**BJ: to Lab-5, PUM**

**1. Random Forest for ML**

1. **Definition of Random Forest (RF)**: RF is a popular learning method used in machine learning for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputting the mode (for classification) or mean prediction (for regression) of the individual trees. Each decision tree in the forest is trained on a subset of the training data, and at each node of the tree, a random subset of features is considered for splitting. This randomness and diversity among the trees in the forest help to reduce overfitting and improve the generalization performance of the model.
2. **Types of Problems Solved by Random Forest in ML**: Random Forest can be applied to a wide range of machine learning tasks, including:
   * ***Classification: Predicting categorical outcomes such as class labels or categories.***
   * ***Regression: Predicting continuous outcomes such as numeric values.***
   * Anomaly Detection: Identifying outliers or anomalies in the data.
   * Feature Importance: Ranking the importance of features in predicting the target variable.
   * Imbalanced Data: Handling imbalanced datasets by providing robust predictions even with unequal class distributions.
3. **Problems Solved Using Random Forest in Orange**: In Orange, RF can be used for **classification** and ***regression*** tasks. It offers a user-friendly interface for building, visualizing, and evaluating RF models. Users can utilize it to analyze datasets, make predictions, and understand the importance of features in the dataset. Additionally, Orange provides various tools for data preprocessing, feature selection, and model evaluation, which can be integrated with RF workflows.
4. **Datasets Suitable for Random Forest in Orange**:
   * **Iris Dataset**: A classic dataset often used for classification tasks. It contains measurements of iris flowers with three different species.
   * **Titanic Dataset**: A dataset containing information about passengers aboard the Titanic, used for predicting survival.
   * **Boston Housing Dataset**: A regression dataset containing information about housing prices in Boston suburbs. It can be used to predict house prices based on various features such as crime rate, number of rooms, etc.

These datasets can be effectively solved using RF in Orange, providing with practical examples of applying RF for classification and regression analysis.

**History of RF (briefly)**

The development of RF can be traced back to the 1990s, with key contributions made by Leo Breiman and Adele Cutler. Here's a brief overview of the history of RF formation and development:

1. **Decision Trees and Bagging**:
   * The concept of decision trees dates back to the 1960s and 1970s, with early work by researchers such as Ross Quinlan. Decision trees were initially developed as a simple yet powerful method for classification and regression tasks.
   * In the late 1990s, Leo Breiman introduced the idea of bagging (bootstrap aggregating), which involves training multiple models on bootstrap samples of the data and combining their predictions to reduce variance and improve generalization.
2. **Introduction of Random Forest**:
   * In 2001, Leo Breiman and Adele Cutler introduced RF as an extension of bagging. The key innovation of RF was to introduce additional randomness by selecting a random subset of features at each split in the decision tree.
   * This randomness helps to decorrelate the individual trees in the ensemble and further reduce overfitting, leading to improved performance and robustness.
3. **Theoretical Foundations**:
   * Breiman and Cutler's seminal paper "Random Forests" provided a theoretical foundation for the algorithm and demonstrated its effectiveness across a wide range of datasets and applications.
   * The paper highlighted the importance of ensemble learning and the benefits of combining multiple weak learners to create a strong learner.
4. **Popularity and Adoption**:
   * RF quickly gained popularity in both academia and industry due to its simplicity, flexibility, and superior performance compared to other machine learning algorithms.
   * It became widely used in various fields, including bioinformatics, finance, healthcare, and image recognition.
5. **Further Refinements and Variations**:
   * Over the years, researchers have proposed several refinements and variations of the RF algorithm, including improvements to the feature selection process, handling of imbalanced datasets, and scalability to large datasets.
   * Variants such as Extremely Randomized Trees (Extra-Trees), which introduce additional randomness in the tree-building process, have also been developed.
6. **Open Source Implementation**:
   * RF implementations are available in popular machine learning libraries such as scikit-learn in Python, making it easily accessible to practitioners and researchers.
   * The availability of open-source implementations has further contributed to the widespread adoption of RF in the machine learning community.

Overall, RF has become one of the go-to algorithms for classification and regression tasks, owing to its robustness, scalability, and excellent performance on a wide range of real-world datasets. Its development represents a significant milestone in the evolution of ensemble learning methods.

**The basic Algorithms of RF**

RF is a powerful ensemble learning method widely used for classification problems in machine learning. It combines the predictions from multiple decision trees to improve the overall accuracy and robustness of the model. Below are the basic algorithms and theorems for Random Forest in detail:

1. **Decision Trees**:
   * Decision trees are the building blocks of RF. They are constructed recursively by splitting the feature space into regions based on feature values.
   * The splitting process is guided by a *criterion such as* ***Gini*** *impurity or* ***entropy***, which measures the homogeneity of the target variable within each region.
   * Each leaf node in the decision tree represents a class label or a probability distribution over class labels.
2. **Bagging (Bootstrap Aggregating)**:
   * RF employs an ensemble learning technique called bagging, where multiple decision trees are trained independently on random subsets of the training data.
   * Bootstrapping is used to generate these subsets by sampling with replacement from the original training data.
   * Each tree is trained on a different subset of the data, resulting in diverse trees that capture different aspects of the underlying data distribution.
3. **Random Feature Subsets**:
   * In addition to using random subsets of the training data, Random Forest also selects a random subset of features at each node of the decision tree.
   * This randomness helps to decorrelate the trees and reduces the risk of overfitting.
   * The number of features considered at each split is typically sqrt(n\_features), where n\_features is the total number of features in the dataset.
4. **Voting or Averaging**:
   * After training the ensemble of decision trees, predictions are made by aggregating the individual predictions of each tree.
   * For classification tasks, the class label with the most votes among the trees is chosen as the final prediction.
   * For regression tasks, the average of the predicted values from all trees is taken as the final prediction.
5. **Out-of-Bag (OOB) Error Estimation**:
   * RF utilizes out-of-bag samples, which are data points that are not included in the bootstrap sample used to train each individual tree.
   * These out-of-bag samples can be used to estimate the generalization error of the model without the need for a separate validation set.
6. **Optimization Criteria**:
   * The main optimization criteria for RF include maximizing the accuracy or minimizing the misclassification rate on the training data.
   * Additionally, tuning hyperparameters such as the number of trees, maximum depth of trees, and minimum samples per leaf can help optimize the performance of the model.

Overall, Random Forests offer a robust and flexible approach to classification problems, combining the strengths of decision trees with the power of ensemble learning. Their ability to handle large datasets, high-dimensional feature spaces, and noisy data makes them a popular choice across various domains.

**1. Implemented RF Algorithms in Orange:**

Orange uses the scikit-learn library for its machine learning functionalities. While scikit-learn offers various decision tree algorithms, the Random Forest widget in Orange specifically implements the **CART (Classification And Regression Trees)** algorithm as the base learner for building the ensemble.

**2. Detailed Mathematical Algorithm of CART:**

Here's a breakdown of the CART algorithm used in Orange's RF widget:

**2.1. Training Phase:**

1. **Data Selection:** A random subset of training data (with replacement) is drawn (**bootstrapping**). This creates multiple training sets with some data points appearing in multiple sets and others left out.
2. **Feature Selection:** At each node of the tree, a random subset of features (*typically the square root of the total features*) is chosen. From this subset, the best splitting feature is selected based on a **splitting criterion**.
3. **Splitting Criterion:** The most common splitting criterion in CART is the **Gini impurity** for classification and the **variance** for regression.
   * **Gini Impurity (Classification):** It measures the probability of a randomly chosen data point at a node being misclassified if labeled based on the distribution of labels at that node. ***A lower Gini impurity indicates better separation of classes***.
   * **Variance (Regression):** It measures the average squared difference of data points from the mean value at a node. A lower variance indicates a more homogeneous group of data points for regression.
4. **Splitting:** The chosen feature is used to split the data into two child nodes based on a threshold value that minimizes the chosen splitting criterion. This process continues recursively until a stopping criterion is met (e.g., maximum depth of the tree, minimum number of data points in a node).

**2.2. Prediction Phase:**

1. A new unseen data point is passed down the tree starting from the root node.
2. At each node, the feature value of the data point is compared to the splitting threshold.
3. The data point is directed to the left or right child node based on the comparison.
4. This process continues until the data point reaches a leaf node (terminal node).
5. For classification, the data point is assigned the majority class label in the leaf node.
6. For regression, the data point is predicted by the average value of the target variable in the leaf node.

**2.3. Random Forest Ensemble:**

1. Multiple decision trees (typically hundreds or thousands) are grown independently using the steps mentioned above. Each tree uses bootstrapping and random feature selection to introduce diversity.
2. For classification, the final prediction for a new data point is made by a majority vote of the trees in the ensemble. The class with the most votes is assigned.
3. For regression, the final prediction is the average of the predictions from all trees in the ensemble.

**Key Features of CART and Random Forests:**

* **Interpretability:** Individual trees in a RF can be visualized and inspected to understand the splitting rules and feature importance.
* **Handling Missing Values:** CART can handle missing values by choosing the best split that considers both data points with and without missing values for the chosen feature.
* **Robustness to Noise:** RF are generally robust to noise in the data due to averaging predictions from multiple trees.
* **Overfitting Prevention:** Bootstrapping and random feature selection help prevent overfitting by introducing diversity in the ensemble.

**Additional Notes:**

* Orange's RF widget allows you to configure various parameters for tree building, such as the number of trees, maximum tree depth, and minimum number of data points in a leaf node.
* You can also explore other ensemble learning algorithms available in Orange's visualization and analysis workflows, although the Random Forest widget specifically uses CART as the base learner.

By understanding the CART algorithm and the ensemble approach of Random Forests, you can leverage this powerful machine learning technique effectively within the Orange data mining environment.

**3.** ***A RF Regression algorithm in Orange (using CART) doesn't directly result in linear regression concerning features of the dataset***.

1. **Non-Linear Decision Boundaries:** Unlike linear regression, which models a straight line or hyperplane for prediction, Random Forests create decision trees with a series of splits based on feature values. These splits can result in non-linear decision boundaries, allowing the model to capture more complex relationships between features and the target variable.
2. **Ensemble Averaging:** Even though individual trees might learn non-linear splits, the final prediction in a Random Forest Regression is the average of predictions from multiple trees. This averaging can sometimes lead to a smoother, less wiggly prediction surface that might resemble a linear relationship, but it's not strictly linear regression.

**Approximating Polynomial Regression with Random Forests:**

While Orange's Random Forest widget doesn't directly implement polynomial regression, there are techniques to achieve a similar effect:

1. **Feature Engineering:** You can create new features by transforming existing features using polynomial terms (e.g., squares, cubes, interactions between features). Feeding these new features along with the original ones into the Random Forest allows it to capture polynomial relationships implicitly by learning decision boundaries based on the transformed features.
2. **Stacked Ensemble with Polynomial Regression:** You could build a two-stage ensemble. In the first stage, train a Random Forest as usual. In the second stage, use the predictions from the Random Forest as new features to train a separate linear or polynomial regression model. This approach leverages the strengths of both techniques – Random Forest for capturing complex interactions and polynomial regression for modeling smoother, potentially curved relationships.

**Alternatives for Polynomial Regression in Orange:**

Orange offers other tools for polynomial regression directly:

* **Linear Regression with Polynomial Features:** You can use the Linear Regression widget with the "Polynomial Features" option enabled. This will automatically create polynomial terms from your existing features and perform linear regression on the transformed dataset.
* **Support Vector Regression (SVR) with Polynomial Kernel:** The SVR widget allows selecting a polynomial kernel function. This kernel implicitly maps the data into a higher-dimensional space, enabling SVR to learn non-linear relationships with a polynomial flavor.

By understanding the capabilities and limitations of Random Forests, you can choose the most appropriate technique for your regression task in Orange. If you need a strictly polynomial model, consider the alternatives mentioned above.

**4. RF Widget in Orange**

**Detailed Explanation of RF Widget Parameters in Orange**

**1. Number of Trees (30-50 for Bone Marrow dataset /=?**

* **Description:** This parameter specifies how many decision trees will be built in the Random Forest ensemble.
* **Explanation:** More trees generally lead to a more robust and accurate model, but also increase training time and computational cost.
* **Choosing a Value:**
  + A good starting point for the number of trees is typically between 100 and 1000.
  + For smaller datasets (less than 1000 data points), you might consider fewer trees (e.g., 30-50).
  + The optimal number of trees can be determined through experimentation using techniques like cross-validation. Evaluating model performance on unseen data can help you identify the number of trees that maximizes accuracy while avoiding overfitting.

**2. Number of Attributes Considered at Each Split (30-40 /?)**

* **Description:** This parameter controls how many features (attributes) are randomly chosen as candidates for splitting at each node in a decision tree within the forest.
* **Explanation:**
  + A lower number focuses the search on a smaller subset of features at each split, potentially leading to more focused trees but potentially missing important features.
  + A higher number allows for exploring more features at each split, potentially leading to more diverse trees but also increasing the risk of overfitting to noise in the data.
* **Choosing a Value:**
  + *A common rule of thumb is to choose the square root of the total number of features.*
  + Experimentation with different values can help find the sweet spot for your dataset.

**3. Options:**

* **o1: Replicable Training:**
  + **Description:** When checked, this option ensures that the Random Forest model generates the same results when you run it multiple times with the same data. This is useful for debugging or comparing results across different runs.
  + **Recommendation:** For most cases, keeping "Replicable Training" checked is recommended, especially for educational purposes. It allows students to focus on understanding the model's behavior without worrying about random variations in results.
* **o2: Balance Class Distribution:**
  + **Description:** When checked and applied to classification tasks, this option attempts to balance the class distribution within each tree by oversampling the minority class(es) or undersampling the majority class. This can be helpful in datasets with imbalanced class distributions.
  + **Recommendation:** Consider using "Balance Class Distribution" if your dataset has significantly imbalanced classes and class imbalance affects the model's performance. Otherwise, it might be unnecessary.

**4. Growth Control:**

* **Choice 1: Limit Depth of Individual Tree (3-5…. 8-10?)**
* **Description:** This option sets a maximum depth for each tree in the forest. Depth refers to the number of splits (decision nodes) from the root node to a leaf node (terminal node).
* **Explanation:**
  + A deeper tree can capture more complex relationships between features, but also increases the risk of overfitting.
  + Shallower trees are less prone to overfitting but might not capture complex patterns.
* **Choosing a Value:**
  + Start with a moderate depth (e.g., 3-5) and experiment with increasing it if the model underfits (performs poorly on unseen data).
  + You can use techniques like cross-validation to find the optimal depth that balances accuracy and generalization.
* **Choice 2: Do Not Split Subset Smaller Than (5-8/..)**
* **Description:** This option sets a minimum number of data points allowed in a leaf node.
* **Explanation:**
  + If a split results in very small leaf nodes with few data points, it can lead to overfitting.
  + Setting a minimum number of data points per leaf node helps prevent this.
* **Choosing a Value:**
  + A common starting point is 5-10 data points per leaf node. Experiment with different values to see how it affects model performance.

**Key Takeaways:**

* The Random Forest widget offers several parameters that allow you to fine-tune the model's behavior.
* Start with reasonable default values (e.g., number of trees) and experiment to see how adjustments affect model performance.
* Techniques like cross-validation can help you find the best solution(s)

**Refs**: Adele Cutler, D. Richard Cutler and John R. Stevens*,* ***Random Forests****.* Chapter in Machine Learning*. 2011 (21p).*