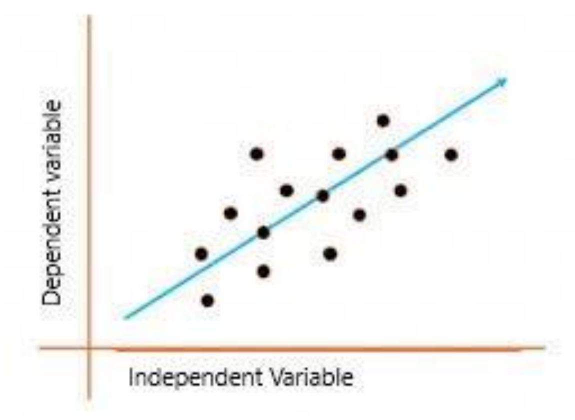
Regression Algorithm

Linear Regression:

Linear regression is a statistical method used to model the linear relationship between a dependent variable and one or more independent variables. In other words, it is used to predict the value of a dependent variable based on the values of one or more independent variables. The relationship between the variables is represented by a straight line in a two-dimensional space.

The goal of linear regression is to find the equation of the line that best fits the data, where "best" means that the sum of the squared differences between the actual and predicted values (i.e., the residuals) is minimized. This is typically done by calculating the slope and intercept of the line using a method called least squares regression.

Linear regression can be used for both simple and multiple regression, where simple regression involves only one independent variable, and multiple regression involves more than one independent variable. Linear regression is widely used in various fields, including economics, finance, social sciences, engineering, and natural sciences, to analyse the relationship between variables and make predictions. **(go to Note Copy)**



Advantages:

- **Simple and easy to understand:** Linear regression is a simple and easy-to-understand statistical method that can be implemented using basic mathematical concepts.
- **Widely applicable:** Linear regression is widely applicable in many fields, including finance, economics, social sciences, engineering, and natural sciences.
- **Provides useful insights:** Linear regression can provide useful insights into the relationship between variables, including the strength and direction of the relationship, as well as the extent to which one variable affects the other.

- **Can be used for prediction:** Linear regression can be used to make predictions about future values of the dependent variable based on the values of the independent variables.

Disadvantages:

- **Assumes a linear relationship:** Linear regression assumes that the relationship between the dependent variable and independent variables is linear, which may not always be the case in real-world scenarios.

- **Sensitive to outliers:** Linear regression can be sensitive to outliers, which can significantly affect the slope and intercept of the regression line.

- **Does not account for nonlinear relationships:** Linear regression cannot capture nonlinear relationships between the dependent and independent variables, which may result in poor predictive performance.

- **Can be affected by multicollinearity:** When multiple independent variables are included in the model, linear regression can be affected by multicollinearity, which can result in unstable estimates of the regression coefficients.

- **Overfitted:** Sometimes it generated overfitted model.

Overall, linear regression is a useful and widely applicable statistical method that provides valuable insights into the relationship between variables. However, it is important to be aware of its limitations and potential biases, and to carefully evaluate the assumptions and requirements of the model before using it for analysis or prediction.

Ridge Regression (L2):

Ridge regression is a regularization technique used in linear regression to prevent overfitting of the model to the training data. Overfitting occurs when the model is too complex and fits the training data too closely, resulting in poor performance on new data.

In ridge regression, a penalty term is added to the least squares objective function of linear regression, which shrinks the regression coefficients towards zero. This penalty term is controlled by a hyperparameter, called the regularization parameter or lambda, which determines the degree of shrinkage applied to the coefficients. **[penalty term: alpha x (slope)^2]**

The effect of ridge regression is to reduce the variance of the regression coefficients, making the model less sensitive to the noise in the data and reducing the risk of overfitting. However, ridge regression also increases the bias of the model slightly, which may result in reduced predictive performance on new data.

Ridge regression is commonly used in situations where the number of predictors is large relative to the number of observations, or when the predictors are highly correlated with each other. It can also be used to stabilize the regression coefficients in situations where there is high multicollinearity among the predictors. **(go to note)**

Assumption of Regression:

Ridge regression is a type of linear regression and therefore, it shares many of the same assumptions as linear regression. In addition, it also has some unique assumptions due to its regularization technique. The key assumptions of ridge regression are:

- **Linearity:** Ridge regression assumes that there is a linear relationship between the independent variables and the dependent variable.
- **Independence:** The observations should be independent of each other. This means that there should be no relationship or correlation between the residuals of the model.
- **Homoscedasticity:** The variance of the residuals should be constant across all values of the independent variables.
- **Normality:** The residuals of the model should be normally distributed.
- **No multicollinearity:** Ridge regression assumes that there is no perfect multicollinearity between the independent variables. Perfect multicollinearity occurs when one independent variable is a perfect linear combination of other independent variables, which can cause problems with the estimation of regression coefficients.
- **The regularization parameter is chosen appropriately:** Ridge regression assumes that the regularization parameter is chosen appropriately, which requires knowledge of the data and the specific problem being solved.

These assumptions are important to ensure that the ridge regression model provides accurate and reliable results. Violations of these assumptions can lead to biased and unreliable results, which can impact the interpretation and utility of the model.

Advantages:

- **Reduces overfitting:** Ridge regression helps to reduce overfitting by shrinking the regression coefficients towards zero, which can improve the generalization performance of the model on new data.
- **Handles multicollinearity:** Ridge regression is effective at handling multicollinearity, which occurs when the predictors are highly correlated with each other. By reducing the impact of correlated predictors, ridge regression can improve the stability and interpretability of the model.
- **Can handle high-dimensional data:** Ridge regression can be used with high-dimensional data, where the number of predictors is larger than the number of observations.
- **Provides a regularization parameter:** Ridge regression provides a regularization parameter that can be tuned to balance the trade-off between the bias and variance of the model.

Disadvantages:

- **May reduce predictive performance:** Ridge regression can increase the bias of the model slightly, which may reduce the predictive performance on new data.

- **Requires tuning of the regularization parameter:** The effectiveness of ridge regression depends on the choice of the regularization parameter, which must be tuned using cross-validation or other techniques.

- **Does not perform feature selection:** Ridge regression does not perform feature selection, which means that all predictors are retained in the model. This can result in reduced interpretability of the model and increased computational complexity.

- **Assumes a linear relationship**: Ridge regression assumes a linear relationship between the predictors and the response variable, which may not be appropriate in some situations where the relationship is nonlinear.

Lasso Regression (L1):

Lasso regression, also known as L1 regularization, is a linear regression technique that adds a penalty term to the cost function that is proportional to the absolute value of the coefficients. This penalty term shrinks the coefficients towards zero, which can help to reduce overfitting and improve the generalization performance of the model.

The L1 penalty has the additional benefit of performing variable selection by setting some coefficients exactly to zero. This means that Lasso regression can automatically select the most important features and remove the irrelevant ones, which can lead to a simpler and more interpretable model.

The level of regularization in Lasso regression is controlled by a hyperparameter called alpha. The higher the value of alpha, the stronger the regularization and the more coefficients will be set to zero. **(go to note)**

Advantages:

- Performs automatic feature selection by setting some coefficients to zero, which can lead to a simpler and more interpretable model.

- Helps to reduce overfitting and improve the generalization performance of the model.

- Provides a simple and easy-to-understand approach to regularization.

Disadvantages:

- Lasso regression can have high variance when there are many features with a small effect size.

- Lasso regression is sensitive to the scaling of the input features, so it is important to scale the data before fitting the model.

- The choice of hyperparameter alpha or lambda can be challenging, and it requires tuning and cross-validation to find the optimal value.

ElasticNet Regression:

Elastic Net regression is a regression algorithm that combines the penalties of Lasso (L1 regularization) and Ridge (L2 regularization) regression. It is used to overcome the limitations of both Lasso and Ridge regression and to provide a more robust and accurate model.

So, we get the value of R-square, when we apply ElasticNet Regression, which is very less than both ridge and lasso regression. The reason is we don't have large data set, ElasticNet performs well when we have huge number of data set.

The Elastic Net regression algorithm involves the following steps:

- Standardize the feature matrix to have zero mean and unit variance.
- Initialize the regression coefficients to zero or some small value.
- Calculate the cost function.
- Update the coefficients using the gradient descent algorithm to minimize the cost function.
- Repeat steps 3 and 4 until convergence or a maximum number of iterations is reached.

Advantages:

- Overcomes the limitations of Lasso and Ridge regression.
- Handles high-dimensional datasets with a large number of features.
- Provides a more interpretable model by performing feature selection and regularization.
- Reduces the risk of overfitting and improves the generalization performance of the model.

Disadvantages:

- Requires tuning of the hyperparameters alpha and lambda.
- Computationally more expensive than Lasso and Ridge regression.

Differentiate between Ridge, Lasso and ElasticNet Regression:

- **Penalty terms:** Ridge regression uses an L2 regularization penalty term, Lasso regression uses an L1 regularization penalty term, while Elastic Net regression uses a combination of both L1 and L2 regularization penalty terms.
- **Variable selection:** Ridge regression does not perform variable selection and includes all variables in the model, while Lasso regression performs variable selection by setting some coefficients to zero, and Elastic Net regression performs variable selection and shrinks the coefficients towards zero.
- **Computational complexity:** Ridge regression has a closed-form solution and is computationally efficient, while Lasso regression and Elastic Net regression require iterative optimization algorithms and are computationally more expensive.
- **Tuning parameters**: Ridge regression has a single tuning parameter, Lasso regression has a single tuning parameter, and Elastic Net regression has two tuning parameters.

- **Robustness:** Lasso regression is sensitive to the outliers, while Ridge regression and Elastic Net regression are less sensitive to the outliers.

- **Application:** Ridge regression is used when all the features are important, and Lasso regression is used when there are redundant features. Elastic Net regression is used when both L1 and L2 regularization is required to obtain a balance between bias and variance.

In summary, Ridge regression is used to prevent overfitting when all the features are important, Lasso regression is used when there are redundant features and feature selection is desired, while Elastic Net regression is used when both L1 and L2 regularization are needed to obtain a balance between bias and variance.

Polynomial Regression:

Polynomial regression is a type of regression analysis in which the relationship between the independent variable (x) and the dependent variable (y) is modelled as an nth degree polynomial function. This allows for the modelling of non-linear relationships between the variables, which cannot be captured by linear regression.

The general form of a polynomial regression equation with one independent variable is:

$$y = b0 + b1.x + b2.x^2 + \dots + bn*x^n$$

where y is the dependent variable, x is the independent variable, and b0, b1, b2, ..., bn are the coefficients of the polynomial equation.

The key advantage of polynomial regression is that it allows for a flexible modelling of the relationship between the variables, capturing non-linear patterns that might be missed by linear regression. Polynomial regression can also be used for interpolation (predicting values within the range of the observed data) and extrapolation (predicting values outside the range of the observed data).

However, polynomial regression can be prone to overfitting, especially when the degree of the polynomial is high. Overfitting occurs when the model fits the noise in the data, rather than the underlying patterns, leading to poor generalization performance on new data. Therefore, it is important to choose the appropriate degree of the polynomial based on the complexity of the data and the model's generalization performance.

How to choose the appropriate degree of polynomial:

Choosing the appropriate degree of the polynomial is an important step in polynomial regression to avoid overfitting or underfitting the data. Here are some common methods to choose the degree of the polynomial:

- **Visual inspection:** One way to choose the degree of the polynomial is to plot the data and visually inspect the relationship between the variables. If the data appears to have a linear relationship, a linear model might be sufficient. If there is a clear curvature, a polynomial model might be appropriate.

- **Cross-validation:** Another way to choose the degree of the polynomial is to use cross-validation. Cross-validation involves partitioning the data into training and validation sets, fitting the model on the training set, and evaluating its performance on the validation set. This process is repeated for different degrees of the polynomial, and the degree that yields the best performance on the validation set is chosen.

- **Information criterion:** Another method to choose the degree of the polynomial is to use information criterion, such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC). These criteria penalize models with more parameters, thus favouring simpler models that generalize better to new data.

It is important to note that the choice of the degree of the polynomial depends on the complexity of the data and the problem at hand, and there is no one-size-fits-all solution. It is often a trade-off between model complexity and generalization performance.

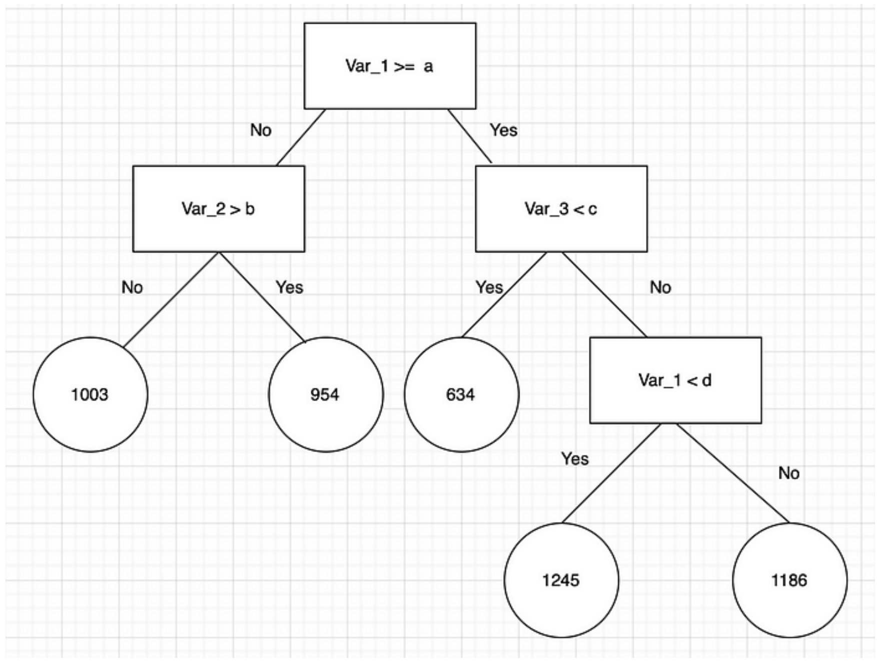
Decision Tree Regressor:

A Decision Tree Regressor is a supervised learning algorithm used for regression tasks. It creates a decision tree model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

The decision tree starts with a root node, which represents the entire dataset, and recursively splits the dataset into smaller subsets based on the feature that best splits the data. The splits are based on the feature that results in the maximum reduction in variance or mean squared error (MSE).

Once the tree is built, the predicted value for a new input is obtained by traversing the tree from the root to a leaf node that corresponds to the input's feature values. The predicted value is the average of the target values in the training set that belong to that leaf node.

Decision Tree Regressor has some advantages, such as being easy to interpret and visualize, and can handle both continuous and categorical data. However, it can overfit the data and perform poorly on unseen data if the tree is too deep or if the data is noisy or has missing values.



Step by Step process to create Decision Tree Regression algorithm:

- **Collect and prepare the data:** Collect the relevant dataset and prepare it for processing. This includes removing any missing values, encoding categorical variables, and scaling numeric variables.
- **Split the dataset:** Split the dataset into training and testing sets. The training set is used to train the decision tree model, while the testing set is used to evaluate its performance.
- **Train the decision tree model:** Use the training set to train the decision tree model. The goal is to create a tree that maximizes the reduction in variance at each split, while also avoiding overfitting.
- **Visualize the decision tree:** Visualize the decision tree to get a better understanding of how it works. This can be done using tools such as Graphviz or scikit-learn's built-in tree visualization.
- **Evaluate the model:** Use the testing set to evaluate the performance of the decision tree model. Common performance metrics for regression problems include mean squared error (MSE), mean absolute error (MAE), and R-squared.
- **Tune the hyperparameters:** Experiment with different hyperparameters of the decision tree model, such as the maximum depth of the tree or the minimum number of samples required to split a node, to see if you can improve its performance.
- **Repeat steps 3-6 with different algorithms:** Try different algorithms, such as random forests or gradient boosting, to see if they can improve the performance of the decision tree model.
- **Choose the best model:** Choose the best model based on its performance on the testing set and deploy it to make predictions on new data.

Note that this is a general process for creating a decision tree regression algorithm, and some of the details may vary depending on the specific implementation and the nature of the dataset.

Advantages:

- **Easy to understand and interpret:** Decision Tree Regression is a simple and intuitive model that is easy to explain and understand. It provides a graphical representation of the decision-making process, which can be easily interpreted by humans.
- **Non-parametric:** Decision Tree Regression does not make any assumptions about the underlying distribution of the data or the relationship between the input and output variables. It is a non-parametric model that can handle both linear and non-linear relationships.
- **Handles missing values:** Decision Tree Regression can handle missing values in the data by using surrogate splits.
- **Can handle irrelevant features:** Decision Tree Regression can handle irrelevant features by ignoring them and selecting only the relevant features for the model.
- **Robust to outliers:** Decision Tree Regression is robust to outliers because it does not depend on the mean and variance of the data.

Disadvantages:

- **Overfitting:** Decision Tree Regression is prone to overfitting the data, especially when the tree is deep and complex. Overfitting occurs when the model captures noise and idiosyncrasies in the data instead of the underlying pattern.

- **Instability:** Decision Tree Regression can be unstable, as small variations in the data can result in a completely different tree.

- **Bias:** Decision Tree Regression can be biased towards features that have a large number of levels or values.

- **Limited to regression problems:** Decision Tree Regression is limited to regression problems and cannot be used for classification problems.

- **Not suitable for continuous outcomes:** Decision Tree Regression may not be suitable for continuous outcomes as it can result in choppy predictions.

- **May not capture complex relationships:** Decision Tree Regression may not be able to capture complex relationships between the input and output variables, especially when the relationship is non-linear or involves interactions between features.

Compare between Linear Regression and Decision Tree Regression:

- **Linearity:** Linear regression assumes that the relationship between the independent variables and the dependent variable is linear, while decision tree regression can model nonlinear relationships.

- **Interpretability:** Decision tree regression provides an easily interpretable model, as the tree structure can be easily visualized and understood. Linear regression, on the other hand, is not as easily interpretable as the relationship between the independent variables and the dependent variable may be complex.

- **Overfitting:** Linear regression is prone to overfitting when the number of features is large, while decision tree regression is less prone to overfitting and can handle high-dimensional datasets.

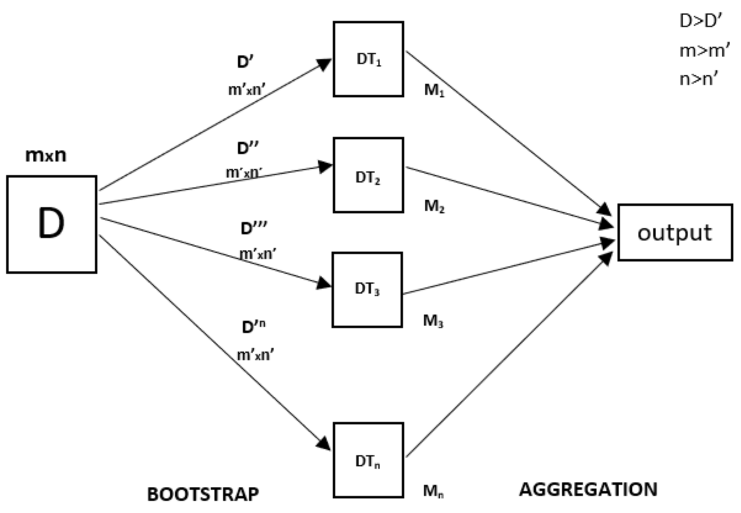
- **Robustness to outliers:** Decision tree regression is robust to outliers and can handle them well, while linear regression may be affected by outliers and can give biased results.

- **Performance:** In terms of performance, linear regression is often faster and more efficient than decision tree regression, especially for datasets with many features.

Overall, the choice between linear regression and decision tree regression depends on the specific problem and the characteristics of the dataset. Linear regression is better suited for linear relationships, while decision tree regression is more flexible and can handle nonlinear relationships. Decision tree regression may be preferred for its interpretability and robustness to outliers, while linear regression may be preferred for its efficiency and speed.

Random Forest Regressor:

Random Forest Regression is a supervised machine learning algorithm used for regression tasks. It is an ensemble learning method that combines multiple decision trees to make more accurate predictions. In Random Forest Regression, each tree in the forest is built independently by randomly selecting data points and features from the training set. The final prediction is then obtained by averaging the predictions of all the trees in the forest.



We will use the sklearn module for training our random forest regression model, specifically the RandomForestRegressor function. The RandomForestRegressor documentation shows many different parameters we can select for our model. Some of the important parameters are highlighted below:

- **n_estimators** — the number of decision trees you will be running in the model.
- **criterion** — this variable allows you to select the criterion (loss function) used to determine model outcomes. We can select from loss functions such as mean squared error (MSE) and mean absolute error (MAE). The default value is MSE.
- **max_depth** — this sets the maximum possible depth of each tree.
- **max_features** — the maximum number of features the model will consider when determining a split.
- **bootstrap** — the default value for this is True, meaning the model follows bootstrapping principles (defined earlier).
- **max_samples** — This parameter assumes bootstrapping is set to True, if not, this parameter doesn't apply. In the case of True, this value sets the largest size of each sample for each tree.
- Other important parameters are **min_samples_split, min_samples_leaf, n_jobs**, and others that can be read in the sklearn's RandomForestRegressor documentation here.

Advantages:

- Random Forest Regression is a powerful algorithm that can handle large datasets with high dimensionality.
- It is less prone to overfitting than other machine learning algorithms.
- It can handle both continuous and categorical data.
- It can provide feature importance scores to help identify the most important features in the dataset.
- It is relatively easy to tune and can provide good results with minimal parameter tuning.

Disadvantages:

- It can be slow to train on large datasets.
- It can be difficult to interpret the results of a Random Forest model.
- It may not perform as well as other algorithms on small datasets with low dimensionality.
- It can be sensitive to noise and outliers in the dataset.

Compare between Linear Regression and Random Forest Regression algorithm:

- **Model Interpretability:** Linear Regression produces a linear equation that directly shows the relationship between the dependent variable and independent variables, which can be easily interpreted. On the other hand, Random Forest Regression produces an ensemble of decision trees, which are more difficult to interpret.
- **Handling of Outliers:** Linear Regression can be sensitive to outliers in the data, as they can disproportionately influence the regression line. Random Forest Regression is less sensitive to outliers, as the ensemble of decision trees reduces their impact.
- **Handling of Nonlinear Relationships:** Linear Regression assumes a linear relationship between the dependent and independent variables, which may not hold in some cases. Random Forest Regression, on the other hand, can capture nonlinear relationships between variables by combining multiple decision trees.
- **Overfitting:** Linear Regression is prone to overfitting when there are too many independent variables or when the model is too complex. Random Forest Regression is less prone to overfitting, as the ensemble of decision trees helps to reduce variance and generalize the model better.
- **Data Size:** Linear Regression can handle large datasets efficiently, while Random Forest Regression can be slower and more computationally expensive with large datasets.

In summary, Linear Regression is a simpler and more interpretable model that is suitable for cases where the relationship between variables is linear, and there are no outliers. Random Forest Regression, on the other hand, is a more complex and robust model that can handle nonlinear relationships, outliers, and large datasets better.

Differentiate between Random Forest Regression and L1 and L2 regularization:

- **Purpose:**
  Random Forest Regression is used to build an ensemble of decision trees that can make accurate predictions by combining the outputs of multiple decision trees. It is a non-linear regression algorithm that can capture complex relationships between input and output variables.

  L1/L2 regularization, on the other hand, is used to prevent overfitting in linear regression models. It adds a penalty term to the loss function to discourage the model from assigning too much weight to any single feature.

- **Model Complexity:**
  Random Forest Regression is a more complex model than linear regression because it involves combining multiple decision trees. The resulting model is more difficult to interpret and explain to stakeholders.

  L1/L2 regularization does not increase the complexity of the linear regression model, but instead reduces it by eliminating features with low weights. This can make the model more interpretable and easier to explain to stakeholders.

- **Handling of Outliers:**
  Random Forest Regression is robust to outliers, as the ensemble of decision trees can accommodate them and make accurate predictions even in the presence of outliers.

  L1/L2 regularization is sensitive to outliers, as outliers can have a large impact on the weights assigned to each feature. This can lead to a poorly performing model if the outliers are not handled properly.

- **Training Time:**
  Random Forest Regression can be computationally expensive to train, especially when using a large number of trees or features.

  L1/L2 regularization is computationally cheap and can be trained quickly, even on large datasets.

- **Hyperparameter Tuning:**
  Random Forest Regression requires tuning of several hyperparameters such as the number of trees, the maximum depth of the trees, and the minimum number of samples required to split a node. This can be time-consuming and requires some expertise to do well.

  L1/L2 regularization has a single hyperparameter, alpha, which controls the strength of the penalty term. It can be tuned using cross-validation to find the optimal value for the specific dataset.

In summary, Random Forest Regression and L1/L2 regularization are two different techniques used in regression analysis for different purposes. Random Forest Regression is a more complex, non-linear model used for accurate predictions, while L1/L2 regularization is used to prevent overfitting and improve interpretability of linear regression models.

KNN Regressor:

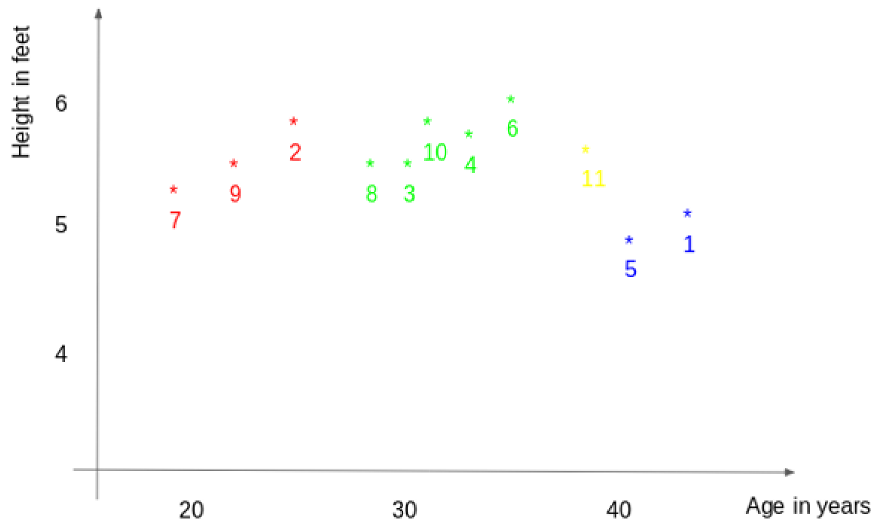
K-Nearest Neighbours (KNN) Regression is a non-parametric supervised learning algorithm used for both regression and classification tasks. In KNN regression, the value of the target variable for a new data point is predicted by finding the k closest data points to the new point and taking the average of their target values.

Understand the KNN Regression Algorithm:

Let us start with a simple example. Consider the following table – it consists of the height, age, and weight (target) values for 10 people. As you can see, the weight value of ID11 is missing. We need to predict the weight of this person based on their height and age.

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

For a clearer understanding of this, below is the plot of height versus age from the above table:



In the above graph, the y-axis represents the height of a person (in feet) and the x-axis represents the age (in years). The points are numbered according to the ID values. The yellow point (ID 11) is our test point.

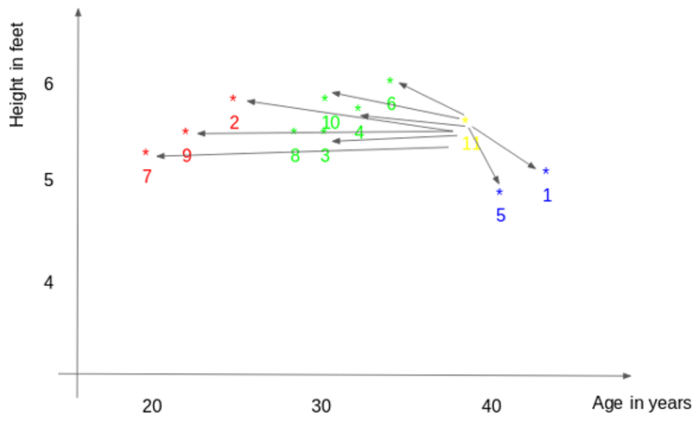
If I ask you to identify the weight of ID11 based on the plot, what would be your answer? You would likely say that since ID11 is **closer** to points 5 and 1, so it must have a weight similar to these IDs, probably between 72-77 kgs (weights of ID1 and ID5 from the table). That actually makes sense, but how do you think the algorithm predicts the values? We will find that out in this article.

How KNN Regression Work:
As we saw above, the KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses '**feature similarity**' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. From our example, we know that ID11 has height and age similar to ID1 and ID5, so the weight would also approximately be the same.

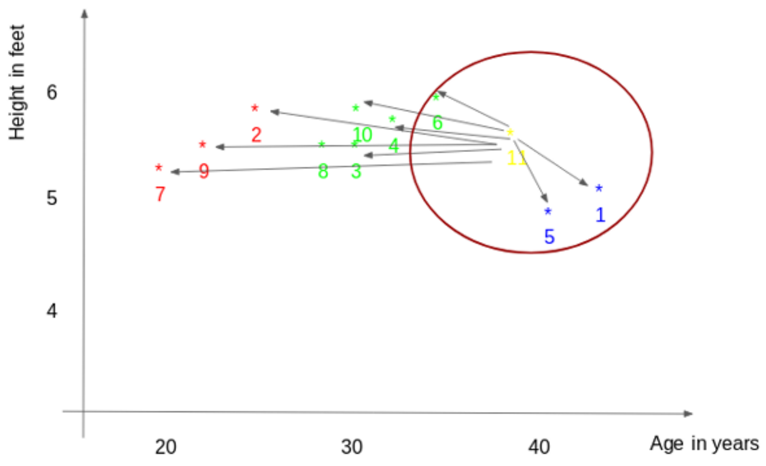
Had it been a classification problem, we would have taken the mode as the final prediction. In this case, we have two values of weight – 72 and 77. Any guesses on how the final value will be calculated? The average of the values is taken to be the final prediction.

Below is a stepwise explanation of the algorithm:

- First, the distance between the new point and each training point is calculated.



- The closest k data points are selected (based on the distance). In this example, points 1, 5, and 6 will be selected if the value of k is 3. We will further explore the method to select the right value of k later in this article.

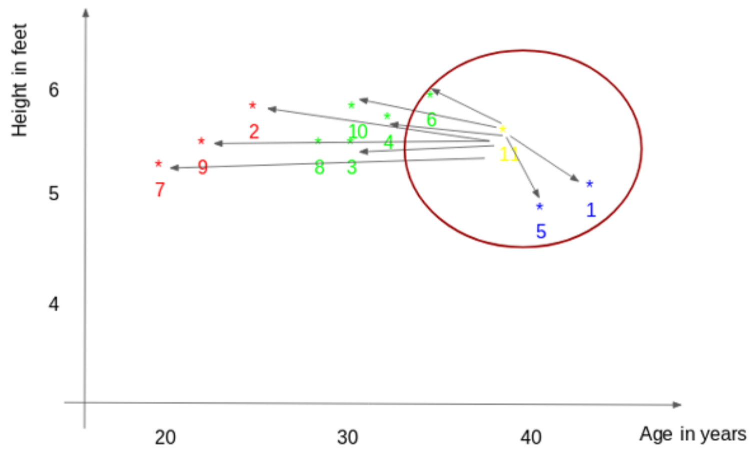


- The average of these data points is the final prediction for the new point. Here, we have the weight of ID11 = (77+72+60)/3 = 69.66 kg.

How to choose K-factor:

The **second step** is to select the k value. This determines the number of neighbours we look at when we assign a value to any new observation.

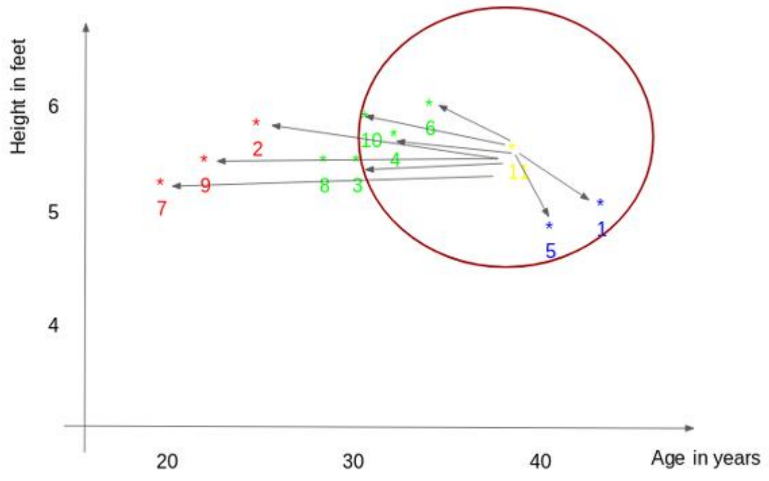
In our example, for a value k = 3, the closest points are ID1, ID5, and ID6.



ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60

The prediction of weight for ID11 will be:
ID11 = (77+72+60)/3
ID11 = 69.66 kg

For the value of k=5, the closest point will be ID1, ID4, ID5, ID6, and ID10.

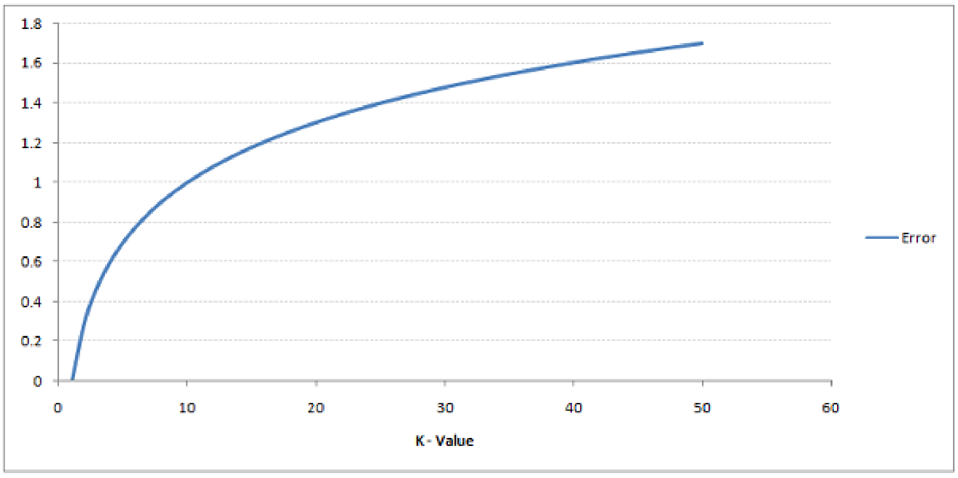


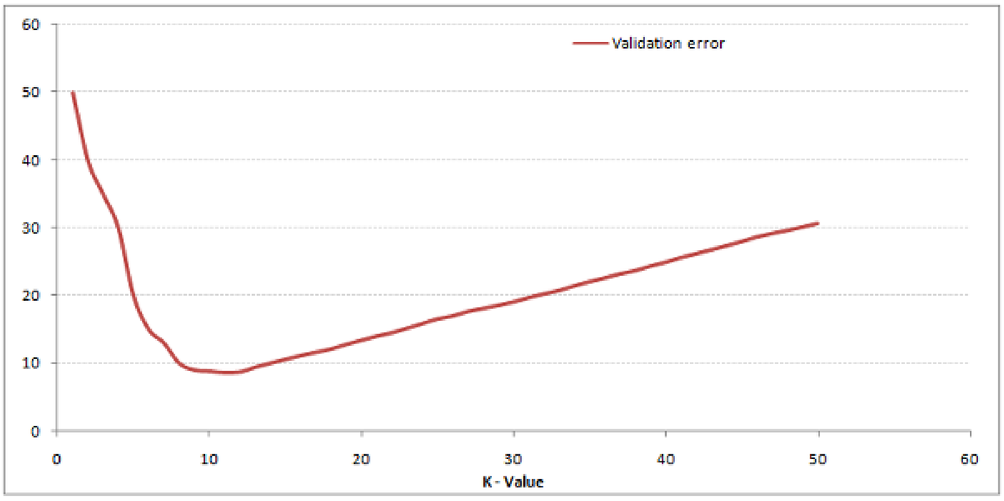
ID	Height	Age	Weight
1	5	45	77
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
10	5.6	32	58

The prediction for ID11 will be:
ID 11 = (77+59+72+60+58)/5
ID 11 = 65.2 kg

We notice that based on the k value, the final result tends to change. Then how can we figure out the optimum value of k? Let us decide based on the error calculation for our train and validation set (after all, minimizing the error is our final goal!).

Have a look at the below graphs for training error and validation error for different values of k.





For a very low value of k (suppose k=1), the model is overfitting the training data, which leads to a high error rate on the validation set. On the other hand, for a high value of k, the model performs poorly on both the train and validation sets. If you observe closely, the validation error curve reaches a minimum at a value of k = 9. This value of k is the optimum value of the model (it will vary for different datasets). This curve is known as an '**elbow curve**' (because it has a shape like an elbow) and is usually used to determine the k value.

You can also use the grid search technique to find the best k value. We will implement this in the next section.

Implementation using Python:

Step 1: Read the file

```
import pandas as pd
df = pd.read_csv('train.csv')
df.head()
```

Step 2: Impute missing values

```
df.isnull().sum()
#missing values in Item_weight and Outlet_size needs to be imputed
mean = df['Item_Weight'].mean() #imputing item_weight with mean
df['Item_Weight'].fillna(mean, inplace =True)

mode = df['Outlet_Size'].mode() #imputing outlet size with mode
df['Outlet_Size'].fillna(mode[0], inplace =True)
```

Step 3: Deal with categorical variables and drop the id columns

```
df.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1, inplace=True)
df = pd.get_dummies(df)
```

Step 4: Create a train and a test set

```
from sklearn.model_selection import train_test_split
train , test = train_test_split(df, test_size = 0.3)

x_train = train.drop('Item_Outlet_Sales', axis=1)
y_train = train['Item_Outlet_Sales']

x_test = test.drop('Item_Outlet_Sales', axis = 1)
y_test = test['Item_Outlet_Sales']
```

Step 5: Preprocessing – Scaling the features

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)

x_test_scaled = scaler.fit_transform(x_test)
x_test = pd.DataFrame(x_test_scaled)
```

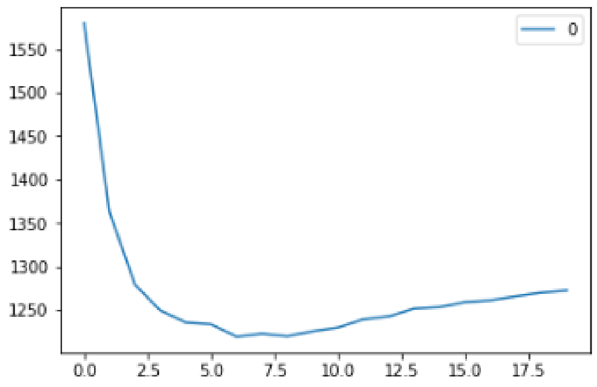
Step 6: Let us have a look at the error rate for different k values

```
#import required packages
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt
%matplotlib inline

rmse_val = [] #to store rmse values for different k
for K in range(20):
    K = K+1
    model = neighbors.KNeighborsRegressor(n_neighbors = K)

    model.fit(x_train, y_train)  #fit the model
    pred=model.predict(x_test) #make prediction on test set
    error = sqrt(mean_squared_error(y_test,pred)) #calculate rmse
    rmse_val.append(error) #store rmse values
    print('RMSE value for k= ' , K , 'is:', error)
```

Output:

```
RMSE value for k = 1 is: 1579.8352322344945
RMSE value for k = 2 is: 1362.7748806138618
RMSE value for k = 3 is: 1278.868577489459
RMSE value for k = 4 is: 1249.338516122638
RMSE value for k = 5 is: 1235.4514224035129
RMSE value for k = 6 is: 1233.2711649472913
RMSE value for k = 7 is: 1219.0633086651026
RMSE value for k = 8 is: 1222.244674933665
RMSE value for k = 9 is: 1219.5895059285074
RMSE value for k = 10 is: 1225.106137547365
RMSE value for k = 11 is: 1229.540283771085
RMSE value for k = 12 is: 1239.1504407152086
RMSE value for k = 13 is: 1242.3726040709887
RMSE value for k = 14 is: 1251.505810196545
RMSE value for k = 15 is: 1253.190119191363
RMSE value for k = 16 is: 1258.802262564038
RMSE value for k = 17 is: 1260.884931441893
RMSE value for k = 18 is: 1265.5133661294733
RMSE value for k = 19 is: 1269.619416217394
RMSE value for k = 20 is: 1272.10881411344
```

```
#plotting the rmse values against k values
curve = pd.DataFrame(rmse_val) #elbow curve
curve.plot()
```



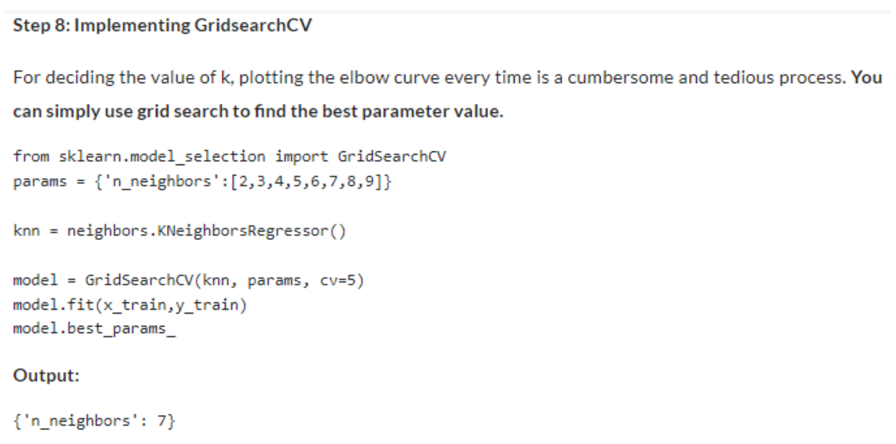
As we discussed, when we take k=1, we get a very high RMSE value. The RMSE value decreases as we increase the k value. At k= 7, the RMSE is approximately 1219.06 and shoots upon further increasing the k value. We can safely say that k=7 will give us the best result in this case.

These are the predictions using our training dataset. Let us now predict the values for test data and make a submission.

Step 7: Make predictions on the test dataset

```
#reading test and submission files
test = pd.read_csv('test.csv')
submission = pd.read_csv('SampleSubmission.csv')
submission['Item_Identifier'] = test['Item_Identifier']
submission['Outlet_Identifier'] = test['Outlet_Identifier']

#preprocessing test dataset
test.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1, inplace=True)
test['Item_Weight'].fillna(mean, inplace =True)
test = pd.get_dummies(test)
test_scaled = scaler.fit_transform(test)
test = pd.DataFrame(test_scaled)

#predicting on the test set and creating submission file
predict = model.predict(test)
submission['Item_Outlet_Sales'] = predict
submission.to_csv('submit_file.csv',index=False)
```

On submitting this file, I get an RMSE of 1279.5159651297.



Step 8: Implementing GridsearchCV

For deciding the value of k, plotting the elbow curve every time is a cumbersome and tedious process. **You can simply use grid search to find the best parameter value.**

```
from sklearn.model_selection import GridSearchCV
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}

knn = neighbors.KNeighborsRegressor()

model = GridSearchCV(knn, params, cv=5)
model.fit(x_train,y_train)
model.best_params_
```

Output:

```
{'n_neighbors': 7}
```

Advantages:

- Simple and easy to understand.
- Non-parametric, meaning it does not make any assumptions about the underlying distribution of the data.
- Can be used for both regression and classification tasks.

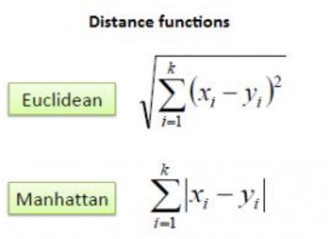
Disadvantages:

- Computationally expensive, especially for large datasets.
- Can be sensitive to the choice of distance metric.
- Can be sensitive to outliers and irrelevant features in the data.
- Requires a large amount of memory to store the training set.

How to calculate the distance between the points:

- **Euclidean Distance:** Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).

- **Manhattan Distance**: This is the distance between real vectors using the sum of their absolute difference.



Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

- **Hamming Distance**: It is used for categorical variables. If the value (x) and the value (y) are the same, the distance D will be equal to 0. Otherwise, D=1.

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Differentiate between Random Forest Regression and KNN Regression:

- **Algorithm:** Random Forest Regression is an ensemble learning method that uses multiple decision trees to make predictions. KNN Regression, on the other hand, is a non-parametric method that estimates the value of a new data point by averaging the values of its k-nearest neighbours in the training set.

- **Model complexity:** Random Forest Regression can handle non-linear and complex relationships between the input features and the target variable, while KNN Regression assumes a linear relationship between the input features and the target variable. Therefore, Random Forest Regression can be more powerful in modelling complex relationships and can capture non-linear patterns better than KNN Regression.

- **Training time:** Random Forest Regression can be computationally expensive and time-consuming to train, especially when dealing with large datasets or a large number of decision trees. KNN Regression, on the other hand, has no training phase, and the prediction time is fast as it simply computes the distances between the new data point and the training data points.

- **Interpretability:** Random Forest Regression can be difficult to interpret as it uses multiple decision trees to make predictions. KNN Regression, on the other hand, can be more interpretable as it simply averages the values of the nearest neighbours to estimate the value of a new data point.

In summary, Random Forest Regression and KNN Regression are both useful regression algorithms, but they have different strengths and weaknesses, and the choice between them depends on the specific task and the characteristics of the dataset.

Overall, the choice of algorithm depends on the nature of the data and the problem at hand. Random Forest Regression is a good choice for non-linear problems and high-dimensional data, while Linear Regression is suitable for linear problems with few features. L1 and L2 regularization can help in reducing overfitting and perform feature selection. KNN Regression is useful when there is no clear relationship between the features and the target variable and when non-linearity needs to be handled.

Support Vector Regression (SVR):

Support Vector Regression (SVR) is a supervised learning algorithm used for regression analysis. It is a variant of Support Vector Machine (SVM) that is used for classification. SVR is used to find the best fit line or hyperplane for data that is not linearly separable. It maps the data to a high-dimensional space and finds the optimal hyperplane that maximizes the margin to predict the output.

The key features of SVR are:

- It can handle nonlinear data by mapping it to a higher-dimensional space.
- It finds the optimal hyperplane by maximizing the margin between the support vectors.
- It uses kernel functions to transform the input space to a higher-dimensional space.
- It can handle outliers and noise in the data.

Advantages:

- It is effective in handling high-dimensional data with a small sample size, making it a useful tool in many applications such as image and speech recognition.
- It has a strong theoretical foundation and has been extensively studied in the literature.
- It can handle both linear and nonlinear data with high accuracy, making it a powerful regression technique.
- It is robust to outliers in the data due to the use of the ε-insensitive loss function.

Disadvantages:

- It can be computationally expensive to train on large datasets with many features, and may require tuning of hyperparameters such as the kernel and regularization parameters.
- The choice of kernel function can be critical and may require some domain knowledge or trial-and-error to determine the best one for a particular problem.
- It can be sensitive to the choice of hyperparameters and can lead to overfitting if not properly tuned.
- Interpretation of the model can be difficult, as the resulting model is often complex and difficult to understand.

Compare between SVR and Random Forest Regression and KNN Regression

SVR (Support Vector Regression), Random Forest Regression, and KNN (K-Nearest Neighbour) Regression are all popular algorithms for regression analysis. Here are some key differences between them:

- **Complexity:** SVR is a more complex algorithm than both Random Forest Regression and KNN Regression.
- **Interpretability:** KNN and Random Forests are considered more interpretable than SVR because they do not involve complex mathematical calculations and are based on more intuitive concepts.
- **Outliers:** SVR is more sensitive to outliers than Random Forest and KNN. Outliers can greatly influence the model performance of SVR as they can affect the location of the decision boundary.
- **Feature Selection:** Random Forest and KNN have built-in feature selection capabilities while SVR does not. However, feature selection can be done before using SVR.
- **Hyperparameter Tuning:** All three algorithms require hyperparameter tuning to obtain optimal results. However, the parameters required for tuning are different for each algorithm.
- **Performance:** The performance of the algorithms depends on the dataset and problem at hand. In some cases, SVR may outperform Random Forest and KNN, while in other cases, the opposite may be true.

In summary, SVR is a more complex algorithm that may perform better than Random Forest and KNN in certain cases, but it is also more sensitive to outliers. Random Forest and KNN are simpler algorithms that are more interpretable and have built-in feature selection capabilities. However, they may not perform as well as SVR in certain cases.

SVR Kernal:

Support Vector Regression (SVR) uses kernel functions to transform the original input space into a higher-dimensional feature space, where a linear regression model can be fit. The kernel function is a mathematical function that takes two arguments, computes the similarity between them, and outputs a scalar value. The idea behind using a kernel function is to implicitly map the data into a higher-dimensional space where it is more likely that a linear model can fit the data.

Some common kernel functions used in SVR are:

- **Linear kernel:** The linear kernel is the simplest kernel function, and it is equivalent to a standard linear regression model. It is used when the data is already linearly separable or when the number of features is very large.

- **Polynomial kernel:** The polynomial kernel function is used when the data is not linearly separable. It maps the data into a higher-dimensional space using a polynomial function.

- **Radial basis function (RBF) kernel:** The RBF kernel is the most commonly used kernel function in SVR. It is a type of **Gaussian kernel** that maps the data into an infinite-dimensional space. It is useful when the data has a non-linear relationship.

- **Sigmoid kernel:** The sigmoid kernel is used when the data has a sigmoid shape. It maps the data into a higher-dimensional space using a sigmoid function.

Gaussian Kernal:

The Gaussian kernel is a commonly used kernel function in machine learning, particularly in support vector machines (SVMs). It is also known as the radial basis function (RBF) kernel. The Gaussian kernel is a similarity function that calculates the similarity between two data points, based on their distance from each other in a high-dimensional space.

The formula for the Gaussian kernel is as follows:

$K(x, y) = \exp(-\text{gamma} * ||x-y||^2)$

Here, x and y are data points, gamma is a hyperparameter that controls the width of the kernel, and $||x-y||^2$ is the Euclidean distance between the two data points.

The Gaussian kernel assigns a high similarity score to data points that are close to each other and a low similarity score to data points that are far away from each other. The kernel is also sensitive to outliers, as data points that are far away from the majority of the data points will have a low similarity score.

In SVMs, the Gaussian kernel is used to transform the data points into a higher-dimensional space, where it becomes easier to separate the data points into different classes or groups. The kernel function calculates the dot product between the transformed data points, which can then be used to determine the decision boundary between different classes.

Overall, the Gaussian kernel is a powerful tool for modelling complex relationships between data points, but it can be sensitive to the choice of hyperparameters, such as gamma.

The choice of kernel function depends on the problem at hand and the characteristics of the data. The hyperparameters of the kernel function, such as the degree of the polynomial kernel and the width of the Gaussian kernel, need to be tuned using cross-validation to find the optimal values.

In summary, SVR uses kernel functions to transform the data into a higher-dimensional space where a linear regression model can be fit. The choice of kernel function and its hyperparameters can significantly affect the performance of the model.

Gaussian Process Regression (GPR):

Gaussian Process Regression (GPR) is a non-parametric probabilistic (Bayesian Approach) machine learning algorithm used for regression analysis. In GPR, the relationship between input variables (X) and output variables (y) is modelled as a Gaussian process. The goal is to make predictions on new, unseen data points based on the distribution of the training data.

A Gaussian process is defined by a mean function and a covariance function. The mean function represents the expected value of the output variable for a given input, while the covariance function determines how the output variable varies with the input variables. The covariance function is typically defined as a kernel function that measures the similarity between input variables.

During the training phase, GPR estimates the hyperparameters of the kernel function to fit the training data. The model can then be used to make predictions on new data points by computing the posterior distribution of the output variable at each point. The posterior distribution represents the probability distribution of the output variable given the input variables.

GPR has several advantages over other regression algorithms. It can handle non-linear relationships between input and output variables, and it provides a measure of uncertainty in the predictions, which is useful in many applications. However, GPR can be computationally expensive for large datasets and is sensitive to the choice of kernel function and hyperparameters.

Compare between Gaussian Process Regression and SVR:

- **Model:** GPR is a probabilistic model that models the uncertainty of the predictions, while SVR is a non-probabilistic model that aims to minimize the error between the predicted and actual values.

- **Training data:** GPR can handle small to moderate sized datasets, while SVR can handle large datasets with high-dimensional features.

- **Hyperparameters:** GPR requires tuning of hyperparameters such as kernel function and noise level, while SVR requires tuning of hyperparameters such as kernel function, regularization parameter, and epsilon.

- **Interpretability:** GPR provides a measure of uncertainty and can model non-linear relationships, but it may be harder to interpret the model due to the probabilistic nature of the predictions. SVR provides a simpler model that is easier to interpret, but may not be as flexible in modelling non-linear relationships.

In summary, GPR is better suited for small to moderate sized datasets where uncertainty estimation is important, while SVR is better suited for large datasets with high-dimensional features and when interpretability is important.

Performance Metrics for Regression Problem:

- **Mean Absolute Error (MAE):** This metric calculates the absolute difference between the predicted and actual values and takes the mean of those values. It is given by the formula:
  $MAE = (1/n) * \sum|y_i - \hat{y}_i|$
  where n is the number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

- **Mean Squared Error (MSE):** This metric calculates the squared difference between the predicted and actual values and takes the mean of those values. It is given by the formula:
  $MSE = (1/n) * \sum (y_i - \hat{y}_i) ^2$

- **Root Mean Squared Error (RMSE):** This metric is the square root of MSE and is used to measure the standard deviation of the errors. It is given by the formula:
  $RMSE = sqrt((1/n) * \sum (y_i - \hat{y}_i) ^2)$

- **R-squared (R2):** This metric measures the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It is given by the formula:
  $R2 = 1 - (RSS / TSS)$
  where RSS is the residual sum of squares and TSS is the total sum of squares.

- **Mean Absolute Percentage Error (MAPE):** This metric calculates the percentage difference between the predicted and actual values and takes the mean of those values. It is given by the formula:
  $MAPE = (1/n) * \sum(|(y_i - \hat{y}_i) / y_i|) * 100$
  where n is the number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

- **Adjusted R-Square:** Adjusted R-squared is a modified version of the R-squared statistic used in regression analysis. It is designed to provide a more accurate measure of the goodness of fit of a regression model by taking into account the number of independent variables used in the model.

  $\text{Adjusted R-squared} = 1 - [(1 - R^2) * (n - 1) / (n - k - 1)]$

  where:
  R^2 is R-square value.
  n is the sample size
  k is the number of independent variables in the model

The adjusted R-squared ranges from 0 to 1, where a value of 1 indicates a perfect fit between the model and the data. A higher adjusted R-squared indicates that the model is better able to explain the variation in the dependent variable, while a lower adjusted R-squared indicates that the model is less effective at explaining the variation.

Adjusted R-squared is particularly useful when comparing regression models that contain different numbers of independent variables. It allows us to determine which model provides a better fit to the data, taking into account the complexity of the model.

Note:

MSE (Mean Square Error):

It is not possible to determine whether a mean squared error (MSE) value of 189832 is good or bad for a model's performance without any additional information about the problem being solved and the range of values for the target variable.

In general, a lower MSE indicates better performance of the model in predicting the target variable. However, the interpretation of what constitutes a "good" MSE value varies depending on the specific problem and context.

For example, if the target variable has a wide range of values, such as predicting house prices, then an MSE of 189832 may be considered acceptable. On the other hand, if the target variable has a narrow range of values, such as predicting exam scores on a 0-100 scale, then an MSE of 189832 would likely indicate poor performance.

Therefore, it is important to consider the specific context of the problem and compare the MSE value to other models or benchmarks to determine whether the performance is satisfactory or not.

RMSE (Root Mean Square Error):

In the case of root mean square error (RMSE), a lower value indicates better performance of the model. However, what constitutes a "good" RMSE value depends on the context of the problem and the data being used. Therefore, it is difficult to specify a universal threshold for what is considered a "good" RMSE value. In general, a model with an RMSE value that is significantly lower than the standard deviation of the target variable can be considered a good model.

MAE (Mean Absolute Error):

The best value for mean absolute error (MAE) for model performance depends on the specific problem and the context in which it is being used. In general, a lower MAE indicates better performance, but the acceptable range of values depends on the specific domain and the magnitude of the target variable.

For example, if the target variable is measured in very large units (e.g., thousands or millions), then an MAE of a few hundred units may be acceptable, whereas if the target variable is measured in small units (e.g., tenths or hundredths), then an MAE of a few units may be necessary for good performance. It is also important to compare the MAE of a model to the variability of the target variable to get a sense of the scale of the errors.

MAPE (Mean Absolute Percentage Error):

There is no specific value of Mean Absolute Percentage Error (MAPE) that can be considered as the best for model performance. It depends on the specific problem and the domain.

In general, a lower MAPE value indicates better model performance. However, MAPE has limitations and should be used with caution. For example, if the actual value is close to zero, then MAPE can become very large even for small prediction errors.

Therefore, it is recommended to use multiple evaluation metrics to assess the performance of a model.

R-Square:

The R-squared value, also known as the coefficient of determination, is a measure of how well the regression line fits the data. It takes values between 0 and 1, where a value of 1 indicates a perfect fit and a value of 0 indicates no relationship between the dependent and independent variables.

In general, a higher R-squared value indicates a better fit between the model and the data, but the threshold for a "good" R-squared value can vary depending on the context and the specific problem being solved. As a rule of thumb, an R-squared value of 0.7 or higher is often considered good for many applications.

However, it is important to keep in mind that R-squared is not the only metric to consider when evaluating a model's performance, and it should be used in conjunction with other metrics such as mean squared error, mean absolute error, and so on. Additionally, it is important to ensure that the model is not overfitting to the training data, as this can lead to a high R-squared value but poor generalization performance on new, unseen data.