**Train-Delay-Prediction**

The "Train-Delay-Prediction" project focuses on analyzing historic train data to gain insights and develop a predictive model for predicting train delays. By leveraging machine learning algorithms like Random Forest Regressor and Gradient Boosting Regressor, the project aims to predict whether a train will be delayed based on various characteristics.

The objective of the project is to perform analysis of the historic train data to gain valuable insights and build a predictive model to predict whether a train will be delayed or not given a set of train characteristics.

The objective of the predictive model is to build a model to predict whether a train will be delayed or not based on certain characteristics of the train. Such a model may help both passengers as well as railways to predict future delays and minimize them. The model is trained with a dataset which contains train data including train names, station names, timings and historical data about train delays.

The output shows whether the train is delayed or not depends on the time; if the train is late by 15 minutes or more then it is declared as delayed.

**\*\*Technologies Used:\*\***

1. Numpy

2. Pandas

3. Matplotlib

4. Seaborn

5. Sklearn

6. Random Forest Regressor from Sklearn

7. Gradient Boosting Regressor from Sklearn

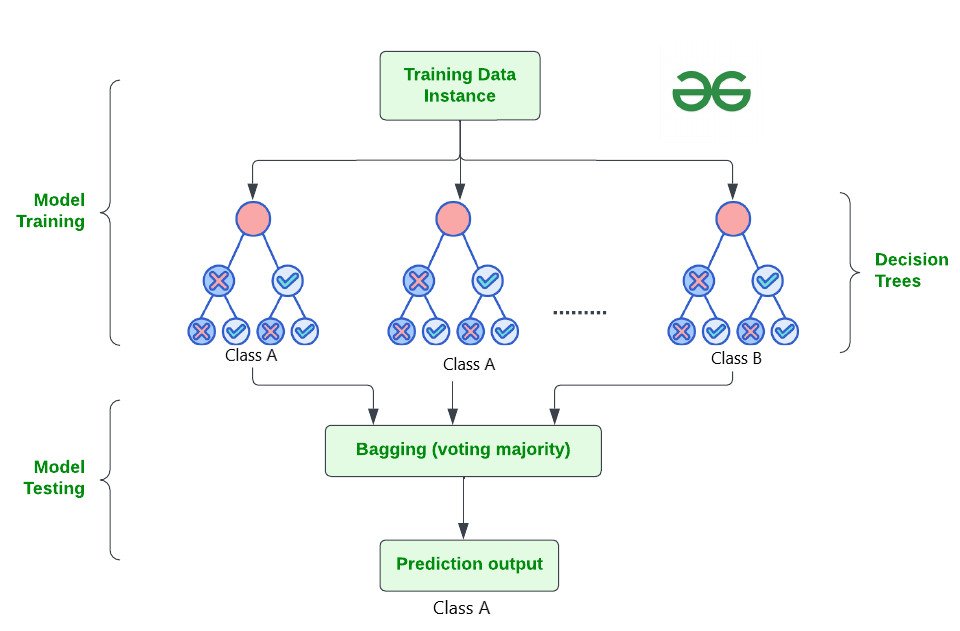
**RANDOM FOREST REGRESSION**

Random Forest Regression is a machine learning algorithm used for regression tasks, which involves predicting a continuous value based on input features. It is an ensemble learning method, meaning it combines the predictions of multiple individual models (trees) to improve the overall accuracy and robustness of the model.

Here's how the Random Forest Regression algorithm works:

1. **Random Forest Overview:** A Random Forest is made up of a collection of decision trees. Each tree in the forest is trained independently on a random subset of the data (bootstrap sample) and a random subset of the features. This randomness helps to ensure that each tree in the forest is different from the others.
2. **Decision Trees:** Each decision tree in the Random Forest makes a prediction by recursively splitting the data into subsets based on the feature values. The splits are chosen to minimize the variance of the target variable (i.e., reduce the error in prediction). The final prediction of the tree is the average (for regression) of the target values of the training instances in the leaf node.
3. **Combining Predictions:** In a Random Forest, the predictions of all the trees are combined to make the final prediction. For regression tasks, this is typically done by averaging the predictions of all the trees in the forest.
4. **Advantages of Random Forest Regression:**
   * Handles large datasets with higher dimensionality well.
   * Can handle missing values in the dataset.
   * Reduces overfitting compared to a single decision tree.
   * Provides feature importances, which can be useful for understanding the impact of different features on the prediction.

Overall, Random Forest Regression is a powerful and versatile algorithm that is widely used for regression tasks in various fields, including finance, healthcare, and transportation.



**Advantages of Random Forest Regressor from Sklearn:**

1. **High Accuracy:** Random forests typically have higher accuracy compared to single decision trees. They reduce overfitting by averaging multiple trees.
2. **Robustness:** They are less likely to overfit on the training data compared to other models. They handle outliers and noisy data well.
3. **Feature Importance:** Random forests provide a feature importance score, which helps in understanding the relative importance of different features in making predictions.
4. **Efficiency:** They can handle large datasets with higher dimensionality efficiently. The training process can be parallelized, making it faster.
5. **Versatility:** Random forests can be used for both regression and classification tasks. They also work well with both numerical and categorical features.
6. **Reduced Variance:** By averaging multiple trees, random forests reduce the variance of the model, making them more stable and less sensitive to noise in the data.

**Advantages of Gradient Boosting Regressor from Sklearn:**

1. **High Predictive Power:** Gradient Boosting Regressor typically provides higher predictive power compared to other algorithms due to its ensemble nature.
2. **Handles Non-linear Relationships:** It can capture complex non-linear relationships in the data, making it suitable for a wide range of regression problems.
3. **Robustness to Outliers:** Gradient Boosting is robust to outliers in the data, as it builds trees sequentially, focusing on reducing errors from previous trees.
4. **Feature Importance:** Similar to Random Forests, Gradient Boosting provides feature importance scores, helping in feature selection and understanding the data.
5. **Less Preprocessing Required:** It can handle missing data and does not require extensive preprocessing of the data such as normalization or scaling.
6. **Flexibility:** Gradient Boosting can be used for both regression and classification tasks, making it a versatile algorithm.

In the train delay prediction project, Random Forest Regression model is implemented using the **RandomForestRegressor** class from the **sklearn.ensemble** module. Here's a breakdown of how it is implemented:

1. **Loading Data:**
   * The **loadData** function reads the data from CSV files (**trains.csv** and **stations.csv**) using pandas.
   * It preprocesses the data by dropping unnecessary columns and encoding categorical variables.
2. **Preprocessing:**
   * The **preprocessing** function splits the data into features (**X**) and target (**Y**) variables.
   * It further splits the data into training and testing sets using **train\_test\_split** function from **sklearn.model\_selection**.
3. **Training the Random Forest Regressor:**
   * The **rfg** function creates an instance of **RandomForestRegressor**.
   * It fits the model on the training data using the **fit** method.
4. **Accepting User Input:**
   * The **accept\_data** function takes user input for month, day, scheduled departure time, distance, arrival delay, train code, origin station code, destination station code, and day of the week.
5. **Making Predictions:**
   * The **prediction** function prepares the input vector for the trained model based on the user input.
   * It then uses the trained **RandomForestRegressor** model to predict whether the train will be delayed or not based on the input data.
6. **Displaying Prediction:**
   * The **main** function is the entry point of the application.
   * It uses Streamlit to create a web interface where users can select the machine learning model (in this case, only the Random Forest Regressor is available) and input their data.

The main use of the project is to provide a predictive model for estimating train delays, which can be valuable for both railways and passengers. By analyzing historical train data and considering various factors such as departure time, railway, and station, the model can forecast the likelihood of a train being delayed by 15 minutes or more. This information can help railways to optimize their scheduling and operations, potentially reducing delays and improving overall efficiency. For passengers, knowing the likelihood of a delay can aid in planning travel itineraries and making informed decisions. Overall, the project aims to enhance the efficiency and reliability of train travel by leveraging machine learning algorithms to predict and mitigate train delays.

Top of Form

Top of Form