FEATURE EXTRACTION METHODS WITH MACHINE LEARNING FOR SATELLITE IMAGE CLASSIFICATION

Zack Strathe

CIS 731

INTRODUCTION

- Satellite imagery has capability to track indicators, like deforestation, desertification, or general crop health
- With such a massive amount of data available, there exists a potential challenge in gathering meaningful insights
- Algorithms can be trained to classify satellite imagery, resulting in a model that could track fluctuations (i.e., when a classification changes)
- While state-of-the-art is a convolutional neural network, I’m implementing “classic” machine learning techniques to allow flexibility with feature extraction methods

DATA SET

- Images: 28x28 pixel, 4-band (red, green, blue, near-infrared), stored in .csv file as a flattened list for each image
  - Each pixel sub-value represented by 0-255 color value
- Labels: one-hot-encoded corresponding to either barren land, trees, grassland, or other, stored in .csv file
  - Labeling process: manually labeled 6,000 x 7,000 pixel tiles, then split into 28x28 samples using sliding window blocks

- Training set: 400,000 images & labels
- Testing set: 100,000 images & labels
- Source: Kaggle (https://www.kaggle.com/crawford/deepsat-sat4)

METHODOLOGY: PLATFORM

- Using PySpark for all steps:
  - Load data, preprocess data, train classification model, evaluate classification model
- Cloud Computing
  - Because of the large size of the dataset (~7 GB), I needed to utilize a VM
    - from Google Cloud with a v16 CPU and 64 GB memory
  - Also tested a AWS EMR notebook with a cluster consisting of 3 VMs, which performed better but was more costly
  - When using a single-VM, PySpark configuration settings should be set to fully utilize available resources
    - Using SparkConf

METHODOLOGY: DATA PREPROCESSING

- Initially, imported each .csv file as a PySpark dataframe, but convert to a RDD
  - Using RDD format for ease of implementation, because a schema is not required
- Mapped X_train and X_test with functions to transform/extract image features
- Mapped Y_train and Y_test with function to convert one-hot-encoded labels into floats
- Performed feature selection on X_train and X_test with ChiSqSelector and Normalizer from MLlib
- Joined X_train and Y_train as a LabeledPoint, with X_train data formatted with the Mllib Vectors class
  - Required for Mllib RDD-based classification algorithms

FEATURE EXTRACTION METHODS

- Pixel-based transformations:
  - Mean value for each pixel (excl. near-infrared value)
  - Near-infrared value only
  - Mean value & near-infrared value
- OpenCV-based (global) transformations:
  - Edge detection (cv2.Sobel)
  - Hu Moments (cv2.HuMoments)
  - Histogram (greyscale)
  - Also tried combining each of these with original image data

Parameter	Transformation Effect	Resulting Number of Features per Image
none (default)	n/a	3,136
flatten_pixels	returns the mean of the RGB values for each pixel	784
infra_only	returns only the infrared value for each pixel	784
flatten_plus_infra	returns the mean of the RGB values, and the infrared value for each pixel	1,568
edges_only	returns an array of edges detected using cv2.Sobel	784
edges_plus_pixels	returns an array of edges, collated within each pixel sub-array in the default image data	3,920
hu_moments	returns an array of HuMoments calculated from cv2.HuMoments	7
hu_moments_plus_ pixels	returns an array of HuMoments, appended to the end of the default image data array	3,143
histogram_greyscale	returns a greyscale histogram array, binned by RGB value (0-255)	256
histogram_greyscal e_plus_pixels	returns a greyscale histogram array, binned by RGB value (0-255), appended to the end of the default image array	3,392

EVALUATION METRIC

- Weighted F1 score:
  - This problem is multiclass, so the trained model may exhibit classification bias
  - Therefore, it's ideal to utilize precision (ratio of true positives to total positives) and recall (ratio of true positives to the sum of true positives and false negatives)
  - To maximize both precision and recall, use F1
    - Harmonic mean of precision and recall
  - Weighted-F1 aggregates over all classes

INITIAL EVALUATION OF MLLIB RDD-BASED ALGORITHMS USING UNMODIFIED IMAGE DATA

Algorithm Name	Precision Score	Recall Score	F1 Score	Accuracy Score	Total Time
Random Forest	0.82	0.81	0.81	0.81	287.75
Decision Tree	0.77	0.76	0.76	0.76	286.21
Logistic Regression	0.73	0.74	0.73	0.74	3,373.62
Naive Bayes	0.57	0.51	0.51	0.51	231.45
Gradient Boosted Trees	0.31	0.44	0.33	0.44	1,465.30
Support Vector Machine	0.04	0.20	0.07	0.20	247.38

- Random Forest, Decision Tree, and Logistic Regression performed fairly well, and will be used to evaluate feature extraction methods

- While Logistic Regression had a substantially longer processing time, I’m assuming time to be insignificant (because a PySpark cluster could simply be scaled-out with additional workers when needed)

FEATURE EXTRACTION EVALUATION RESULTS (1/4): RANDOM FOREST

Algorithm Name	Feature Extraction Method	Precision Score	Recall Score	F1 Score	Accuracy Score	Total Time
Random Forest	*histogram_greyscale_plus_pixels*	0.85	0.83	0.83	0.83	580.61
Random Forest	*hu_moments_plus_pixels*	0.83	0.83	0.82	0.83	771.57
Random Forest	*histogram_greyscale*	0.83	0.83	0.82	0.83	384.29
Random Forest	*edges_plus_pixels*	0.83	0.82	0.82	0.82	861.28
Random Forest	*flatten_plus_infra*	0.79	0.79	0.78	0.79	677.44
Random Forest	*hu_moments*	0.73	0.72	0.71	0.72	392.47
Random Forest	*flatten_pixels*	0.58	0.69	0.62	0.69	489.09
Random Forest	*infra_only*	0.55	0.61	0.53	0.61	296.88
Random Forest	*edges_only*	0.29	0.46	0.35	0.46	401.36

FEATURE EXTRACTION EVALUATION RESULTS (2/4): DECISION TREE

Algorithm Name	Feature Extraction Method	Precision Score	Recall Score	F1 Score	Accuracy Score	Total Time
Decision Tree	*histogram_greyscale_plus_pixels*	0.88	0.87	0.88	0.87	543.25
Decision Tree	*histogram_greyscale*	0.85	0.86	0.85	0.86	367.05
Decision Tree	*edges_plus_pixels*	0.77	0.76	0.76	0.76	798.13
Decision Tree	*hu_moments_plus_pixels*	0.77	0.77	0.76	0.77	744.53
Decision Tree	*flatten_plus_infra*	0.73	0.73	0.73	0.73	653.72
Decision Tree	*hu_moments*	0.73	0.73	0.71	0.73	385.65
Decision Tree	*flatten_pixels*	0.67	0.68	0.67	0.68	471.40
Decision Tree	*infra_only*	0.49	0.59	0.52	0.59	293.86
Decision Tree	*edges_only*	0.38	0.44	0.38	0.44	388.65

FEATURE EXTRACTION EVALUATION RESULTS (3/4): LOGISTIC REGRESSION

Algorithm Name	Feature Extraction Method	Precision Score	Recall Score	F1 Score	Accuracy Score	Total Time
Logistic Regression	*histogram_greyscale*	0.93	0.92	0.93	0.92	591.54
Logistic Regression	*histogram_greyscale_plus_pixels*	0.92	0.92	0.92	0.92	4,348.75
Logistic Regression	*edges_plus_pixels*	0.88	0.88	0.88	0.88	5,217.39
Logistic Regression	*hu_moments_plus_pixels*	0.76	0.76	0.76	0.76	4,267.64
Logistic Regression	*flatten_plus_infra*	0.46	0.46	0.46	0.46	2,357.31
Logistic Regression	*hu_moments*	0.47	0.44	0.37	0.44	402.00
Logistic Regression	*flatten_pixels*	0.33	0.30	0.23	0.30	1,382.87
Logistic Regression	*edges_only*	0.14	0.35	0.19	0.35	1,147.40
Logistic Regression	*infra_only*	0.15	0.28	0.17	0.28	1,193.51

FEATURE EXTRACTION EVALUATION RESULTS (4/4): OVERALL

- Best performing model:
  - Algorithm: Logistic Regression
  - Feature extraction method: Greyscale histogram
  - Weighted-F1 score: 0.93

- Interesting observations:
  - Using greyscale histogram features, only the Logistic Regression algorithm was significantly improved
  - Results from the Random Forest algorithm didn't significantly improve with any feature extraction method

FEATURE SELECTION FOR FURTHER MODEL IMPROVEMENT

New Parameter	ChiSq Num	Algorithm Name	Feature Extraction Method	Precision Score	Recall Score	F1 Score	Accuracy Score	Total Time
Normalize Features	*N/A*	*Logistic Regression*	*histogram _greyscale*	0.93	0.93	0.93	0.93	609.16
ChiSq Selection	*200*	*Logistic Regression*	*histogram _greyscale*	0.92	0.91	0.91	0.91	779.75
ChiSq Selection	*100*	*Logistic Regression*	*histogram _greyscale*	0.73	0.75	0.73	0.75	692.11

- Evaluated best model (Logistic Regression with greyscale histogram for features) with:
  - Normalize Features
    - Normalizes the features for each image
  - Chi-Square Selection
    - Selects the top number of features, using chi-squared test
    - Evaluated with "top number" specified to 200 and 100

- Only normalizing features improved the evaluation results, boosting both precision and accuracy by 0.01

10-FOLD CROSS-VALIDATION (1/2): AVG. WEIGHTED-F1 SCORE

- For cross-validation of best model, combined training and testing sets into a single RDD consisting of 500,000 images and labels
- Iterated through 10 unique seed values, used RDD.randomSplit() function to uniquely split data for each fold
- Cross-validated weighted-F1 score for improved model: 0.92

10-FOLD CROSS-VALIDATION (2/2): PAIRED T-TEST

- Computed cross-validation metrics for baseline model
  - Random Forest with unmodified image data
  - Avg. weighted-F1 score: 0.82
  - Saved weighted-F1 score of each fold

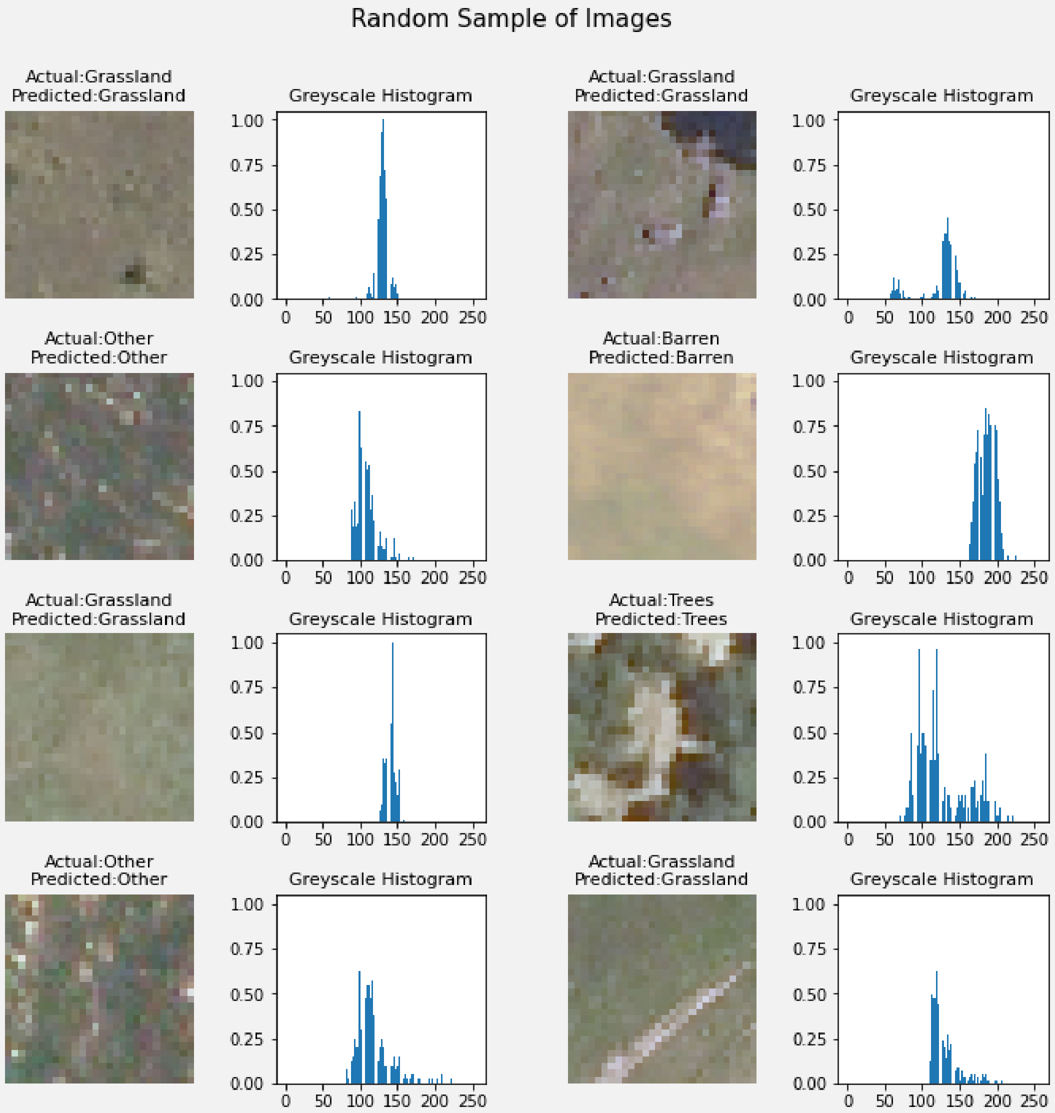
- Computed cross-validation metrics for best-model
  - Saved weighted-F1 score of each fold

- Using SciPy ttest_rel function, computed paired t-test between baseline and improved model
  - T-test statistic: -236.34
  - P-value: 2.21e-18
  - Because $p < 0.05$, conclude with 95% confidence to reject the null hypothesis

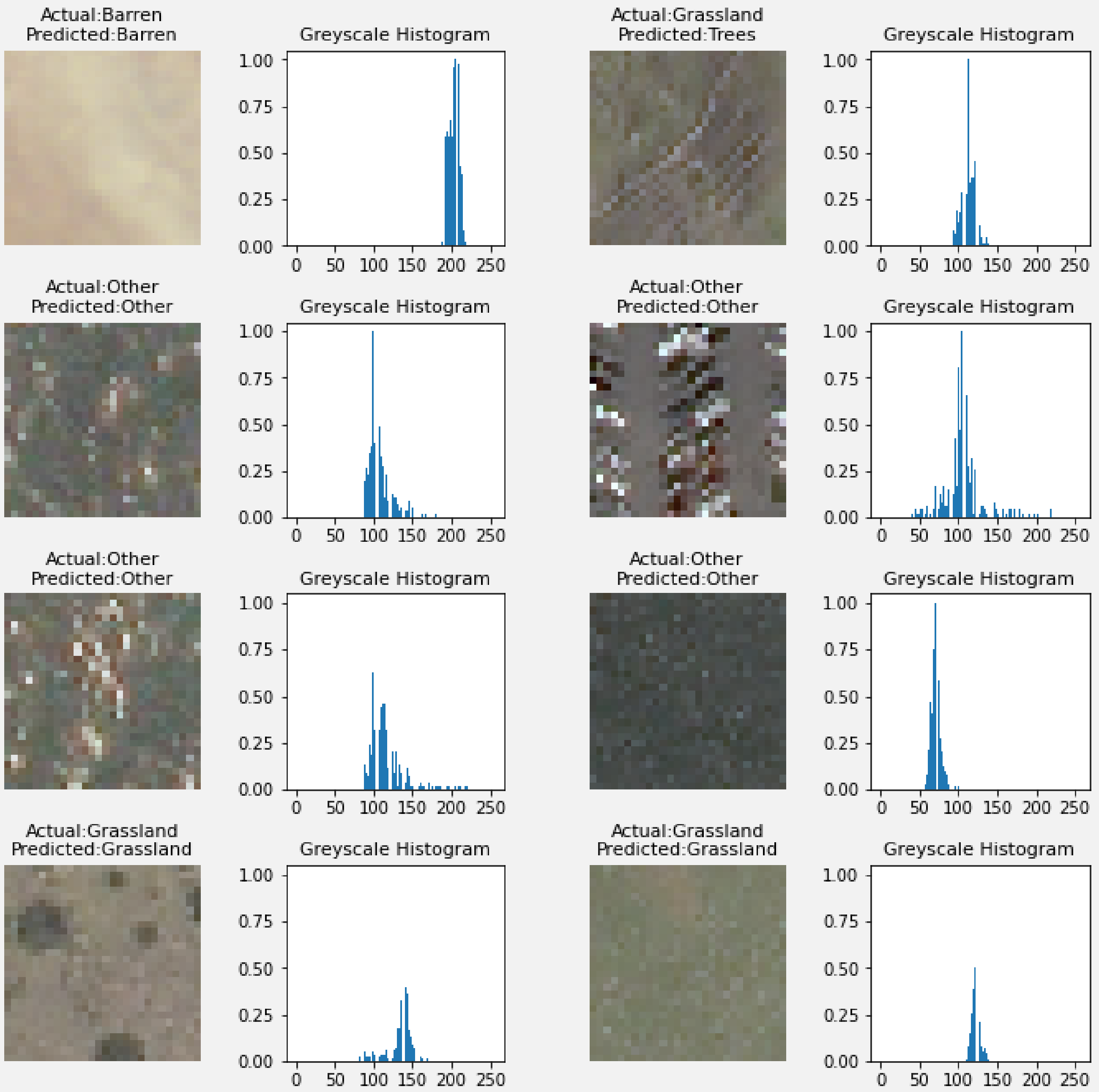
CONCLUSION

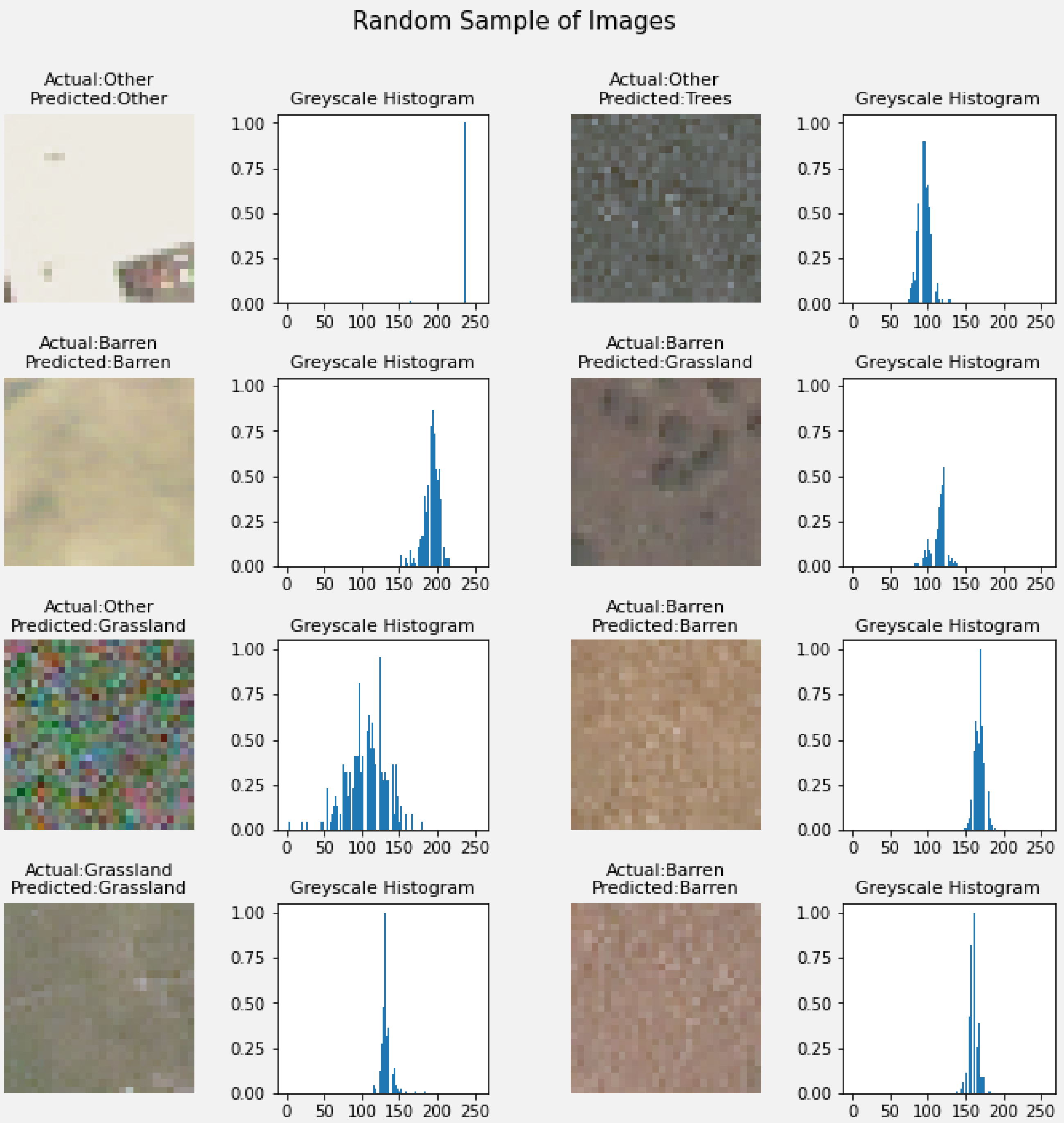
- Best model:
  - Algorithm: Logistic Regression
  - Feature extraction method: Greyscale histogram
  - Feature selection: Normalize

- Improved weighted-F1 score from 0.81 to 0.93

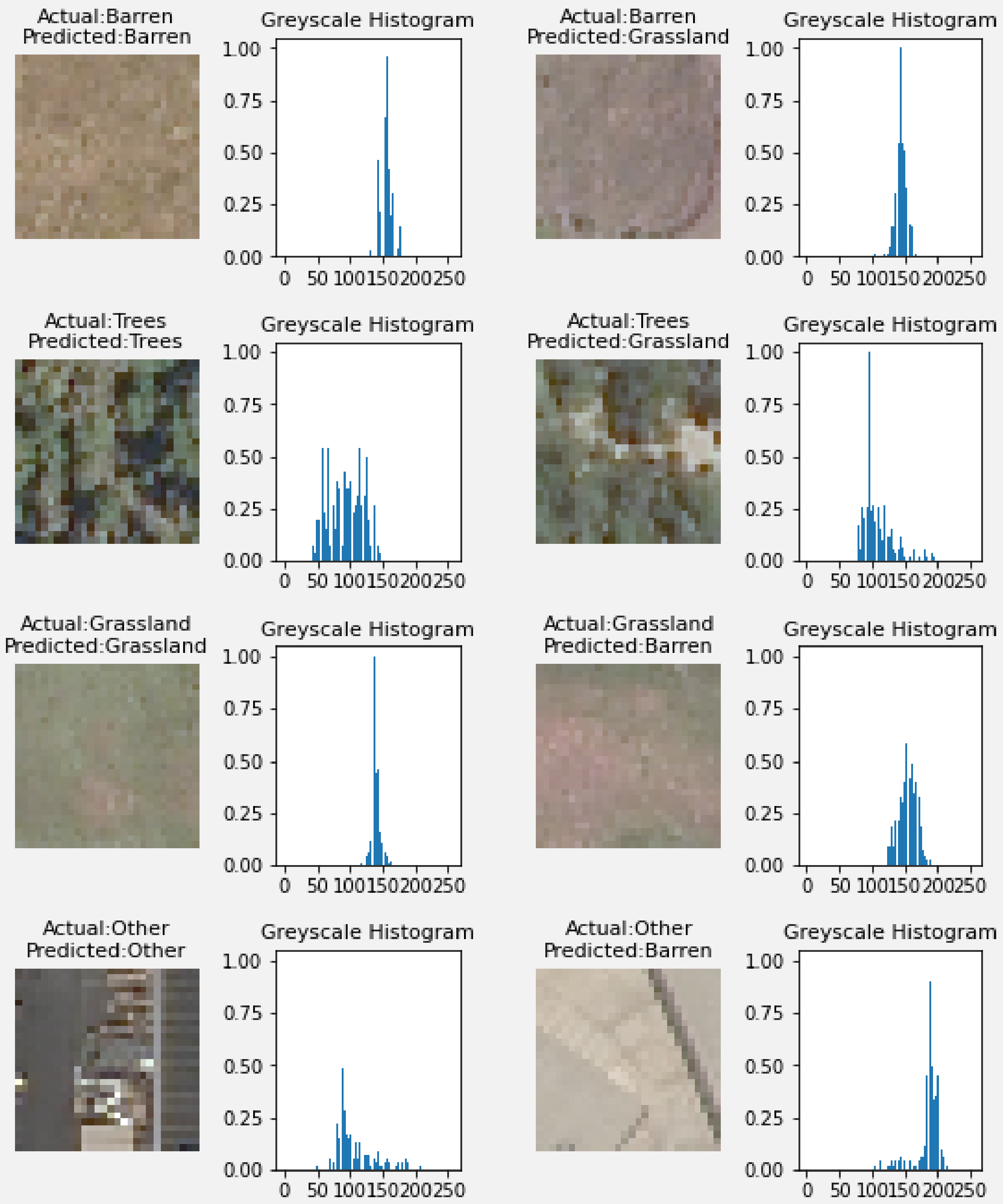


Random Sample of Images




Random Sample of Images
Actual:Other
Predicted:Other
Greyscale Histogram
Actual:Other
Predicted:Trees
Greyscale Histogram
Actual:Barren
Predicted:Barren
Greyscale Histogram
Actual:Barren
Predicted:Grassland
Greyscale Histogram
Actual:Other
Predicted:Grassland
Greyscale Histogram
Actual:Barren
Predicted:Barren
Greyscale Histogram
Actual:Grassland
Predicted:Grassland
Greyscale Histogram
Actual:Barren
Predicted:Barren
Greyscale Histogram
0.00 0.25 0.50 0.75 1.00
0 50 100 150 200 250

Sample of Images
Correct and Incorrect Prediction for Each Category



FUTURE WORK

- Attempt other histogram extraction techniques
  - Full-color or near-infrared values instead
- Evaluate Mllib dataframe-based algorithms that I didn’t test
  - Multilayer Perceptron Classifier, One-Vs-Rest Classifier, Factorization Machines Classifier
- Add many additional classification categories
  - Improve overall usability of model
  - Example categories:
    - Mountainous
    - Water
    - Clouds (to identify regions where clouds have obscured the image, and should have new imagery sourced)