Mathematical Foundations of the Pipeline Script

In this project, I use a RandomForestClassifier to predict loan defaults. Here I dive into the mathematics and steps involved, including all relevant variables, their uses, weights, and the logic behind the calculations.

1. Data Preprocessing

The first step is to preprocess the data to make it suitable for the RandomForest model.

Mapping Variables to Numerical Values:

1. **emp_length**: This variable represents the number of years the borrower has been employed. We map categorical employment lengths to numerical values:

$$\text{emp\_map} = \{\text{'}< 1 \text{ year'} : 0, \text{'1 year'} : 1, \text{'2 years'} : 2, \ldots, \text{'10+ years'} : 10\}$$

2. **grade**: The loan grade, originally categorical, is mapped to numerical values:

$$\text{grade\_map} = \{\text{'A'} : 1, \text{'B'} : 2, \ldots, \text{'G'} : 7\}$$

3. **term**: The loan term (e.g., '36 months') is converted to a numerical value:

$$\text{term} = \text{int}(\text{term}[1:3])$$

4. **home_ownership**: The ownership status of the borrower's home, mapped to numerical values:

$$\text{home\_map} = \{\text{'MORTGAGE'} : 1, \text{'RENT'} : 2, \ldots, \text{'NONE'} : 6\}$$

Binary Encoding of Loan Status:

loan_status: This categorical variable is converted to a binary variable where

'Fully Paid' = 0 and 'Charged Off' = 1:

$$\text{binary} = \{\text{"Fully Paid"} : 0, \text{"Charged Off"} : 1\}$$

so...
data['defaulted'] = data['loan_status'].map(binary)

2. Data Splitting

The data is split into training and testing sets using train_test_split.

Training Set $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$: 80% of the data used to train the model.

Testing Set $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$: 20% of the data used to evaluate the model.

3. RandomForest Model Training

A RandomForest is used as an ensemble of decision trees where each tree gets trained on a random subset of features and samples. The final prediction is an average (for regression) or majority vote (for classification) of all trees.

RandomForest Parameters:

n_estimators: Number of trees in the forest. In this case, $n_{\text{estimators}} = 100$.
max_depth: Maximum depth of each tree. In this case, $\max_{\text{depth}} = 10$.
max_features: Number of features to consider for the best split. Here, it's the square root of the total number of features (sqrt(d)).
bootstrap: Whether bootstrap samples are used when building trees. Here, it's True.

Training the Model:

$$\text{model.fit}(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$$

4. Feature Importance and Selection

The importance of each feature is calculated based on how much it improves the split quality for all of the trees.

Gini Impurity:

$$G = 1 - \sum_{i=1}^{c} p_i^2$$

where p_i is the probability of class i at a given node, and c is the number of classes.

Feature Importance Calculation:

$$\text{Feature Importance} = \frac{1}{T} \sum_{t=1}^{T} \sum_{\text{nodes}} \frac{N_{\text{node}}}{N_{\text{total}}} \Delta G_{\text{node}}$$

Where: T is the number of trees, N_{node} is the number of samples reaching the node, and ΔG_{node} is the decrease in Gini impurity due to the split.

Weighted Feature Importance:

Certain features are given additional weights to emphasize their importance:

$$\text{weight\_factors} = \{\text{'dti'} : 5, \text{'annual\_inc'} : 3, \text{'loan\_amnt'} : 1\}$$

$$\text{feature\_importance}[i] * = \text{weight\_factors}[\text{feature\_names}[i]]$$

5. Model Evaluation

The model is evaluated on the test set using accuracy, precision, and recall metrics.

Accuracy:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Mathematical Foundations of the Loan Approval Script

Here, I use a pre-trained RandomForestClassifier model (from above) to predict loan approvals (approval/denial), assign credit grades (can), and generate personalized summaries using OpenAI's GPT-4. Let's dive into the mathematics and steps involved, making sure to include all relevant variables, their uses, weights, and the logic behind the calculations.

1. Data Preprocessing

(same as above)

2. Prediction

I load the pre-trained model and top features from a pickle file. The prediction involves using the model to predict probabilities and converting them to binary predictions with a threshold of 0.5.

Loading Model and Features:

$$\mathbf{P} = \text{model.predict\_proba}(\mathbf{X}_{\text{test}})$$

where **P** is the probability matrix with shape (n_{samples}, n_{classes}).

Predicting Probabilities:

$$\mathbf{y}_{\text{pred}} = \begin{cases} 1 & \text{if } \mathbf{P}[:, 1] \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$