**ADVANCED SUPERVISED LEARNING ALGORITHMS**

**What is Classification in Machine Learning?**

[Classification in machine learning](https://www.geeksforgeeks.org/machine-learning/getting-started-with-classification/) is a type of supervised learning approach where the goal is to predict the category or class of an instance that are based on its features. In classification it involves training model ona dataset that have instances or observations that are already labeled with Classes and then using that model to classify new, and unseen instances into one of the predefined categories.

**List of Machine Learning Classification Algorithms**

Classification algorithms organize and understand complex datasets in machine learning. These algorithms are essential for categorizing data into classes or labels, automating decision-making and pattern identification. Classification algorithms are often used to detect email spam by analyzing email content. These algorithms enable machines to quickly recognize spam trends and make real-time judgments, improving email security.

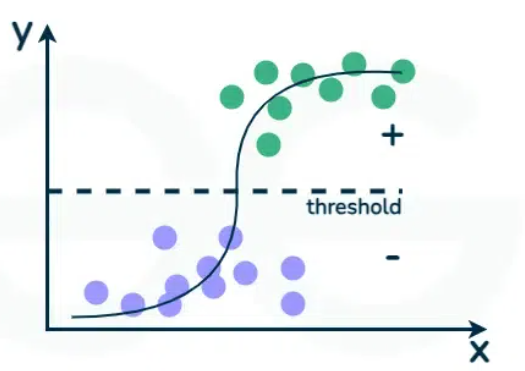
Some of the top-ranked machine learning algorithms for Classification are:

1. **Logistic Regression**
2. **Decision Tree**
3. **Random Forest**
4. **Support Vector Machine (SVM)**
5. **Naive Bayes**
6. **K-Nearest Neighbors (KNN)**

## **1. Logistic Regression Classification Algorithm in Machine Learning**

In Logistic regression is classification algorithm used to estimate discrete values, typically binary, such as***0 and 1, yes or no.*** It predicts the probability of an instance belonging to a class that makes it essectial for binary classification problems like spam detection or diagnosing disease.

Logistic functions are ideal for classification problems since their output is between **0 and 1.** Many fields employ it because of its simplicity, interpretability, and efficiency. ***Logistic Regression*** works well when features and event probability are linear. Logistic Regression used for binary classification tasks. Logistic regression is used for binary categorization. Despite its name, it predicts class membership likelihood. A logistic function models probability in this linear model.



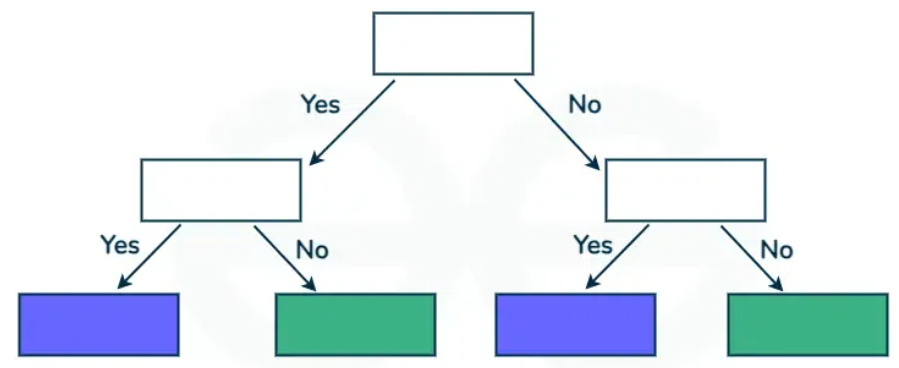
**Features of Logistic Regression**

1. **Binary Outcome: Logistic regression** is used when the dependent variable is binary in nature, meaning it has only two possible outcomes **(e.g., yes/no, 0/1, true/false).**
2. **Probabilistic Results:** It predicts the probability of the occurrence of an event by fitting data to a logistic function. The output is a value between **0 and 1**, which represents the probability that a given input belongs to the '1' category.
3. **Odds Ratio:**It estimates the odds ratio in the presence of more than one explanatory variable. The odds ratio can be used to understand the strength of the association between the independent variables and the dependent binary variable.
4. **Logit Function:**Logistic regression uses the logit function (**or logistic function**) to model the data. The logit function is an**S-shaped curve** that can take any real-valued number and map it into a value between **0 and 1.**

**2. Decision Tree**

Decision Trees are versatile and simple ***classification and regression*** techniques. Recursively splitting the dataset into key-criteria subgroups provides a tree-like structure. Judgments at each node produce leaf nodes. Decision trees are easy to understand and depict, making them useful for decision-making. Overfitting may occur, therefore trimming improves generality. A tree-like model of decisions and their consequences, including chance event outcomes, resource costs and utility.

The algorithm used for both classification and regression tasks. They model decisions and their possible results as tree, with branches representing choices and leaves representing outcomes.



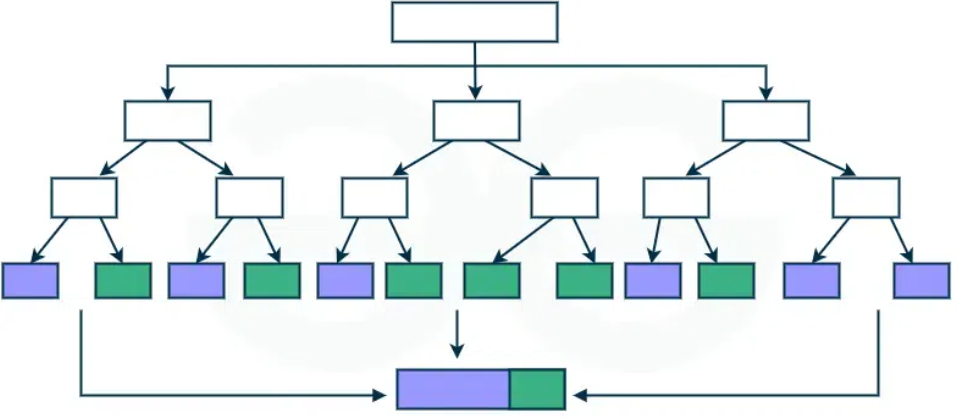
**Features of Decision Tree**

1. **Tree-Like Structure:**Decision Trees have a flowchart-like structure, where each internal node represents a "**test"** on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.
2. **Simple to Understand and Interpret**: One of the main advantages of Decision Trees is their simplicity and ease of interpretation. They can be visualized, which makes it easy to understand how decisions are made and explain the reasoning behind predictions.
3. **Versatility**: Decision Trees can handle both numerical and categorical data and can be used for both regression and classification tasks, making them versatile across different types of data and problems.
4. **Feature Importance**: Decision Trees inherently perform feature selection, giving insights into the most significant variables for making the predictions. The top nodes in a tree are the most important features, providing a straightforward way to identify critical variables.

**3. Random Forest**

Random forest are an ensemble learning techniques that combines multiple decision trees to improve predictive accuracy and control over-fitting. By aggregating the predictions of numerous trees, Random Forests enhance the decision-making process, making them robust against noise and bias.

Random Forest uses numerous decision trees to increase prediction accuracy and reduce overfitting. It constructs many trees and integrates their predictions to create a reliable model. Diversity is added by using a random dataset and characteristics in each tree. Random Forests excel at high-dimensional data, feature importance metrics, and overfitting resistance. Many fields use them for classification and regression.



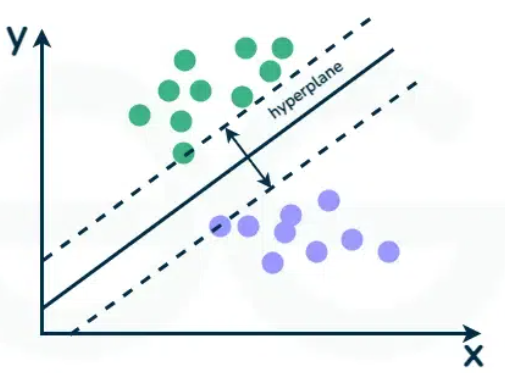
**Features of Random Forest**

1. **Ensemble Method**: Random Forest uses the ensemble learning technique, where multiple learners (decision trees, in this case) are trained to solve the same problem and combined to get better results. The ensemble approach improves the model's accuracy and robustness.
2. **Handling Both Types of Data**: It can handle both categorical and continuous input and output variables, making it versatile for different types of data.
3. **Reduction in Overfitting**: By averaging multiple trees, Random Forest reduces the risk of overfitting, making the model more generalizable than a single decision tree.
4. **Handling Missing Values**: Random Forest can handle missing values. When it encounters a missing value in a variable, it can use the median for numerical variables or the mode for categorical variables of all samples reaching the node where the missing value is encountered.

**4.Support Vector Machine (SVM)**

SVM is an effective classification and regression algorithm. It seeks the hyperplane that best classifies data while increasing the margin. SVM works well in high-dimensional areas and handles nonlinear feature interactions with its kernel technique. It is powerful classification algorithm known for their accuracy in high-dimensional spaces

SVM is robust against overfitting and generalizes well to different datasets. It finds applications in image recognition, text classification, and bioinformatics, among other fields. Its use cases span image recognition, text categorization, and bioinformatics, where precision is paramount.



**Feature of Support Vector Machine**

1. **Margin Maximization**: SVM aims to find the hyperplane that separates different classes in the feature space with the maximum margin. The margin is defined as the distance between the hyperplane and the nearest data points from each class, known as support vectors. Maximizing this margin increases the model's robustness and its ability to generalize well to unseen data.
2. **Support Vectors**: The algorithm is named after these support vectors, which are the critical elements of the training dataset. The position of the hyperplane is determined based on these support vectors, making SVMs relatively memory efficient since only the support vectors are needed to define the model.
3. **Kernel Trick**: One of the most powerful features of SVM is its use of kernels, which allows the algorithm to operate in a higher-dimensional space without explicitly computing the coordinates of the data in that space. This makes it possible to handle non-linearly separable data by applying linear separation in this higher-dimensional feature space.
4. **Versatility**: Through the choice of the kernel function (linear, polynomial, radial basis function (RBF), sigmoid, etc.), SVM can be adapted to solve a wide range of problems, including those with complex, non-linear decision boundaries.

**5.Naive Bayes**

Text categorization and spam filtering benefit from Bayes theorem-based probabilistic classification algorithm Naive Bayes. Despite its simplicity and **"naive"**assumption of feature independence, Naive Bayes often works well in practice. It uses conditional probabilities of features to calculate the class likelihood of an instance. Naive Bayes handles high-dimensional datasets quickly.

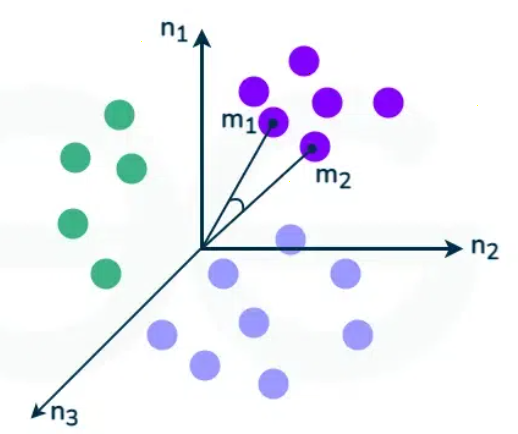
Naive Bayes which describes the probability of an event, based on prior knowledge of conditions that might be related to the event. Naive Bayes classifiers assume that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable

**Features of Naive Bayes**

1. **Probabilistic Foundation**: Naive Bayes classifiers apply Bayes' theorem to compute the probability that a given instance belongs to a particular class, making decisions based on the posterior probabilities.
2. **Feature Independence**: The algorithm assumes that the features used to predict the class are independent of each other given the class. This assumption, although naive and often violated in real-world data, simplifies the computation and is surprisingly effective in practice.
3. **Efficiency**: Naive Bayes classifiers are highly efficient, requiring a small amount of training data to estimate the necessary parameters (probabilities) for classification.
4. **Easy to Implement and Understand**: The algorithm is straightforward to implement and interpret, making it accessible for beginners in machine learning. It provides a good starting point for classification tasks.

**6.K-Nearest Neighbors (KNN)**

KNN uses the majority class of k-nearest neighbours for easy and adaptive classification and regression. **Non-parametric KNN** has no data distribution assumptions. It works best with uneven decision boundaries and performs well for varied jobs. K-Nearest Neighbors (KNN) is an instance-based, or lazy learning algorithm, where the function is only approximated locally, and all computation is deferred until function evaluation. It classifies new cases based on a similarity measure (e.g., distance functions). KNN is widely used in recommendation systems, anomaly detection, and pattern recognition due to its simplicity and effectiveness in handling non-linear data.



**Fetures of K-Nearest Neighbors (KNN)**

1. **Instance-Based Learning**: KNN is a type of instance-based or lazy learning algorithm, meaning it does not explicitly learn a model. Instead, it memorizes the training dataset and uses it to make predictions.
2. **Simplicity**: One of the main advantages of KNN is its simplicity. The algorithm is straightforward to understand and easy to implement, requiring no training phase in the traditional sense.
3. **Non-Parametric**: KNN is a non-parametric method, meaning it makes no underlying assumptions about the distribution of the data. This flexibility allows it to be used in a wide variety of situations, including those where the data distribution is unknown or non-standard.
4. **Flexibility in Distance Choice**: The algorithm's performance can be significantly influenced by the choice of distance metric (e.g., Euclidean, Manhattan, Minkowski). This flexibility allows for customization based on the specific characteristics of the data.

**Random Forest Regression in Python**

A random forest is an ensemble learning method that combines the predictions from multiple decision trees to produce a more accurate and stable prediction. It is a type of supervised learning algorithm that can be used for both classification and regression tasks.

In regression task we can use **Random Forest Regression** technique for predicting numerical values. It predicts continuous values by averaging the results of multiple decision trees.

**Working of Random Forest Regression**

Random Forest Regression works by creating multiple of decision trees each trained on a random subset of the data. The process begins with Bootstrap sampling where random rows of data are selected with replacement to form different training datasets for each tree. After this we do **feature sampling** where only a random subset of features is used to build each tree ensuring diversity in the models.

After the trees are trained each tree make a prediction and the final prediction for regression tasks is the average of all the individual tree predictions and this process is called as **Aggregation.**

This approach is beneficial because individual decision trees may have high variance and are prone to overfitting especially with complex data. However by averaging the predictions from multiple decision trees Random Forest minimizes this variance leading to more accurate and stable predictions and hence improving generalization of model.

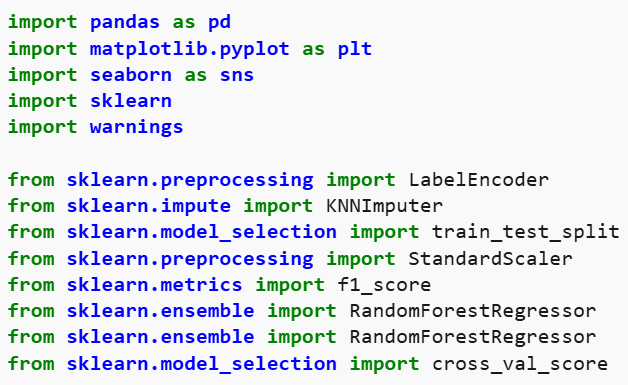
**Implementing Random Forest Regression in Python**

We will be implementing random forest regression on salaries data.

**1. Importing Libraries**

Here we are importing numpy, pandas, matplotlib, seaborn and scikit learn.

* **RandomForestRegressor:** This is the regression model that is based upon the Random Forest model.
* **LabelEncoder:** This class is used to encode categorical data into numerical values.
* **KNNImputer:** This class is used to impute missing values in a dataset using a k-nearest neighbors approach.
* **train\_test\_split:** This function is used to split a dataset into training and testing sets.
* **StandardScaler:** This class is used to standardize features by removing the mean and scaling to unit variance.
* **f1\_score:** This function is used to evaluate the performance of a classification model using the F1 score.
* **RandomForestRegressor:** This class is used to train a random forest regression model.
* **cross\_val\_score:** This function is used to perform k-fold cross-validation to evaluate the performance of a model



**2. Importing Dataset**

Now let's load the dataset in the panda's data frame. For better data handling and leveraging the handy functions to perform complex tasks in one go.

**3. Data Preparation**

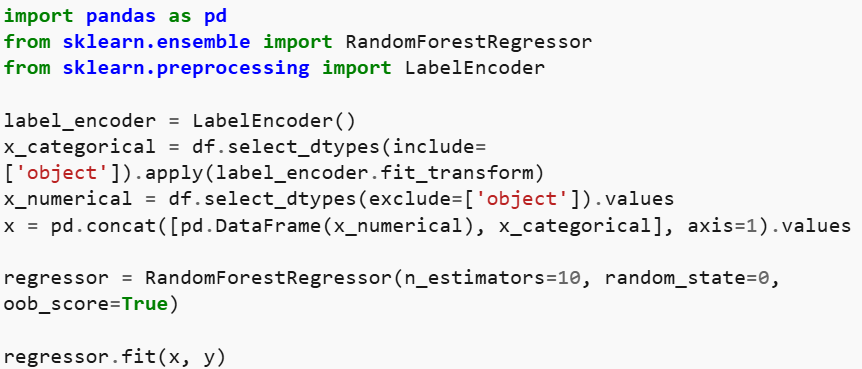
Here the code will extracts two subsets of data from the Dataset and stores them in separate variables.

* **Extracting Features:**It extracts the features from the DataFrame and stores them in a variable named X.
* **Extracting Target Variable:** It extracts the target variable from the DataFrame and stores it in a variable named y.

**4. Random Forest Regressor Model**

The code processes categorical data by encoding it numerically, combines the processed data with numerical data and trains a Random Forest Regression model using the prepared data.

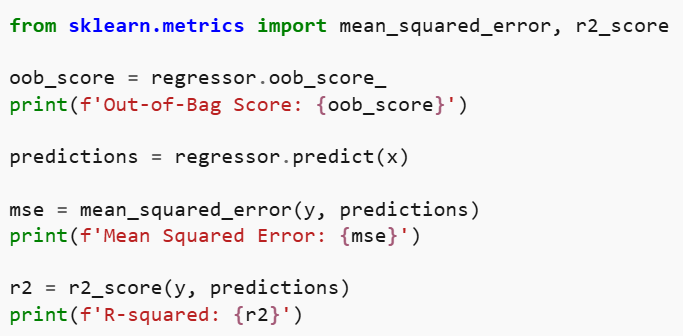
* **RandomForestRegressor:** It builds multiple decision trees and combines their predictions.
* **n\_estimators=10**: Defines the number of decision trees in the Random Forest.
* **random\_state=0:** Ensures the randomness in model training is controlled for reproducibility.
* **oob\_score=True:** Enables out-of-bag scoring which evaluates the model's performance using data not seen by individual trees during training.
* **LabelEncoder():**Converts categorical variables (object type) into numerical values, making them suitable for machine learning models.
* **apply(label\_encoder.fit\_transform):** Applies the LabelEncoder transformation to each categorical column, converting string labels into numbers.
* **concat():**Combines the numerical and encoded categorical features horizontally into one dataset which is then used as input for the model.

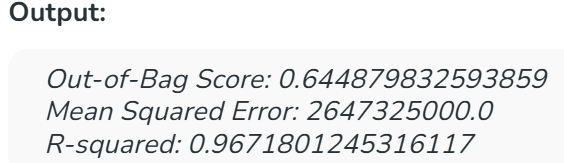


**5. Making predictions and Evaluating**

The code evaluates the trained Random Forest Regression model:

* **oob\_score\_:** Retrive out-of-bag (OOB) score which estimates the model's generalization performance.
* Makes predictions using the trained model and stores them in the 'predictions' array.
* Evaluates the model's performance using the Mean Squared Error (MSE) and R-squared (R2) metrics.

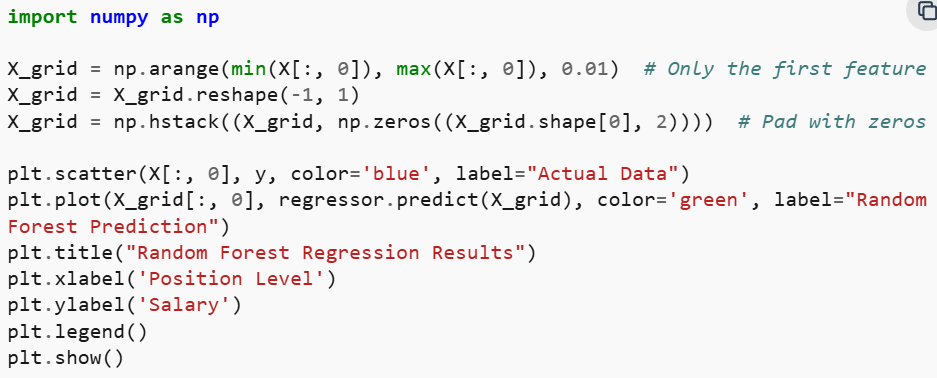


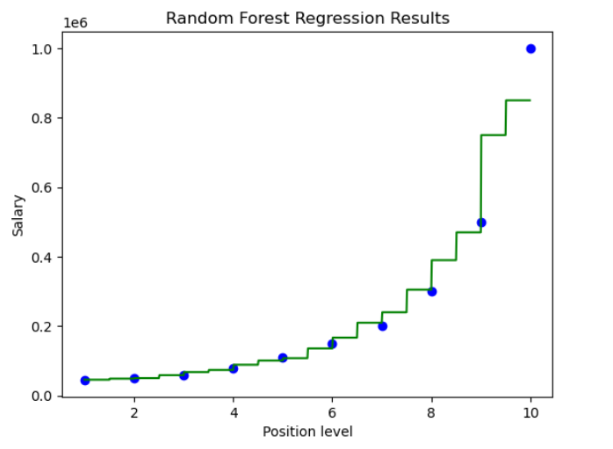


**6. Visualizing**

Now let's visualize the results obtained by using the RandomForest Regression model on our salaries dataset.

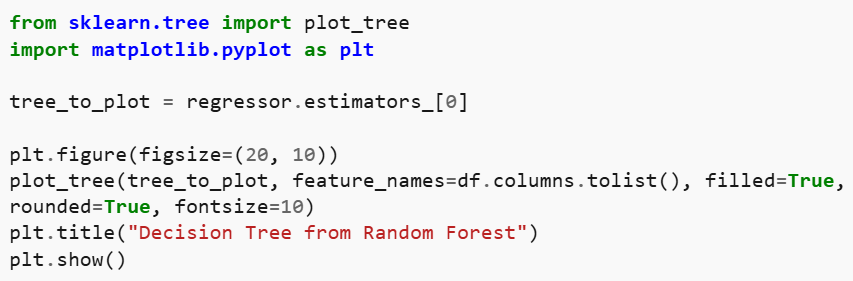
* Creates a grid of prediction points covering the range of the feature values.
* Plots the real data points as blue scatter points.
* Plots the predicted values for the prediction grid as a green line.
* Adds labels and a title to the plot for better understanding.

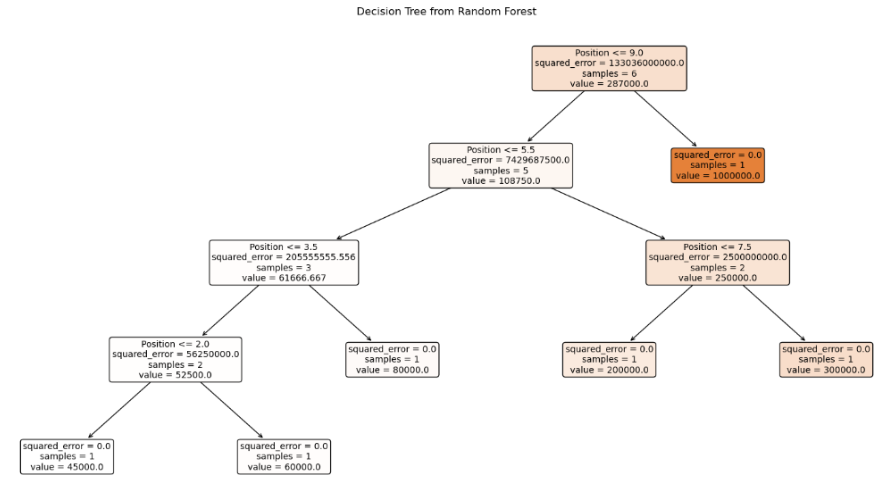




**7. Visualizing a Single Decision Tree from the Random Forest Model**

The code visualizes one of the decision trees from the trained Random Forest model. Plots the selected decision tree, displaying the decision-making process of a single tree within the ensemble.





**Applications of Random Forest Regression**

The Random Forest regression has a wide range of real-world problems including:

* **Predicting continuous numerical values:** Predicting house prices, stock prices or customer lifetime value.
* **Identifying risk factors:** Detecting risk factors for diseases, financial crises or other negative events.
* **Handling high-dimensional data:** Analyzing datasets with a large number of input features.
* **Capturing complex relationships:** Modeling complex relationships between input features and the target variable.

**Advantages of Random Forest Regression**

* **Handles Non-Linearity**: It can capture complex, non-linear relationships in the data that other models might miss.
* **Reduces Overfitting**: By combining multiple decision trees and averaging predictions it reduces the risk of overfitting compared to a single decision tree.
* **Robust to Outliers**: Random Forest is less sensitive to outliers as it aggregates the predictions from multiple trees.
* **Works Well with Large Datasets**: It can efficiently handle large datasets and high-dimensional data without a significant loss in performance.
* **Handles Missing Data**: Random Forest can handle missing values by using surrogate splits and maintaining high accuracy even with incomplete data.
* **No Need for Feature Scaling**: Unlike many other algorithms Random Forest does not require normalization or scaling of the data.

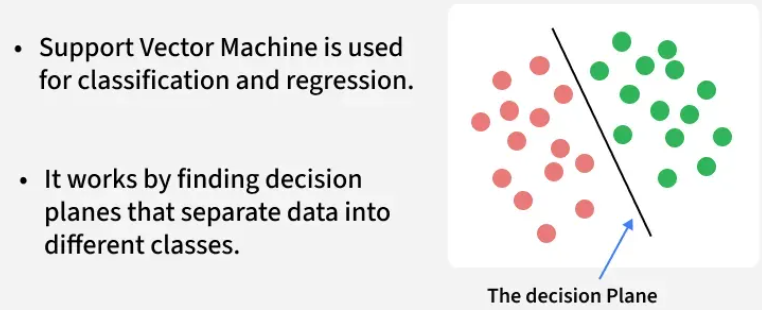
**Disadvantages of Random Forest Regression**

* **Complexity**: It can be computationally expensive and slow to train especially with a large number of trees and high-dimensional data. Due to this it may not be suitable for real-time predictions especially with a large number of trees.
* **Less Interpretability**: Since it uses many trees it can be harder to interpret compared to simpler models like linear regression or decision trees.
* **Memory Intensive**: Storing multiple decision trees for large datasets require significant memory resources.
* **Overfitting on Noisy Data**: While Random Forest reduces overfitting, it can still overfit if the data is highly noisy especially with a large number of trees.
* **Sensitive to Imbalanced Data**: It may perform poorly if the dataset is highly imbalanced like one class is significantly more frequent than another.

**Support Vector Machine (SVM) Algorithm**

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It tries to find the best boundary known as hyperplane that separates different classes in the data. It is useful when you want to do binary classification like spam vs. not spam or cat vs. dog.

The main goal of SVM is to maximize the margin between the two classes. The larger the margin the better the model performs on new and unseen data.

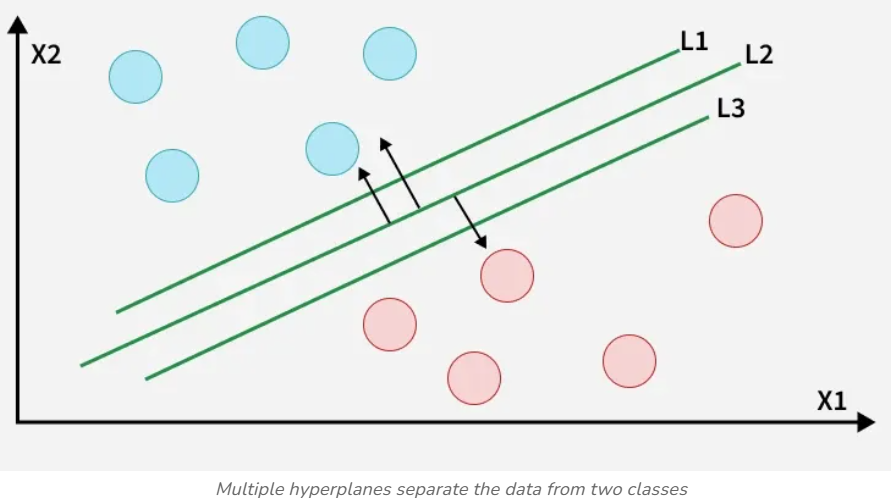


**Key Concepts of Support Vector Machine**

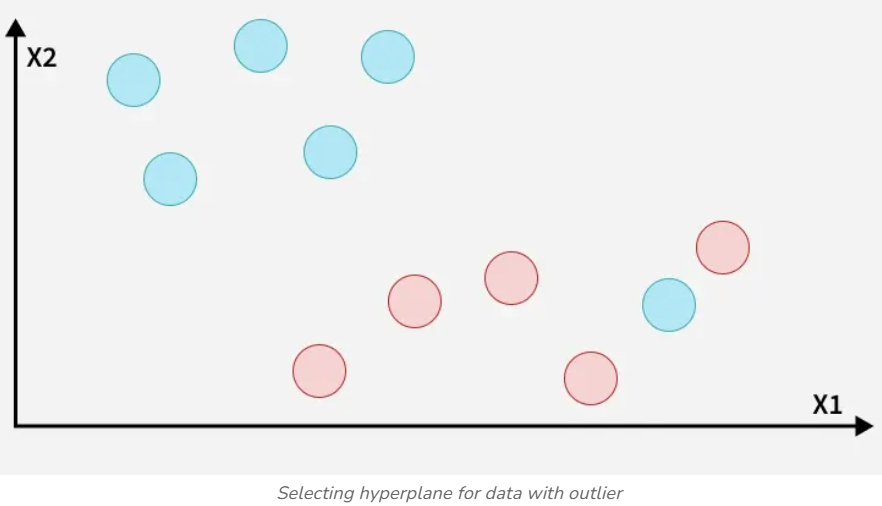
* **Hyperplane**: A decision boundary separating different classes in feature space and is represented by the equation wx + b = 0 in linear classification.
* **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
* **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
* **Kernel**: A function that maps data to a higher-dimensional space enabling SVM to handle non-linearly separable data.
* **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
* **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
* **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value forces stricter penalty for misclassifications.
* **Hinge Loss**: A loss function penalizing misclassified points or margin violations and is combined with regularization in SVM.
* **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

**How does Support Vector Machine Algorithm Work?**

The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.



The best hyperplane also known as the **"hard margin"** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So from the above figure, we choose L2 as hard margin. Let's consider a scenario like shown below:



**Types of Support Vector Machine**

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

* **Linear SVM:**Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.
* **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

**Advantages of Support Vector Machine (SVM)**

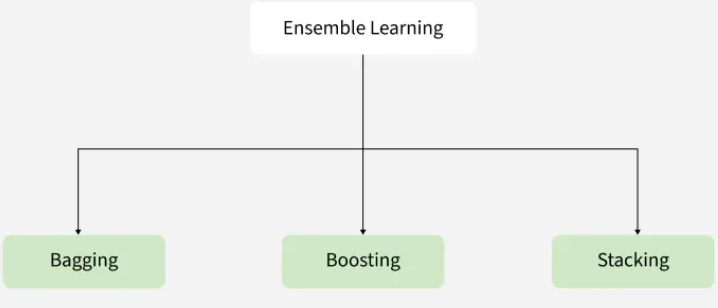
1. **High-Dimensional Performance**: SVM excels in high-dimensional spaces, making it suitable for image classification and gene expression analysis.
2. **Nonlinear Capability**: Utilizing kernel functions like RBF and polynomial SVM effectively handles nonlinear relationships.
3. **Outlier Resilience**: The soft margin feature allows SVM to ignore outliers, enhancing robustness in spam detection and anomaly detection.
4. **Binary and Multiclass Support**: SVM is effective for both binary classification and multiclass classification suitable for applications in text classification.
5. **Memory Efficiency**: It focuses on support vectors making it memory efficient compared to other algorithms.

**Disadvantages of Support Vector Machine (SVM)**

1. **Slow Training**: SVM can be slow for large datasets, affecting performance in SVM in data mining tasks.
2. **Parameter Tuning Difficulty**: Selecting the right kernel and adjusting parameters like C requires careful tuning, impacting SVM algorithms.
3. **Noise Sensitivity**: SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.
4. **Limited Interpretability**: The complexity of the hyperplane in higher dimensions makes SVM less interpretable than other models.
5. **Feature Scaling Sensitivity**: Proper feature scaling is essential, otherwise SVM models may perform poorly

**Ensemble Learning**

Ensemble learning is a method where we use many small models instead of just one. Each of these models may not be very strong on its own, but when we put their results together, we get a better and more accurate answer. It's like asking a group of people for advice instead of just one person—each one might be a little wrong, but together, they usually give a better answer.



**Types of Ensembles Learning in Machine Learning**

There are three main types of ensemble methods:

1. **Bagging (Bootstrap Aggregating):** Models are trained independently on different random subsets of the training data. Their results are then combined—usually by averaging (for regression) or voting (for classification). This helps reduce variance and prevents overfitting.
2. **Boosting:** Models are trained one after another. Each new model focuses on fixing the errors made by the previous ones. The final prediction is a weighted combination of all models, which helps reduce bias and improve accuracy.
3. **Stacking (Stacked Generalization):** Multiple different models (often of different types) are trained and their predictions are used as inputs to a final model, called a meta-model. The meta-model learns how to best combinethe predictions of the base models, aiming for better performance than any individual model.

While stacking is also a method but bagging and boosting method is widely used and lets see more about them.

**1. Bagging Algorithm**

Bagging classifier can be used for both regression and classification tasks. Here is an overview of Bagging classifier algorithm**:**

* **Bootstrap Sampling:** Divides the original training data into ‘N’ subsets and randomly selects a subset with replacement in some rows from other subsets. This step ensures that the base models are trained on diverse subsets of the data and there is no class imbalance.
* **Base Model Training:** For each bootstrapped sample we train a base model independently on that subset of data. These weak models are trained in parallel to increase computational efficiency and reduce time consumption. We can use different base learners i.e. different ML models as base learners to bring variety and robustness.
* **Prediction Aggregation:** To make a prediction on testing data combine the predictions of all base models. For classification tasks it can include majority voting or weighted majority while for regression it involves averaging the predictions.
* **Out-of-Bag (OOB) Evaluation**: Some samples are excluded from the training subset of particular base models during the bootstrapping method. These “out-of-bag” samples can be used to estimate the model’s performance without the need for cross-validation.
* **Final Prediction:** After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.

**2. Boosting Algorithm**

Boosting is an ensemble technique that combines multiple weak learners to create a strong learner. Weak models are trained in series such that each next model tries to correct errors of the previous model until the entire training dataset is predicted correctly. One of the most well-known boosting algorithms is AdaBoost (Adaptive Boosting). Here is an overview of Boosting algorithm:

* **Initialize Model Weights:** Begin with a single weak learner and assign equal weights to all training examples.
* **Train Weak Learner**: Train weak learners on these dataset.
* **Sequential Learning**: Boosting works by training models sequentially where each model focuses on correcting the errors of its predecessor. Boosting typically uses a single type of weak learner like decision trees.
* **Weight Adjustment**: Boosting assigns weights to training datapoints. Misclassified examples receive higher weights in the next iteration so that next models pay more attention to them.

**Benefits of Ensemble Learning in Machine Learning**

Ensemble learning is a versatile approach that can be applied to machine learning model for:

* **Reduction in Overfitting**: By aggregating predictions of multiple model's ensembles can reduce overfitting that individual complex models might exhibit.
* **Improved Generalization**: It generalizes better to unseen data by minimizing variance and bias.
* **Increased Accuracy**: Combining multiple models gives higher predictive accuracy.
* **Robustness to Noise**: It mitigates the effect of noisy or incorrect data points by averaging out predictions from diverse models.
* **Flexibility**: It can work with diverse models including decision trees, neural networks and support vector machines making them highly adaptable.
* **Bias-Variance Tradeoff**: Techniques like bagging reduce variance, while boosting reduces bias leading to better overall performance.

**Gradient Boosting in ML**

Gradient Boosting is a **boosting** algorithm and here each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration the algorithm computes the gradient of the loss function with respect to predictions and then trains a new weak model to minimize this gradient. Predictions of the new model are then added to the ensemble (all models prediction) and the process is repeated until a stopping criterion is met.

**Shrinkage and Model Complexity**

A key feature of Gradient Boosting is shrinkage which scales the contribution of each new model using **learning rate** (denoted as ).

* **Smaller learning rates:** mean the contribution of each tree is smaller which reduces the risk of overfitting but requires more trees to achieve the same performance.
* **Larger learning rates:** mean each tree has a more significant impact but this can lead to overfitting.

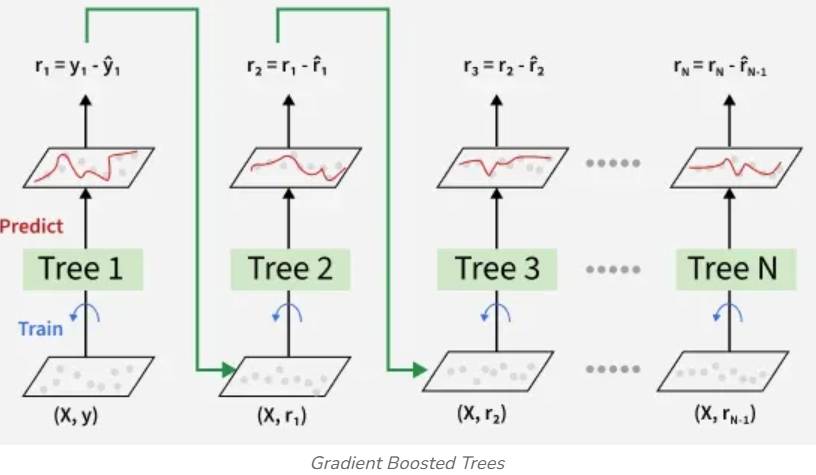
**Working of GB:**

1. **Sequential learning process:**

The ensemble consists of multiple trees each trained to correct the errors of the previous one. In the first iteration **Tree 1** is trained on the original data  and the true labels . It makes predictions which are used to compute the errors.

1. **Residuals calculation:**

In the second iteration **Tree 2** is trained using the feature matrix  and the errors from Tree 1 as labels. This means Tree 2 is trained to predict the errors of Tree 1. This process continues for all the trees in the ensemble. Each subsequent tree is trained to predict the errors of the previous tree.



**3. Shrinkage**

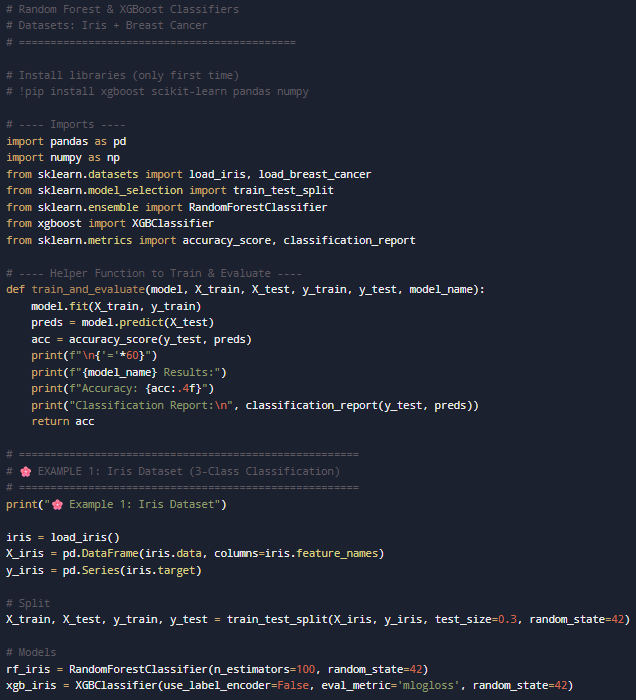
After each tree is trained its predictions are **shrunk** by multiplying them with the learning rate η which ranges from 0 to 1. This prevents overfitting by ensuring each tree has a smaller impact on the final model.

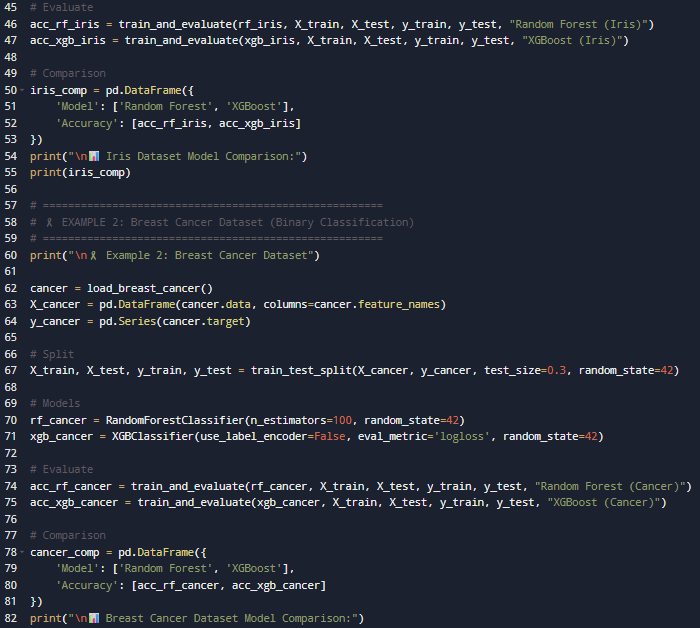
Once all trees are trained predictions are made by summing the contributions of all the trees. The final prediction is given by the formula:

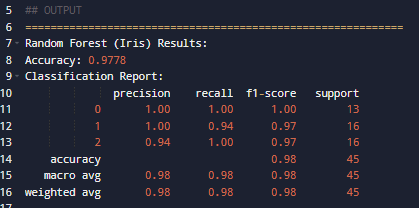


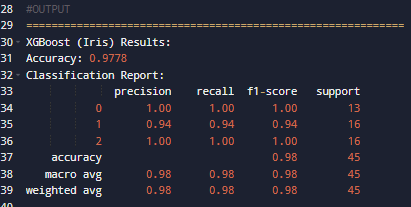
Where  are the errors predicted by each tree.

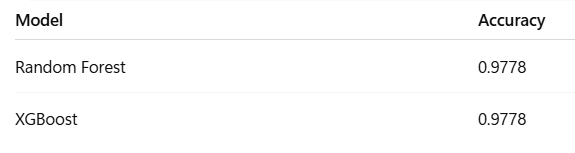
**RANDOM FOREST OR XG BOOST CLASSIFIER:**

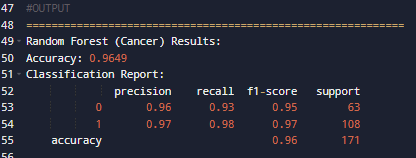


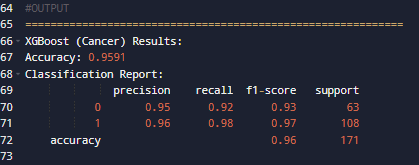


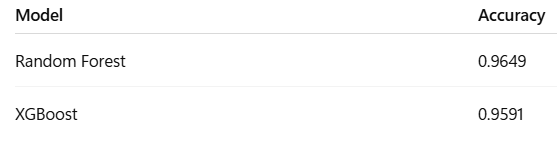




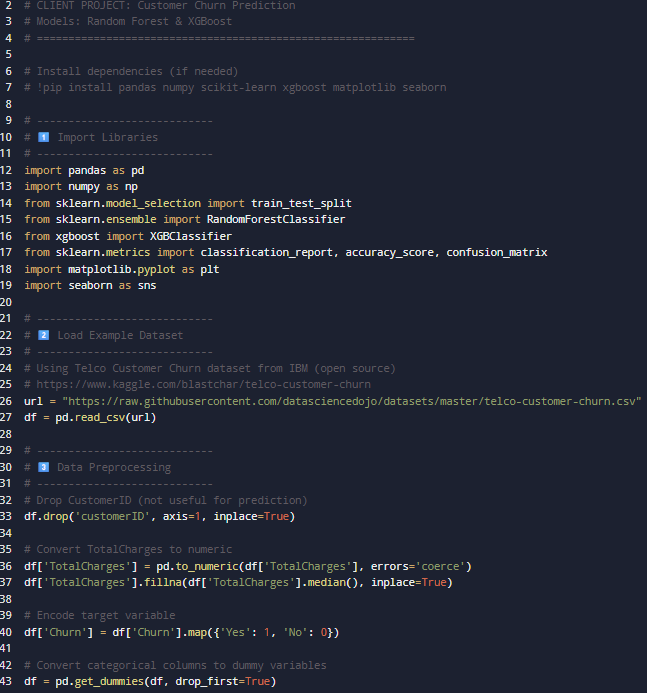


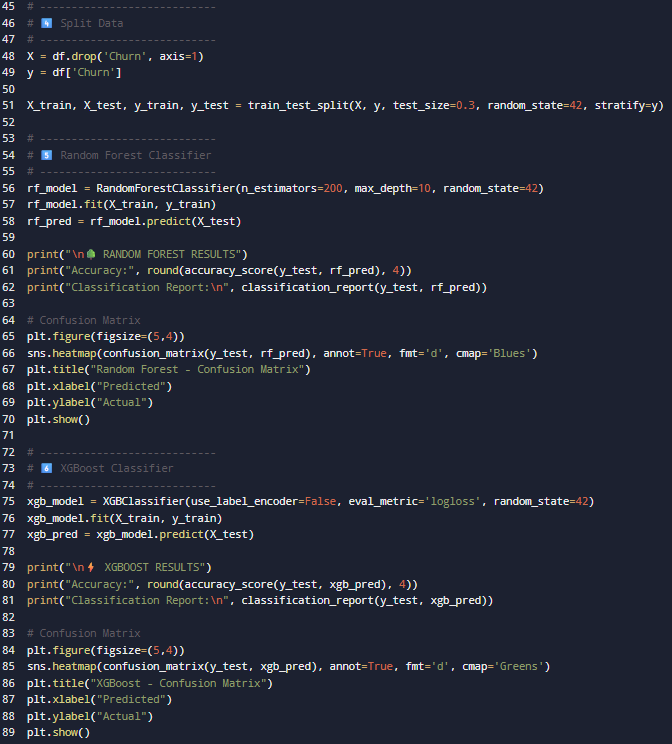


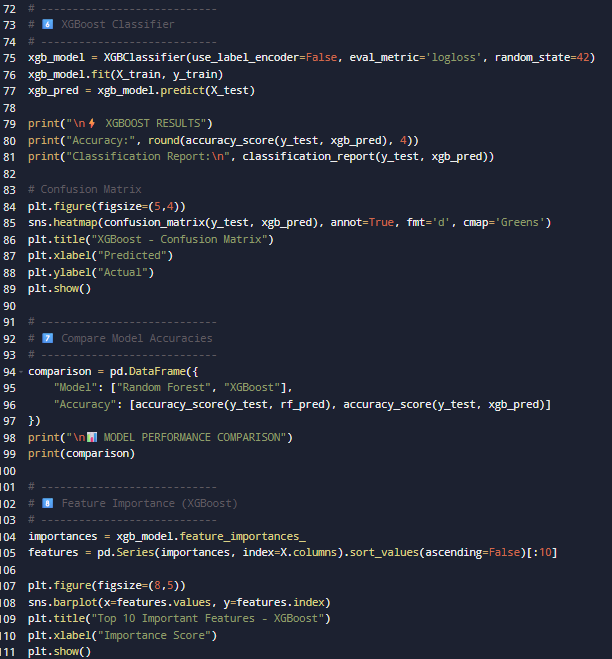




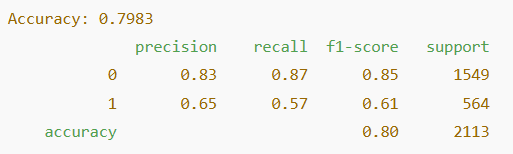
**Customer Churn Prediction using Advanced Classifiers (Random Forest & XGBoost)**

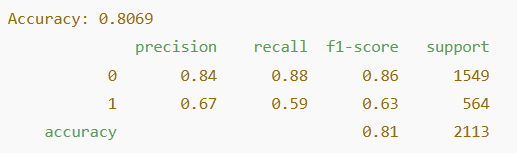
****

****

****

**OUTPUT:**

****

****

**Business Interpretation**

* **Tenure** (how long a customer stays) is the strongest factor in predicting churn.
* Customers with **fiber optic internet** and **electronic check payment** are more likely to churn.
* Offering **loyalty discounts or switching to annual contracts** can reduce churn.