert categorical features into numerical features, One hot encoding is done before applying machine learning techniques. K-fold cross-validation is applied to the models. To optimize the results of the prediction model, hyperparameter tuning is done with the help of a grid search. Best hyperparameters are found and the model is trained on those parameters.

## Accuracy of Decision Tree Regressor (DTR)

The decisionTreeRegressor() function is imported from the sklearn library. The dataset consists of both categorical as well as numerical values so one hot encoding is done to convert all categorical values into numerical values. One hot encoder is present in sklearn.preprocessing library. After applying One hot encoding to all the categorical features, the model is trained on the new dataset

having more features due to One hot encoding. The new dataset has features namely, Techniques\_sandwiched, Techniques\_grooved, Techniques\_burried, Composition\_A, Composition\_B, Composition\_B, and Passes. 5-fold cross-validation is applied to the DTR model, and hyperparameter tuning is done to predict better results. The model is trained with hyperparameters in each fold and predicted results are appended to the list. Hyperparameter tuning is done with the help of grid search. 5-6 parameters are taken in the grid with different values and the best parameters values are chosen among all values to predict the output. **Fig. 7** represents the actual and predicted data points in the graph. It is clearly shown in **Fig. 7**, that most of the predicted data points have values closer to the actual values. Around 12-13 data points have values slightly different from the actual values. As shown in Fig., around 60 data points have their predicted values near to the true values. K fold and grid search are imported from sklearn.model\_selection library. To evaluate the performance of the DTR model, 5 types of errors are calculated namely R^2, Root Mean Square Error (RMSE), Mean Square Error (MSE), Variance Accounted For (VAF), and Index of Scatter (IOS).

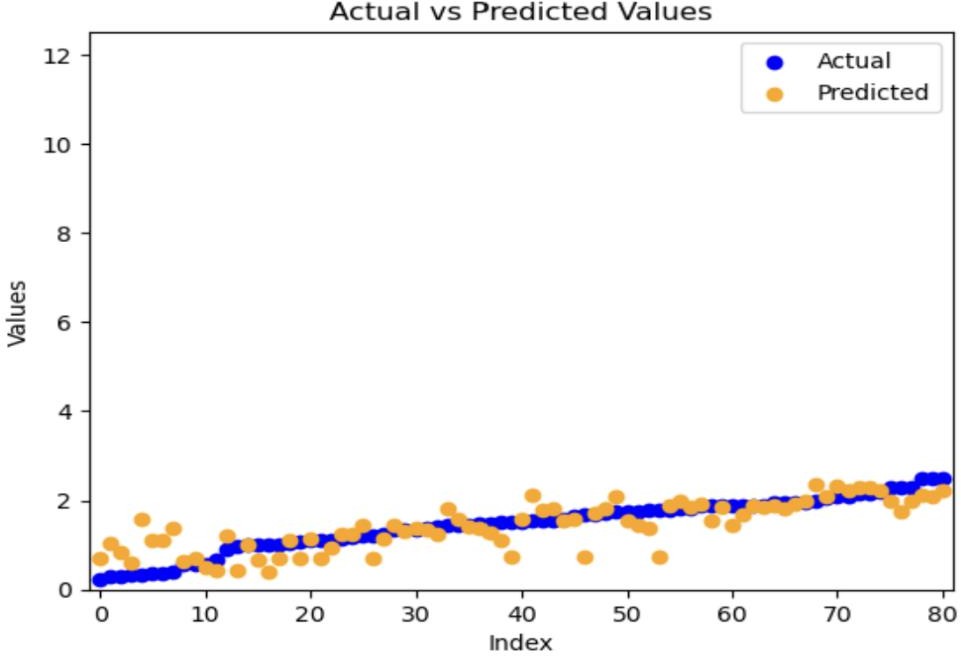


Fig. 7 DTR

## Accuracy of Random Forest Regressor (RFR)

The RandomForestRegressor() function is imported from sklearn.ensemble library. Due to different data values present in the dataset, such as categorical and numerical, one hot encoding is applied to the whole dataset. After applying one hot encoding, features in the dataset increase and some other features are added like Techniques\_sandwiched, Techniques\_grooved, Techniques\_burried, Composition\_A, Composition\_B, Composition\_B, and Passes. RFR is applied to training data. 5-fold

cross-validation is applied to the model. K-fold cross-validation and grid search are imported from sklearn.model\_selection library. In each iteration, the model is trained with hyperparameters. Hyperparameter tuning is done with the help of grid search. A grid of 5-6 hyperparameters is made and the best parameter values are used among all parameter values to train the model. **Fig. 8** represents the actual and predicted value of data points with the RFR model. As shown in Fig, most of the predicted data points are pretty close to their true values. Around 60-70 data points have values near their true values. Some predicted points are slightly far from the actual data points. There are no outliers present as clearly shown in the graph. To evaluate the performance of the RFR model, 5 types of errors are calculated such as R^2, Root Mean Square Error (RMSE), Mean Square Error (MSE), Variance Accounted For (VAF), and Index of Scatter (IOS).

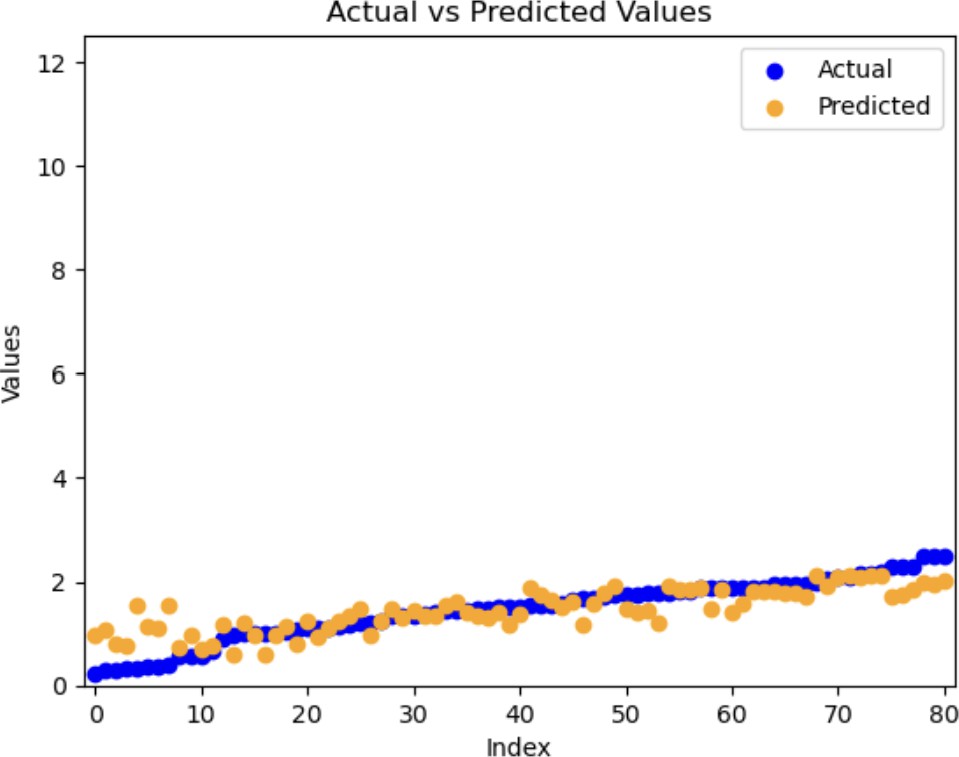


Fig. 8 RFR

## Accuracy of Nearest Neighbor Regressor (NNR)

kNeighborsRegressor() function is imported from sklearn.ensemble library. The value of k is chosen from among the list of k values. The model is trained with every value of k and the value that gives the best result is used for the final training of the model. One hot encoding is applied to the whole dataset before applying the NNR model to it. It converts all categorical values to the numerical values. One hot encoder is imported from sklearn.preprocessing library. 5-fold cross-validation is applied to the dataset. In each fold, the model is trained with hyperparameters. Hyperparameter tuning is done with the help of grid search. K-fold and grid search are imported from

sklearn.model\_selection library. A grid is made with parameters namely, n\_neighbors, p, leaf\_size, metric, and algorithm. Different values of parameters are present in the grid. With the help of the grid search function, only those values are selected which gives the best results. **Fig. 9** depicts the actual and predicted values of all the data points. Around 10-20 predicted data points have values slightly different from the true values. There are no outliers present in the graph. All the other data points are near their actual values. The graph is plotted with the help of the matplotlib library. NNR model performance is judged based on errors. The minimum is the error, the better is the result of the prediction. R^2, RMSE, MSE, VAF, and IOS are calculated from the predicted and actual lists.

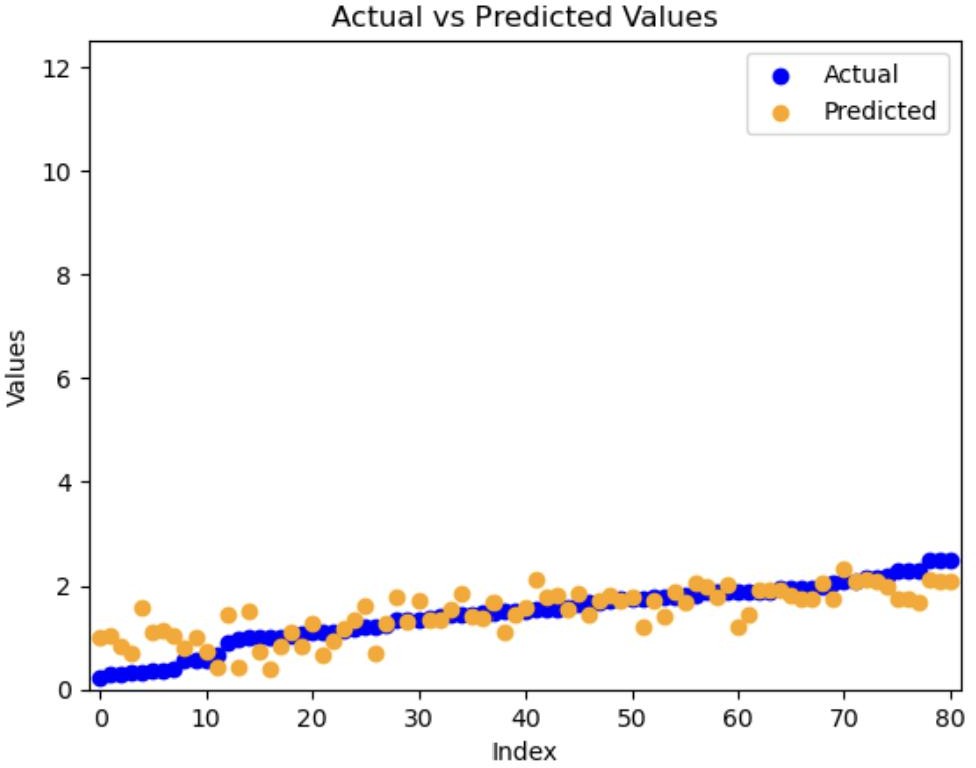


Fig. 9 NNR

## Accuracy of Support Vector Regression (SVR)

The SVR() function is imported from sklearn.svm library. Sklearn also provides several hyperparameters to optimize model output. Hyperparameters namely, C, kernel, degree, gamma, epsilon, shrinking, and tol are applied. Grid search is used to search for the best value among the different values of the hyperparameter present in a grid. The best values are then used for training and testing the model. One hot encoding is done before training the model as the dataset contains different types of data, numerical as well as categorical. One hot encoder is imported from sklearn.preprocessing library. As clearly shown in **Fig. 10**, around 55-60 predicted data points are near the actual values while others have slightly different predicted values than the actual values.

Model performance is evaluated with the help of different types of errors such as R^2, RMSE, MSE, VAF, and IOS. Error metric and Taylor diagram are models for each model.

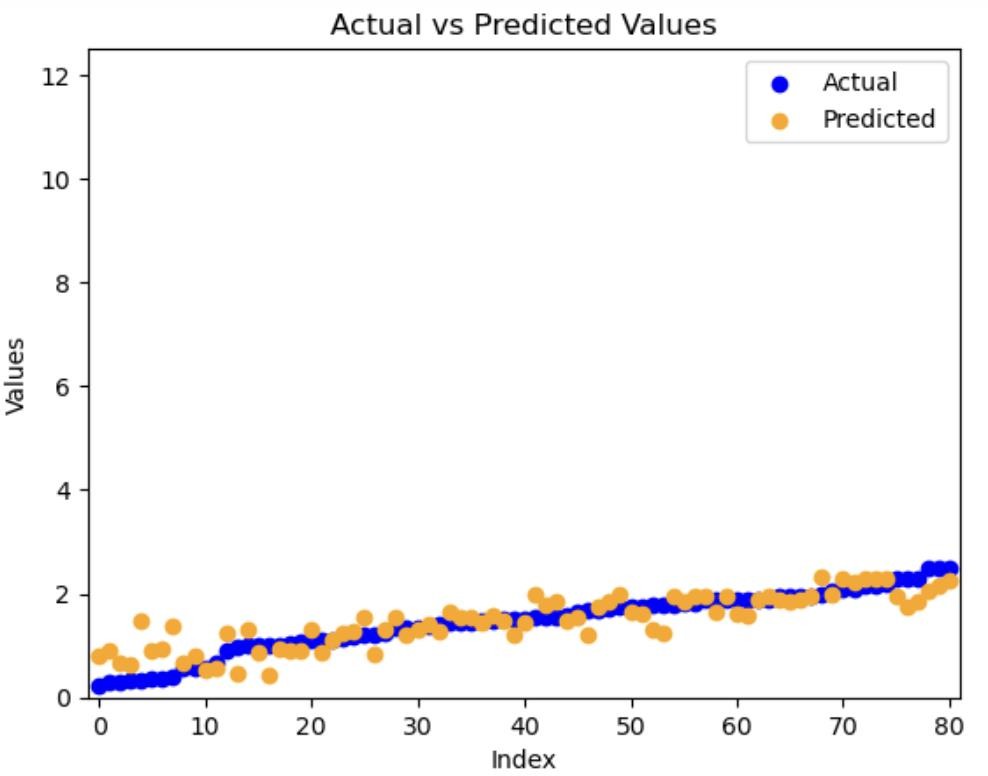


Fig. 10 SVR

## Accuracy of Ridge Regression

The Ridge() function is imported from sklearn.linear\_model library. The dataset consists of 3 parameters, some of them are categorical features, output features have continuous values. To convert all categorical values into numerical values, one hot encoding is used. One hot encoder is imported from sklearn.preprocessing library. After one hot encoding, 5-fold cross-validation is applied to the model. In each fold, training data is tuned with hyperparameters. Hyperparameter tuning is done with the help of grid search. Predicted data points are stored in a list after each fold. **Fig. 11** depicts the actual and predicted data points. As is clearly shown in Fig, most of the predicted values are closer to the true values. Around 65-70 data points have values pretty close to the actual ones while other data points are slightly far away from their true values. Most of the predicted data points are overlapping the actual data points. It uses a tolerance of 0.001 in training and testing the model. Implementation of k-fold cross-validation and grid search is done with the help of

sklearn.model\_selection library. Model evaluation is based on errors. Error metric is made with different errors such as R^2, MSE, RMSE, VAF, and IOS.

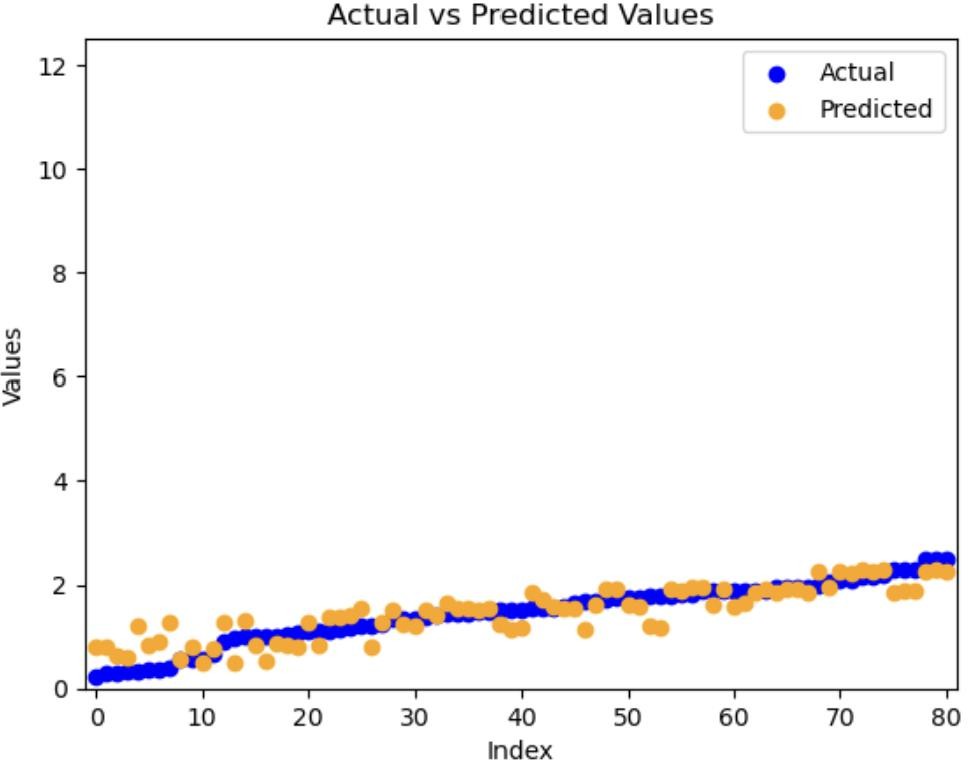


Fig. 11 Ridge

## Accuracy of Lasso Regressor

The Lasso() function is imported from sklearn.linear\_model library. One hot encoding is done before applying the lasso model as the dataset contains both numerical and categorical data. One hot encoder is imported from sklearn.preprocessing library. Number of columns increases after one hot encoding. Now, the dataset contains columns namely, Techniques\_burried, Techniques\_sandwiched, Techniques\_grooved, Composition\_A, Composition\_B, Composition\_C, and passes. After applying one hot encoder, 5-fold cross-validation is applied to the whole dataset. In each fold, hyperparameter tuning is done with the help of a grid search. Grid search and k-fold cross-validation are imported from sklearn.model\_selection library. As clearly shown in **Fig. 12**, some points are far away from the true values but some data points are overlapping with the actual values. Around 50- 55 predicted data points are pretty close to their actual values. As depicted in Fig., there are no outliers present, so the output of the lasso does not vary largely from the actual values as the lasso is sensitive to the outliers. Lasso has fewer predicted data points overlapping with actual data points as compared to Ridge regression. Different hyperparameters are used to optimize the output of lasso such as alpha, fit\_intercept, max\_iter, positive, tol, and selection. The best value among different values of hyperparameter is selected to train the model using grid search. Lasso performance is

judged based on errors such as R^2, RMSE, MSE, VAF, and IOS. Error metric and Taylor diagram are made for all the machine learning models.

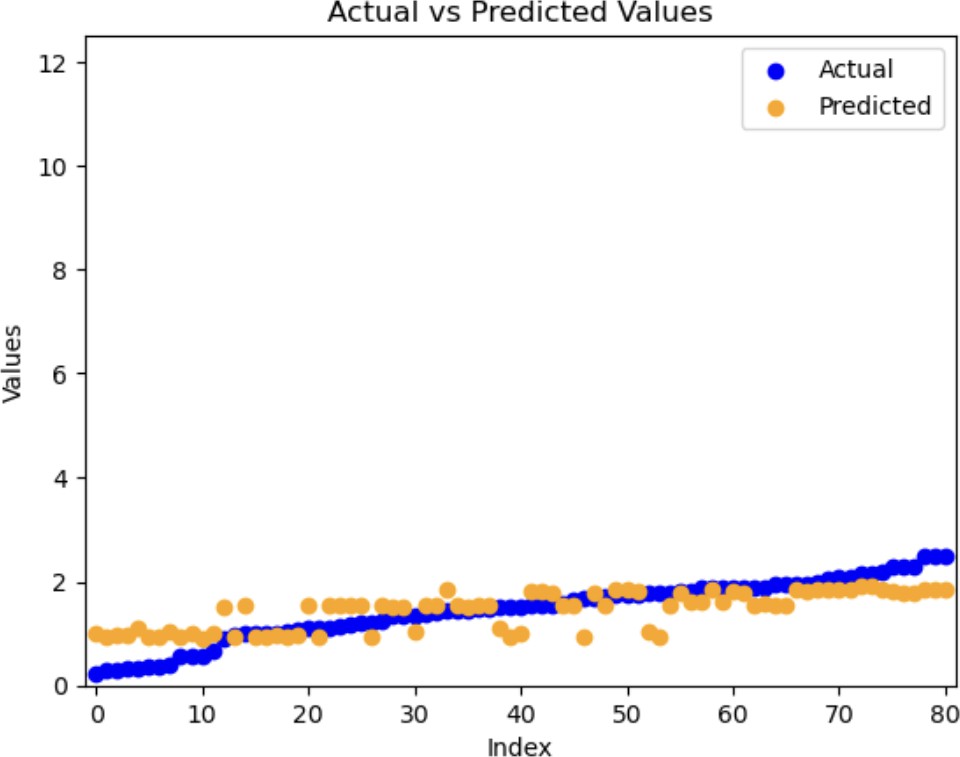


Fig. 12 Lasso

## Comparison of Machine Learning Technique

The whole dataset is divided into testing and training data in the ratio 1:3. After testing and training data, different errors are calculated to evaluate the model's performance. Errors such as Root Mean Square Error (RMSE), R^2, Mean Square Error (MSE), Variance Accounted For (VAF), and Index of Scatter (IOS) are calculated. Error metric is made based on the errors. In addition to that, the best parameter values are found that are used during hyperparameter tuning of the model. The best parameter values among different values are selected from the grid using grid search. Taylor diagram and sensitivity analysis is done to evaluate the model results. **Fig. 13** depicts the RMSE values for all the models through a line graph. As clearly shown in Fig., the Ridge model has the lowest RMSE value and gives the best results.

𝐑𝐌𝐒𝐄 = √∑𝒏

(𝒚𝒑𝒊− 𝒚𝒎𝒊)𝟐 … … … … … … (𝟏)

𝒏

𝒊=𝟏

𝐌𝐒𝐄 = ∑𝒏

(𝒚𝒑𝒊− 𝒚𝒎𝒊)𝟐

𝒏

𝒊=𝟏

… … … … … … (𝟐)

𝐕𝐀𝐅 = 𝟏 − (𝑽𝒂𝒓(𝒚′ − 𝒚)/𝑽𝒂𝒓(𝒚) (𝟑)

𝐈𝐎𝐒 = 𝐬𝐪𝐫𝐭 (∑𝒏 (𝒚𝒑𝒊− 𝒚𝒎𝒊)𝟐 ) /𝒎𝒆𝒂𝒏(𝒚𝒑

… … … … … … (𝟒)

𝒊=𝟏 ⬚

𝒊)

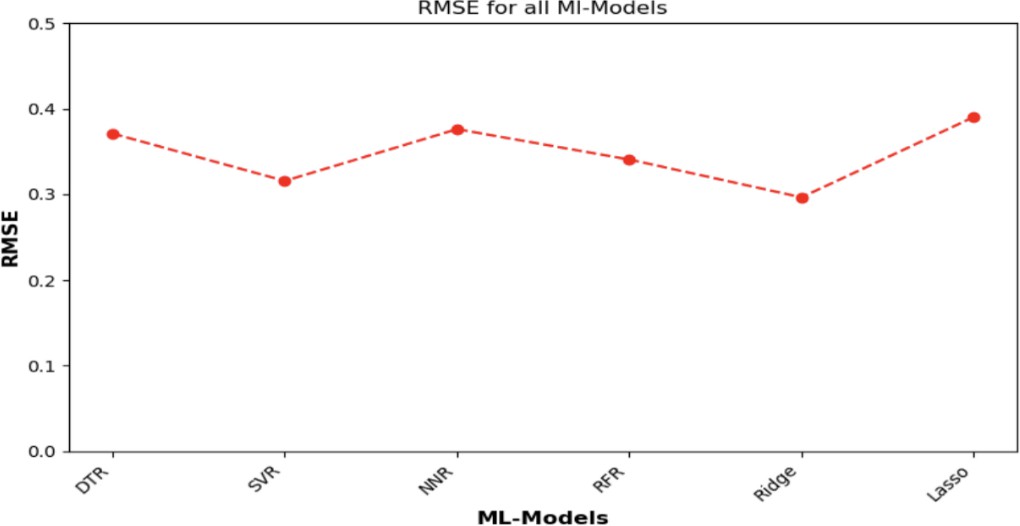


Fig. 13

**Eq. 1,2,3, and 4** show the mathematical equation of errors that are calculated to evaluate the performance of the model. RMSE is calculated by eq1, mse is calculated by eq2, vaf and ios are calculated by eq3 and eq4 respectively.

## Best Hyper Parameter Used

Several values of each parameter are given in the grid. The model is trained on each parameter value one by one, and out of all the given values, the best parameter values are chosen with the help of a grid search. Grid search is imported from sklearn.model\_selection library. Model function and parameter grid are taken as parameters for the grid search. The model is trained n number of times where n is the size of the grid search.

# DTR

best

|  |  |
| --- | --- |
| **HYPER PARAMETERS** | **APPROXIMATE VALUE** |
| CRITERION | Squared\_error |
| MAX\_DEPTH | None |
| MAX\_FEATURES | None |
| MIN\_SAMPLES\_LEAF | 4 |
| MIN\_SAMPLE SPLIT | 5 |
| SPLITTER |  |

Table 1

6 hyperparameters are taken for the DTR model, and their approximate best values are shown in

**Table 1.**

# NNR

|  |  |
| --- | --- |
| **HYPER PARAMETERS** | **APPROXIMATE VALUE** |
| ALGORITHM | Brute |
| LEAF\_SIZE | 10 |

|  |  |
| --- | --- |
| METRIC | Manhattan |
| P | 1 |
| WEIGHTS | uniform |
| N\_NEIGHBORS | 5 |

Table 2

6 hyperparameters are taken as input to the grid search for the NNR model and their best values are found using grid search as shown in **Table 2**.

# RFR

|  |  |
| --- | --- |
| **HYPER PARAMETERS** | **APPROXIMATE VALUE** |
| LEARNING RATE | 0.01 |
| MIN\_SAMPLES\_LEAF | 1 |
| MAX\_FEATURES | Sqrt |
| MIN\_SAMPLES\_SPLIT | 5 |
| N\_ESTIMATORS | 50 |

Table 3

Hyperparameter values along with their names are shown in **Table 3** for the RFR model. 5 hyperparameters are used in optimizing model performance.

# RIDGE

|  |  |
| --- | --- |
| **HYPER PARAMETERS** | **APPROXIMATE VALUE** |
| ALPHA | 1.0 |
| FIT\_INTERCEPT | True |
| MAX\_ITER | 500 |
| SOLVER | saga |
| TOL | 0.001 |

Table 4

**Table 4** shows the values and names of the hyperparameters used while training the Ridge regression model.

# LASSO

|  |  |
| --- | --- |
| **HYPER PARAMETERS** | **APPROXIMATE VALUE** |
| ALPHA | 0.1 |
| FIT\_INTERCEPT | True |
| POSITIVE | False |
| SELECTION | random |
| TOL | 0.001 |

|  |  |
| --- | --- |
| MAX\_ITER | 200 |

Table 5

5 hyperparameters are used in optimizing the model performance of LASSO regression as depicted in

**Table 5**.

# SVR

|  |  |
| --- | --- |
| **HYPER PARAMETERS** | **APPROXIMATE VALUE** |
| C | 1 |
| DEGREE | 2 |
| EPSILON | 0.1 |
| GAMMA | Scale |
| KERNEL | Linear |
| TOL | 0.001 |
| SHRINKING | True |

Table 6

Hyperparameters along with their values are shown in **Table 6** for the SVR model. These are the best values among different values of hyperparameters.

## Sensitivity Analysis

Sensitivity analysis is used to analyze the effect of each input parameter on the output of the model. Each input parameter is removed once and output is calculated with the remaining input parameters. Sensitivity is calculated as the difference between the value of error with all input parameters taken and the value of error when one input parameter is removed. The ranking is done based on the sensitivity value. The parameter having a high sensitivity value means it is having a greater effect on the output results. The R^2 value is calculated after removing each parameter and values are compared with the R^2 value calculated when all input parameters are taken. **Table 7** shows the sensitivity analysis for all the input parameters. As is clearly shown in Table 7, Passes parameters show the highest sensitivity value, which means it has the greatest impact on the result of the model while composition\_B has the least impact on the final results with the lowest sensitivity value among all input parameters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameters Removed | Parameters Taken | R^2 | Sensitivity | Rank |
| --- | Passes, Techniques\_Buired, Techniques\_Groove, Techniques\_Sandwich, Composition\_A, Composition\_B, Composition\_C | 0.745251 | --- | - |
| Passes | Techniques\_Buired, Techniques\_Groove, Techniques\_Sandwich, Composition\_A, Composition\_B, Composition\_C | 0.069756 | 0.675495 | 1 |
| Techniques\_Buired | Passes, Techniques\_Groove, Techniques\_Sandwich, Composition\_A, Composition\_B, Composition\_C | 0.745185 | 0.000066 | 7 |
| Techniques\_Groove | Passes, Techniques\_Buired, Techniques\_Sandwich, Composition\_A, Composition\_B, Composition\_C | 0.745017 | 0.000233 | 5 |
| Techniques\_Sandwich | Passes, Techniques\_Buired, Techniques\_Groove, Composition\_B, Composition\_C | 0.745018 | 0.000233 | 6 |
| Composition\_A | Passes, Techniques\_Buired, Techniques\_Groove, Techniques\_Sandwich, Composition\_A, Composition\_B, Composition\_C | 0.744404 | 0.000847 | 2 |
| Composition\_B | Passes, Techniques\_Buired, Techniques\_Groove, Techniques\_Sandwich, Composition\_A, Composition\_C | 0.745648 | -0.000397 | 8 |
| Composition\_C | Passes, Techniques\_Buired, Techniques\_Groove, Techniques\_Sandwich, Composition\_A, Composition\_B | 0.744702 | 0.000549 | 4 |

Table 7 Sensitivity Analysis

## Taylor Diagram

Taylor diagram is a tool to represent how patterns, achieved with different models that are used in training the dataset match with the referenced dataset. It compares different models against the reference model. Taylor diagram is made with the help of the correlation of different outputs with the reference output and standard deviation calculated after training the dataset on different models. The standard deviation and correlation coefficient of the referenced dataset are also taken as the input and are represented by a dot on the graph. Each model is shown as a dot on the Taylor diagram and the distance between the origin and each model dot is the standard deviation and the angle from the x-axis is the correlation coefficient of the models. **Fig. 14** shows the Taylor diagram involving all the models such as DTR, RFR, NNR, SVR, Ridge, and Lasso regression. Included a gap of

0.2 between each point model as the value of standard deviation and correlation coefficients for each model are very close.

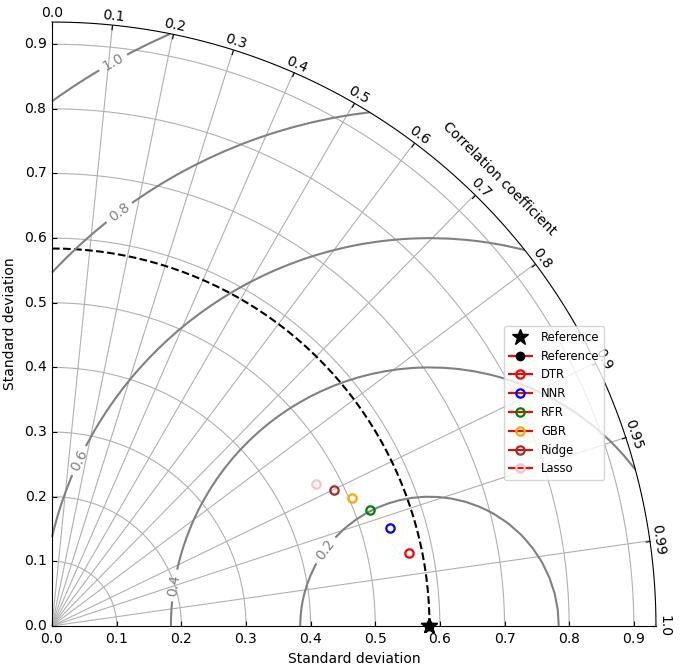


Fig. 14 Taylor Diagram