Project 2: Random Forests and Ensembles

Due date: October 18, 2023

Honour Statement

In doing this project, you must adhere to the following honour statement:

Red River College is committed to protecting the integrity of our curriculum ensuring the college continues to add value to our students and industry, while ensuring that students have opportunities to pursue their marks fairly, honestly and ethically.

This includes, but is not limited to, the fact that no collaboration, plagiarism, cheating, unauthorized collaboration, or false representation is permitted on assessments and is a violation of the S4 – Academic Integrity policy.

I understand I am subject to all of the same academic honesty requirements that apply during an in person assessment or online assessment. I understand that by beginning an exam, I accept and agree not to commit any violation of academic integrity.

I understand that there are consequences for violating the policy and as a Red River College student, I will not participate in or condone academic dishonesty.

Project 2: Random Forests and Ensembles

Objective: The goal of the project is to successfully train and test a random forest, compare it to a decision tree and knn classifier. Also to train a simple stacking ensemble using a dt, knn and svm as base learners and a random forest as the meta-model.

Instructions:

1. From the Caltech-UCSD CUB-200-2011 data set you will need `image_attribute_labels.txt` and `image_class_labels.txt` they can be found at the following links:

   http://www.vision.caltech.edu/datasets/cub_200_2011/

   https://drive.google.com/drive/folders/1kFzIqZL_pEBVR7Ca_8IKibfWoeZc3GT1

2. Follow the guidelines for the data preprocessing phase given below. You are welcome to preprocess the data as you wish, but good preprocessing is crucial for this project.

Task 1: Data Preprocessing

A correspondence should set up between the attributes and the labels. The aim is to set up a structure where each image's attributes can be related to its label. By the end, there are separate numpy arrays for training/testing attributes and their corresponding labels. In other words, your code takes two separate data sources – one with image attributes and the other with image labels – and processes them to form training and testing datasets where machine learning algorithms can learn the relationship between attributes and their corresponding labels.

1.1 The Data:

- The first file "image_attribute_labels.txt" contains information about image attributes. It is read with space as a delimiter('\s+)
- Only the first three columns should be loaded ('igid','attid','present')
- The second file "image_class_labels.txt" has class labels for the image. It is read with space delimeter too.
- Output the heads of both files, and the file sizes/shapes. Discuss contents in detail.

1.2 Handling Bad Entries:

- Occasionally, files may contain corrupted or wrongly formatted lines. We use the `on_bad_lines='skip'` argument to ensure such lines are skipped during loading.
- Use the `pandas` library to load the two text files, `image_attribute_labels.txt` and `image_class_labels.txt`. Name them imgatt, and imglabels, respectively.
- Always inspect the loaded data using methods like `head()`, `info()`, or `describe()` to understand its structure and to identify any inconsistencies.
- Output shape, and head of the image attributes

1.3 Data Transformation:

- The data from `image_attribute_labels.txt` needs to be transformed from a long format to a wide format. The `pivot` method in `pandas` will help transform the data such that each image ID has all the attributes associated with it in separate columns. Make sure there are no duplicates or pivot will give you an error. Name this imgatt2.
- Output the head of the new arrangement; and the number of rows and columns

1.4 Data Merging and Shuffle:

- Set your imglabels data index to "imgid'
- Merge the two dataframes on a common column, 'imgid' to create a new dataframe
- Randomly shuffle your dataframe using .sample with frac=1, and random_state=10. (I want this reproducible)

- Split your new dataframe into attributes and labels where your attributes contains all columns but the last. Your labels contain only the last column. Using .iloc is useful here as it preserves indexing.
- Convert to a numpy arrays
- Flatten your labels

2. **Training and Testing Sets**
    - Create your training and testing sets.

3. **Random Forest Classifier**:
    - Implement a Random Forest classifier. With max features=15, and 25 trees.
    - Train the classifier on the training data and test its performance on the testing data.
    - Print the training and testing accuracies. What do the scores suggest?
4. **Confusion Matrix**:
    - Generate a confusion matrix for the Random Forest classifier's predictions on the test data.
    - Visualize this matrix (you may need to implement or use a utility function for visualization).
5. **Model Comparisons**:
    - **KNN**:
        - Implement a K-Nearest Neighbors (KNN) classifier.
        - Train and test the KNN model.
        - Output the test results.
    - **Decision Tree**:
        - Implement a Decision Tree classifier.
        - Train and test the Decision Tree model.
        - Output the test results.
    - Compare the performance of the Random Forest, KNN, and Decision Tree models and report their accuracies.

6. **Cross-Validation**:
    - Perform 5-fold cross-validation on the Random Forest, KNN, and Decision Tree models.
    - Print the average accuracy and standard deviation for all three models.
7. **Hyperparameter Tuning**:
    - For the Random Forest classifier, experiment with different values of `max_features` (range: 5 to 50 in steps of 5) and `n_estimators` (range: 10 to 200 in steps of 10). Note: you may use GridSearchCV.(this can take a while you can shorten the search if you run into memory problems)
    - Output the grid search results, and indicate the best settings.
    - Visualize the results in a 3D scatter plot with the axes representing `max_features`, `n_estimators`, and `accuracy`.
    - Retrain your RF with the optimized settings and perform a final test.

- Output your test results.

8. **Stacking Ensemble**:
   - Create a stacking ensemble using the Decision Tree, KNN, and SVM as base learners, and a Random Forest as the meta-model.
   - Train this ensemble on the training data and evaluate its performance on the test data.
   - Compare the performance of the stacking ensemble to the previous models and report its accuracy.

9. **Improving Performance**
   - Improve the score of your ensemble using any means.

Hints:

- For hyperparameter tuning, you can use nested loops or other methods to systematically explore the parameter space.

Grading Rubric:

Criteria/Task	**Excellent (10)**	**Good (8)**	**Satisfactory (5)**	**Needs Improvement (1-2)**	**Not Attempted (0)**
1.1 The Data	Displayed the shape, and discussed contents in detail.	Briefly discussed the contents and display functions	Minor issues or missed discussing some aspects of the content or display functions	Faced significant issues while using display functions or major elements in discussion.	Did not attempt
1.2 Handling Bad Entries	Efficiently handled bad entries, inspected data using multiple methods, presented comprehensive output, and successfully loaded all data	Handled bad entries and inspected the data with one method, presented clear output. Successful data loading.	Addressed bad entries, but faced minor issues during inspection or presentation. Successful data loading	Missed handling some bad entries or lacked clarity in presentation. Data has issues.	Did not attempt or data is unusable.
1.3 Data Transformation	Demonstrated expertise in data transformation, presented the	Completed data transformation	Completed data transformation but faced minor issues or missed	Faced significant challenges in data	Did not attempt.

	head of the new arrangement and accurately counted rows and columns.	and presented most outputs.	presenting some outputs.	transformation or missed major outputs.	
1.4 Data Merging	Excellently merged data, displayed all required outputs, and separated attributes and labels accurately.	Successfully merged data and displayed most required outputs.	Merged data but faced minor issues or missed some outputs.	Had significant problems merging data or missed displaying many outputs.	Did not attempt.
2. Training and Testing Sets	Correctly and efficiently separated training and testing sets.	Separated training and testing sets with minor hitches.	Faced issues in separation	Had significant difficulties in creating training/testing sets.	Did not attempt.
3. Random Forest Classifier	Successfully implemented, trained, and tested the classifier, displaying accurate training and testing accuracies.	Implemented the RF classifier and displayed accuracies but with minor issues.	Faced issues in implementation or displayed one of the accuracies.	Struggled significantly with implementation or missed displaying accuracies.	Did not attempt.
4. Confusion Matrix	Skillfully generated and visualized the confusion matrix.	Generated the matrix and provided a basic visualization.	Faced minor issues in generation or visualization.	Had significant problems generating or visualizing the matrix.	Did not attempt.
5. Model Comparisons	Successfully implemented, trained, tested, and compared all three models, providing a detailed report.	Implemented and compared models but with minor shortcomings.	Faced issues in one or more model implementations or lacked detail in the comparison.	Had significant difficulties implementing or comparing models.	Did not attempt.
6. Cross-Validation	Performed 5-fold CV on all models, displaying average accuracy and standard	Performed CV on all models but missed some details in presentation.	Faced minor issues in CV or missed one of the required outputs.	Struggled with CV or missed multiple required outputs.	Did not attempt.

	deviation accurately.				
7. Hyperparameter Tuning	Expertly tuned hyperparameters, visualized results in 3D plot, and retrained RF, displaying test results.	Conducted tuning and visualization, but with minor issues or omissions.	Faced challenges in tuning, visualization, or retraining, or missed one required output.	Had significant issues with tuning or visualization or missed multiple outputs.	Did not attempt.
8. Stacking Ensemble	Successfully created and trained the ensemble, evaluated its performance, and provided a detailed comparison to other models.	Created and trained the ensemble and provided basic comparison.	Faced minor issues in ensemble creation, training, or comparison.	Struggled significantly with ensemble tasks or lacked detail in comparison.	Did not attempt.
9. Ensemble Improvement	Significantly improved ensemble with discussion.	Successfully improved ensemble	Minor improvements with discussion	Minor improvements	Did not attempt
10. Comments	All work is neat, organized, and submitted on time. Code is commented thoroughly.	Code has minor commenting issues	Code is somewhat commented.	Code has minor commenting	No comments
11. Organization	Work is thoroughly organized	Work has minor organizational problems	Work has significant organizational problems	Work has major organizational problems	No Organization
Total:	**/140**				

Submission: Submit your Jupyter notebook with your working code and answers to the Dropbox before the due date.