**Week 8 - Develop Third Modeling Approach**

**Maria Alice Fagundes Vieira**

**Aritra Ray**

- **The Model Approach**

This week we chose a Random Forest Multilabel Framework for our recommendation system model. The model chosen can be beneficial for several reasons, including its capacity of handling multiple labels, which is extremely important for us as our target outcome is composed of 24 columns. It can also handle non-linear relationships between features and target variables. RF is also a model that is generally robust to outliers and noisy data, and it is less prone to overfitting compared to single decision trees.

- **Complexity of the modeling approach**

Regarding the complexity of the Random Forest model, it consists of multiple decision trees, each with its own complexity and the complexity increases with the number of trees (n_estimators) and the depth of each tree (max_depth), which is two of our parameters. In addition to that, we also had another parameter that we tuned, which can increase the overall complexity of the modeling process. Our Random Forest model has moderate to high complexity, especially considering the multi-label approach and the range of hyperparameters used. The model's complexity increases from Variation 1 to Variation 3, potentially offering better performance but at the cost of increased computational requirements.

- **Hyperparameters Evaluated**

This week we evaluated 3 different parameters: n_estimators, max_depth, and min_samples_leaf.

- **n_estimators**: specifies the number of trees in the forest. More trees generally lead to better performance because they reduce variance by averaging predictions across multiple models. However, increasing the number of trees also increases computational cost and memory usage.
- **max_depth**: controls the maximum depth of each tree in the forest. A deeper tree can capture more complex patterns in the data but may lead to overfitting.

- **min_samples_leaf**: sets the minimum number of samples required to be at a leaf node. Higher values prevent trees from learning overly specific patterns that are not generalizable. It acts as a regularization parameter by ensuring that leaves have enough samples to make reliable predictions.
- **random_state**: ensures reproducibility by setting a seed for the random number generator used during model training. It allows us to obtain consistent results across different runs.

```
hyperparameter_variations = [
    {'n_estimators': 50, 'max_depth': 10, 'min_samples_leaf': 5, 'random_state':17},
    {'n_estimators': 100, 'max_depth': 15, 'min_samples_leaf': 3, 'random_state':17},
    {'n_estimators': 150, 'max_depth': 20, 'min_samples_leaf': 2, 'random_state':17}
]
```

- **Model Performance Metrics**

We kept the same metrics as before during this week: ROC AUC, F1 Score and Confusion Matrix

- ROC AUC: Suitable for binary classification tasks like product recommendations. It measures the model's ability to distinguish between classes across various thresholds, which is crucial in a recommendation system where you might want to adjust the threshold for different products.
- F1 Score: This balances precision and recall, which is important in a recommendation system where you want to minimize both false positives and false negatives.
- Confusion Matrix: This provides a detailed breakdown of true positives, false positives, true negatives, and false negatives. In a product recommendation context, this can help us understand which products are being over or under recommended.

- **Training and Validation Metrics Across All 3 Variations**

**Variation 1:** Shows a Train ROC AUC of 0.5500 and a Val ROC AUC of 0.5417. The F1 Scores are relatively low, with 0.1384 for training and 0.1124 for validation. Accuracy was high, with 0.9596 for training and 0.9719 for validation. Variation 1 showed the worst performance between all 3 variations, but still shows a good balance between training and validation performance.

**Variation 2:** Train ROC AUC of 0.5658 and Val ROC AUC of 0.5505, and F1 Scores were: train: 0.1849, Val: 0.1380. Accuracy was also high with 0.9610 for train and 0.9719 for validation. Variation 2 had the second best performance between all 3 variations.

**Variation 3:** Has the highest Train ROC AUC at 0.5914 and Val ROC AUC of 0.5548. It also had the highest F1 Scores, Train: 0.2607, Val: 0.1525. Accuracy was 0.9647 for train and 0.9717 for validation showing it is the best variation for the week. The loss between train and validation predictions are low and the model performs well in unseen data.

| Variation | Train ROC AUC | Val ROC AUC | Train F1 Score | Val F1 Score | Train Accuracy | Val Accuracy |
|---|---|---|---|---|---|---|
| Variation 1 | 0.550031 | 0.541662 | 0.138405 | 0.112408 | 0.959554 | 0.971892 |
| Variation 2 | 0.56575 | 0.550482 | 0.184944 | 0.138092 | 0.961033 | 0.971856 |
| Variation 3 | 0.591426 | 0.554772 | 0.260752 | 0.152463 | 0.964707 | 0.971651 |

- **Identify the Best Model For the Week**

The best model for this week is Variation 3, since it performs better in terms of validation performance, achieving the highest scores in both F1 Score and ROC AUC, indicating better fit an generalization to unseen data. This model shows good performance across all metrics and demonstrates a good balance between training and validation scores, indicating it generalizes well to unseen data.

Overall, we see this week that we had a small loss between training and validation dataset predictions which shows a good generalization of the model.