Financial Recommendation System with Regression Trees With Multiple Input Criteria

By:

Swathi Rao

Vaibhavi Mahindrakar

Ishita Binwal

Aishwarya Rajkumar

Abstract:

According to a study, only 27% of Indians are financially literate. Without proper financial knowledge, people tend to make poor financial decisions- such as taking unnecessary risks and falling into debts. This also has an effect on the larger scale throughout the country, leading to low capital rates and inhibiting entrepreneurship.

Our project aims to solve that problem- by giving users personalized financial recommendations based on demographic data. It also keeps in mind the risk appetite of the user and compares it with their income to prevent users from taking unnecessary risks, thus giving them a comprehensive breakdown of where to invest their savings.

Related Work:

1. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* **9**, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5 - Provides a comprehensive research-based perspective on recommendation systems, their uses and the types of recommender systems available.

2. Song YY, Lu Y. Decision tree methods: applications for classification and prediction. Shanghai Arch Psychiatry. 2015 Apr 25;27(2):130-5. doi: 10.11919/j.issn.1002-0829.215044. PMID: 26120265; PMCID: PMC4466856. – talks about decision trees and their uses in various settings. Also provides an introduction to the types of decision trees for classification and regression tasks.

3. Harvard Business Review- Decision Trees for Decision Making

Architecture:

1. Data Loading and Preprocessing:

- Before constructing decision trees, load the investment dataset. This dataset contains historical information about investments, such as age, monthly income, occupation, family size, and risk profile.
- Data preprocessing involves cleaning the data, which may include dealing with missing values and encoding categorical variables. Preprocessing ensures that the data is suitable for training decision trees.

2. Decision Tree Models:

- Decision trees are a type of machine learning model used for both classification and regression tasks. In our application, we are using them for regression, which means they predict continuous numerical values (investment percentages).
- For each target attribute, we construct a separate decision tree.

3. Regression Tree Node:

- Each decision tree is composed of nodes, and they have defined a `RegressionNode` class. A node can have children nodes, a feature, a split value, and a prediction value.
- When a new node is created, it initially has no children, and the `feature`, `split_value`, and `prediction` attributes are undefined.

4. Splitting Criterion and Tree Construction for Regression:

- Decision tree construction, the mean squared error (MSE) is used as a splitting criterion. MSE measures the variance of target values. Lower MSE implies a better split.
- Search for the best split by iterating through attributes (features) and their unique values. The split that minimizes the MSE is selected.
- Once the best split is found, the node's `feature` and `split_value` attributes are set, and the data is divided into left and right subsets.

5. Recursive Tree Building:

- Decision tree construction is a recursive process. Keep splitting the data into subsets based on the selected feature and split value.
- New nodes are created for these subsets, and the process continues until stopping conditions are met. For example, if all examples in a subset belong to the same class or if a predefined depth is reached.

6. Prediction Function for Regression:

- When making predictions, the decision tree traverses the tree from the root node to a leaf node.
- If the current node has a `prediction` value, it returns this prediction. If not, it checks the feature value of the input example and follows the appropriate child node (left or right) based on the comparison with the `split_value`.
- The traversal continues until a leaf node with a prediction is reached.

7. Building Decision Trees for Different Target Attributes:

- In this phase, the construction of decision trees is tailored for specific target attributes, such as "InvestAmt_Percent" and "Insurance_Percent." Each attribute represents a distinct investment category.
- The construction process is replicated for each target attribute, and the decision tree is created based on the characteristics and patterns specific to the dataset associated with that attribute.
- This ensures that the predictive models are finely tuned to provide accurate predictions for each investment category, taking into account the unique factors that influence those categories. The same rigorous approach is applied independently for each target attribute, allowing the application to offer precise investment recommendations across a range of distinct investment types.

8. Application of Decision Trees:

- Once the decision trees are built, they serve as prediction models for investment application.
- When a user inputs values for age, monthly income, occupation, family size, and risk profile, these values are used to create a new_example dictionary.
- The decision tree for the target attribute (e.g., "Equity_Percent") is then used to predict the investment percentage for that category.

9. Display Predictions:

- The predicted values for each investment category are displayed in GUI through labels.
- After prediction, these labels are updated with the predicted values, making them visible to the user.

10. User Interaction:

- Users can interact with the GUI by providing input data and clicking the "Predict" button.
- The application ensures that user inputs are valid and within expected ranges to prevent errors and provide a smooth user experience.

Multiple Criteria Decision-Making Procedure:

We make use of a Decision Tree to give the user investment predictions. We make use of a Regression Tree as it is better with numerical values and it performs automatic feature interaction i.e., it can capture features between the data better without requiring any explicit programming. Regression trees are also scalable and can work efficiently on large datasets.

The program implements a decision tree-based regression model for predicting different investment percentages based on various input features. The decision tree is constructed using a recursive algorithm that selects the best feature and split value at each node to minimize the mean squared error (MSE) of the predicted values. The tree-building process continues until a stopping criterion is met, such as reaching a specified depth or having no further reduction in MSE.

The architecture of the decision tree is represented by a 'Regression Node' class, where each node contains information about its children, the selected feature, split value, and the predicted value for regression. The splitting criterion is based on the Mean Squared Error and the tree-building process is applied separately for different target attributes (Investment percentage, Equity percentage, Mutual Fund percentage, Debt percentage, Gold and Silver Percentage, Real Estate Percentage and Cryptocurrency Percentage).

The program also includes a user interaction loop where users can input values for various features - the model provides predictions for each target attribute based on the constructed regression trees. This approach allows users to obtain investment percentage predictions tailored to their specific input values.

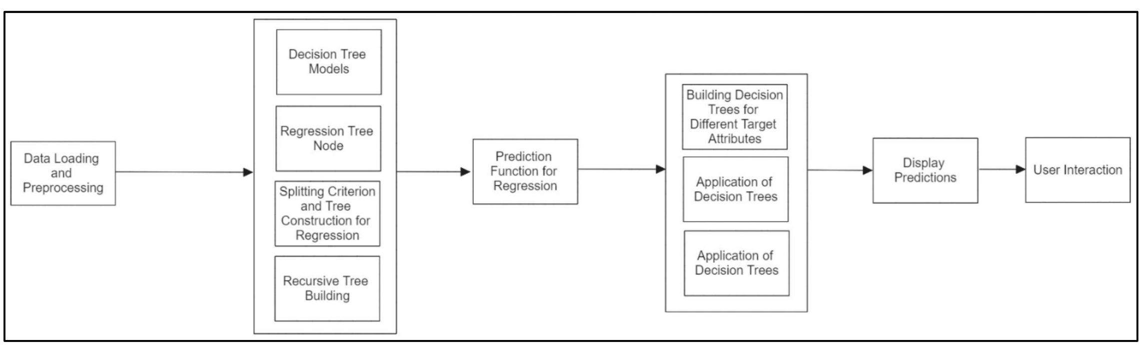
Pseudocode:

```
# Define the RegressionNode class
class RegressionNode:
    feature = ""
    split_value = None
    prediction = None
    children = []

# Define the mean squared error loss function
def mse_loss(examples):
    actual_values = examples[target_attribute]
    mean_value = actual_values.mean()
    mse = ((actual_values - mean_value) ** 2).mean()
    return mse

# Define the function to find the best split
def best_split(examples, attributes):
    best_mse = infinity
    best_feature = None
    best_split_value = None

    for feature in attributes:
        unique_values = examples[feature].unique()
        for value in unique_values:
            left_subset = examples[examples[feature] <= value]
            right_subset = examples[examples[feature] > value]
            mse = mse_loss(left_subset) + mse_loss(right_subset)

            if mse < best_mse:
```

```
            best_mse = mse
            best_feature = feature
            best_split_value = value

    return best_feature, best_split_value

# Define the recursive function to build the regression tree
def build_regression_tree(examples, attributes):
    node = RegressionNode()

    if len(attributes) == 0 or mse_loss(examples) == 0.0:
        node.prediction = examples[target_attribute].mean()
        return node

    best_feature, best_split_value = best_split(examples, attributes)
    node.feature = best_feature
    node.split_value = best_split_value

    left_subset = examples[examples[best_feature] <= best_split_value]
    right_subset = examples[examples[best_feature] > best_split_value]

    if len(left_subset) > 0:
        node.children.append(build_regression_tree(left_subset, attributes))
    if len(right_subset) > 0:
        node.children.append(build_regression_tree(right_subset, attributes))

    return node

# Define the prediction function
def predict_regression(node, example):
```

```
    if node.prediction is not None:
        return node.prediction
    if example[node.feature] <= node.split_value:
        return predict_regression(node.children[0], example)
    else:
        return predict_regression(node.children[1], example)

# Example usage:
# Build the regression tree for a target attribute
target_attribute = "InvestAmt_Percent"
root = build_regression_tree(data, features)

# Make predictions
new_example = {"Age": ..., "MonthlyIncome": ..., "Occupation": ..., "FamilySize": ..., "RiskProfile": ...}
prediction = predict_regression(root, new_example)
```

Block Diagram Showing the Program Workflow:



Implementation:

Implementing a Financial Recommendation System using Multiple Criteria Decision Making (MCDM) and Decision Trees involves several steps, including data collection, preprocessing, model selection, training, and deployment. Here's a high-level outline of how we implemented this system:

1.Data Collection and Preprocessing:
a. Collect demographic data from users, including age, income, savings, liabilities, risk tolerance, etc.
b. Gather financial data such as investment options, their historical performance, risk factors, tax implications, etc.
c. Clean and preprocess the data, handling missing values, outliers, and any inconsistencies.

2.Risk Profiling:
a. Utilize the collected demographic data to assess the risk tolerance of each user.
b. Categorize users based on their risk profiles, such as conservative, moderate, or aggressive investors.

3.Decision Tree Model:
a. Develop a decision tree model that takes the pre-processed data as input.
b. Train the decision tree to predict suitable investment options based on user profiles and risk tolerance.
c. Use techniques like Mean Squared Error.

4.Multiple Criteria Decision Making (MCDM):
a. Define the criteria for evaluating investment options, considering factors like potential returns, risk, liquidity, tax implications, etc.

5.Personalized Financial Recommendations:
a. Combine the output from the decision tree model generate personalized investment recommendations for each user.
b. Consider the user's risk profile, income, and financial goals to tailor the recommendations.

6.Real-Time Data Analytics (Future Scope):
a. Integrate Spark or similar real-time data analytics frameworks to provide up-to-date investment insights and recommendations.
b. Continuously update and adapt the model based on real-time financial data and market trends.

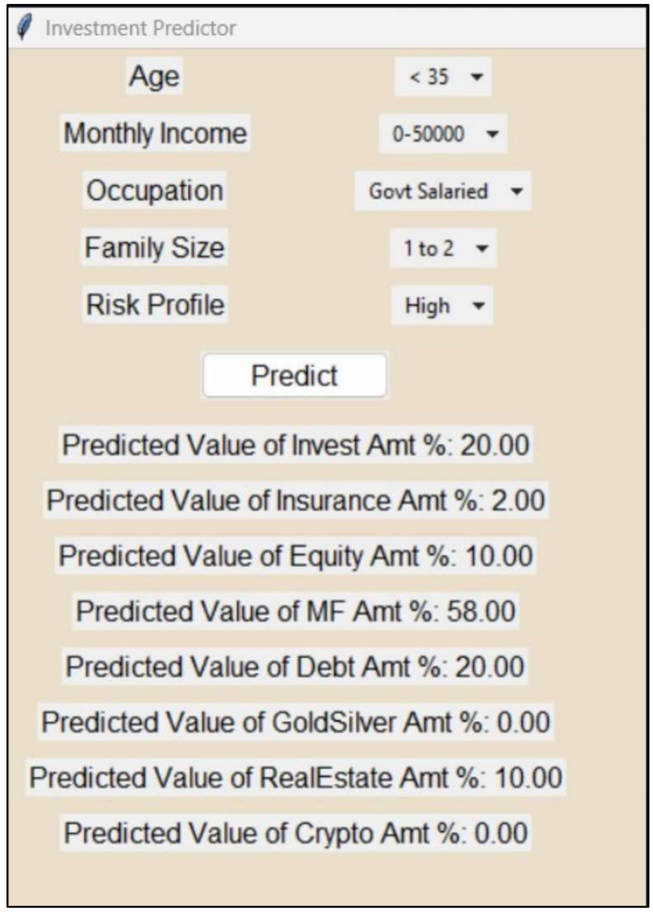
7.Enhancements for Future Scope:
a. Develop algorithms to provide a highly personalized plan for each user, considering their unique circumstances and preferences.
b. Implement a system to drill down into specific investment categories (e.g., debts, bonds, NPS, government securities) and provide recommendations within those categories.
c. Integrate tax planning services to optimize investments for tax efficiency and overall financial growth.

8.Deployment:
a. Create a user interface or platform where users can input their demographic and financial information to receive personalized recommendations.
b. Deploy the system securely on a cloud or server, ensuring data privacy and encryption.

Performance Analysis:

The dataset for this project has been specially curated and follows the generally accepted suggestions of most financial advisors. However, investments by individuals is a highly personal choice so there will always be a percentage of people who will take decisions not dictated by the standard norms. Besides this, a lot of financial decisions depend on new, unpredictable parameters known as exogenous variables (Geopolitical situations, acts of good, economic cycles etc.) and the model will have to be adjusted accordingly to suit these parameters.

Photos of the project:



Conclusion:

This project was a wonderful learning experience that gave us multidisciplinary exposure to finance and technology. Through this project, we learnt about decision trees and came to appreciate its versatility and how it can be adapted to make decisions based on multiple criteria. We also gained valuable exposure to machine learning packages in python and flask, which allows us to deploy our project as a website. We gained valuable insights about various investment avenues while curating our data. We also learnt about project scheduling, teamwork and incorporating feedback into our project.

This project introduces technology and serves as a valuable supplement in one's decision-making process for personal finance investments, especially for beginners and people who are less financially-savvy. It also serves as a check to correlate with other advisors.

In conclusion, this project is a flexible model capable of accepting new parameters- financial products, financial criteria and investment profiles. We learnt a lot through this project and would like to thank IEEE CS Society Bengaluru Section and Bhagya ma'am, our mentor for the support and guidance.