Title: ML Algorithm Implementation Details

Course: Intro to AI

Cohort C

Date: April 30, 2025

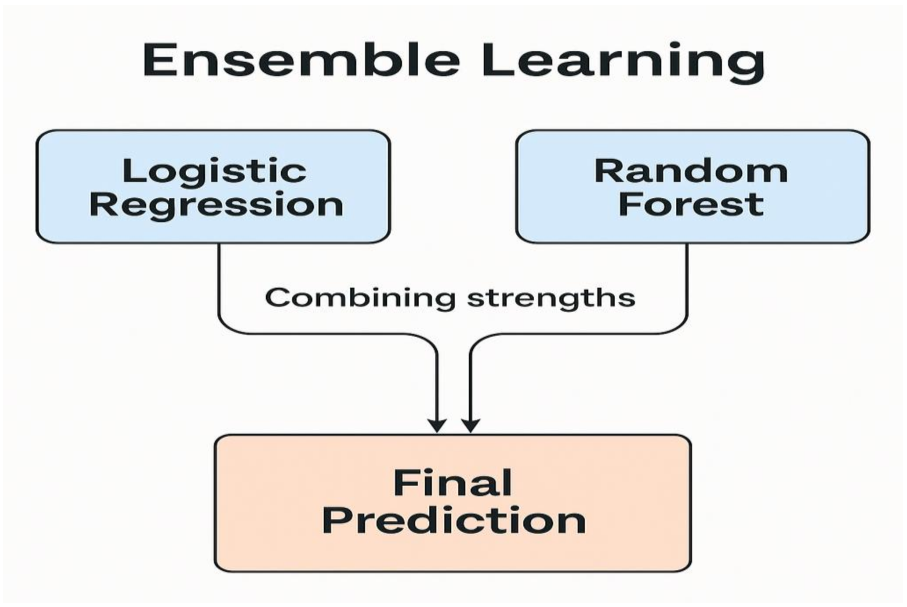
Lecturer: Owusu Asamoah Dennis

Group 5 Members:

1. Matthew Tuurozeeng
2. Benjamin Mutie Charles
3. Alan Kofi Safo Ofori
4. Hassan Maltiti Yakubu

Chosen Algorithm: Ensemble Learning (Voting Classifier)

To create our scholarship eligibility prediction model, we used an ensemble learning approach, specifically, a Voting Classifier that combines predictions from two different models: Random Forest and Logistic Regression (with cross-validation). Ensemble learning is an excellent choice because it aggregates the strengths of other algorithms, reducing overfitting, variance, and bias in making predictions on future unseen data.



The Voting Classifier operates using hard voting, meaning each model predicts a class label (eligible or not eligible), and the class receiving the majority vote becomes the final prediction. This approach is practical when base models complement each other, as we did in our case.

Implementation Approach

1. Model Selection:

    - Random Forest: A robust ensemble of decision trees that can effectively capture nonlinear relationships and handle feature importance.
    - LogisticRegressionCV: A variant of logistic regression that automatically tunes the regularisation strength using cross-validation, making it efficient and stable for binary classification tasks like ours (eligible of not eligible).

2. Data Preparation:

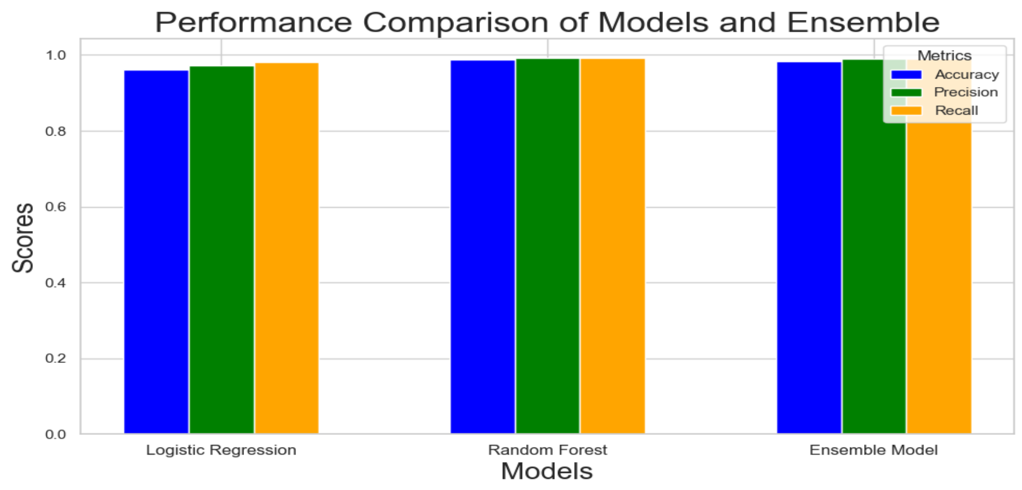
- We generated a synthetic yet realistic and balanced dataset with 20,000 student records to avoid class imbalance between eligible and not eligible cases.
- Null and duplicate values were checked for and ensured all values were in place
- We dropped columns such as ‘sat_score’ and ‘admitted’ from the data that we thought were irrelevant and therefore should not contribute to a student being eligible for a scholarship at Ashesi. We dropped these particular columns because Ashesi financial aid do not consider Sat score of a student.
- Categorical features such as region, first_gen, and recommendation_strength were encoded numerically (into to 0s and 1s) for compatibility with the machine learning models.
- The dataset was split into three parts:
    - ✓ 60% for training, which allowed the model to learn ‘enough’ patterns in the data.
    - ✓ 20% for validation, which was used to tune the model parameters to prevent overfitting.
    - ✓ 20% for testing, which was used to evaluate the model's performance on new, unseen data

3. Model Training:

- Both the Random Forest and LogisticRegressionCV were trained on the same 20,000 training dataset.
- These trained models were then passed to the VotingClassifier from sklearn.ensemble library, and the ensemble was fitted to the training data.
- The ensemble model was saved using joblib for deployment in the Streamlit application.

4. Model Performance

The ensemble classifier yielded stronger predictive performance than either individual model alone. The graph below juxtaposes the test set accuracy of the random forest, logistic regression and the ensemble models:



The ensemble appropriately captured the strengths of both models: the interpretability and regularisation from LogisticRegressionCV and the nonlinear pattern recognition of Random Forest.

5. Hyperparameter Considerations

- LogisticRegressionCV was selected for its built-in ability to perform cross-validation while tuning the regularization parameter (C), which helps prevent overfitting. We used:
- cv=10: 10-fold cross-validation ensured the model's performance was robust across different data subsets.
- Cs=[-2, 2, 10]: This range of regularization strengths allowed the model to evaluate how much to penalize large coefficients, balancing between underfitting (high regularization) and overfitting (low regularization).
- max_iter=10000: Increased the number of iterations to ensure convergence, especially with multiple regularization values being tested.
- scoring='accuracy': Ensured that the regularization parameter chosen during cross-validation was the one that maximized classification accuracy.

RandomForestClassifier was trained using the following hyperparameter settings:

- n_estimators=200: The model constructs 200 decision trees. Increasing the number of trees enhances the model's generalization ability by reducing variance, especially with our dataset like 20,000 rows
- max_depth=15: Limits the maximum depth of each tree to prevent overfitting, ensuring the model captures patterns without memorising noise.
- min_samples_split=10: A node must have at least 10 samples to be split. This reduces the model's complexity and encourages wider splits, which improve generalizations.
- min_samples_leaf=4: Ensures that each leaf node has at least 4 samples, helping to reduce overfitting.
- random_state=42: Maintains reproducibility by ensuring consistent results across runs.
- n_jobs=-1: Leverages all available CPU cores to parallelize training to speed up the model fitting.