Rice Classification Methods: Derived Features vs. Images

Nick Melamed, Stephanie Owyang, Suvass Ravala

Motivation for Looking at Rice Grains

Why is this helpful?

- Rice is consumed by roughly half of the world's population
- Understanding the characteristics of different grains allows for seed quality evaluation

Current Work on the Subject

- Main paper: Koklu, M., Cinar, I., and Taspinar, Y.S. 2021
    - Used image dataset to achieve 100% accuracy w/ CNN, derived feature dataset to achieve 99% accuracy via ANN
    - Previous work referenced in the paper (w/ rice and other crops) hardly broke above 95%.

Our Data

Image Dataset - Turkey

- 15K images per rice variety, 5 grains
  - Arborio
  - Basmati
  - Ipsala
  - Jasmine
  - Karacadag

Derived Feature Dataset - Cinar 2019

- 106 features derived from images
  - 90 color (RGB, other color scales)
  - 12 morphological
  - 4 shape

Why 2 Different Datasets?

Can’t we just do one?

- We were curious if features derived from images could produce the same results as the images themselves
  - We would assume images are superior…

Derived Feature Analysis

Data Cleaning/EDA

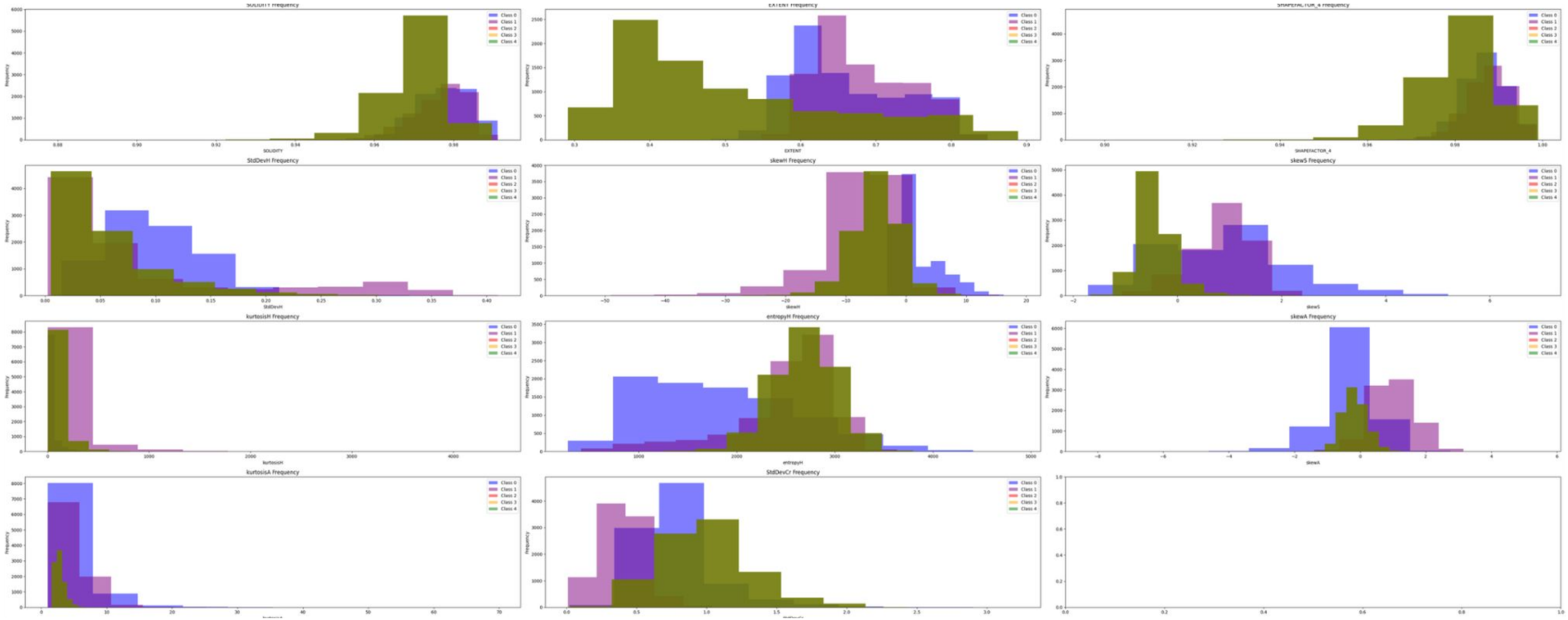
Train/Test/Validation: 60-20-20

Missing Values: 16 training, 6 validation

Normalization: Min/Max to reduce range of values

Multicollinearity: Drop columns with $r > 0.8 \rightarrow$ 106 to 11 features

Feature Inspection



Baseline Model

What is our starting point?

- Majority Class Classifier: All predictions = majority class (0)
  - Accuracy: 0.2018
  - Loss: 139599171820.3884 (!)
  - Why so poor? Good class balance:

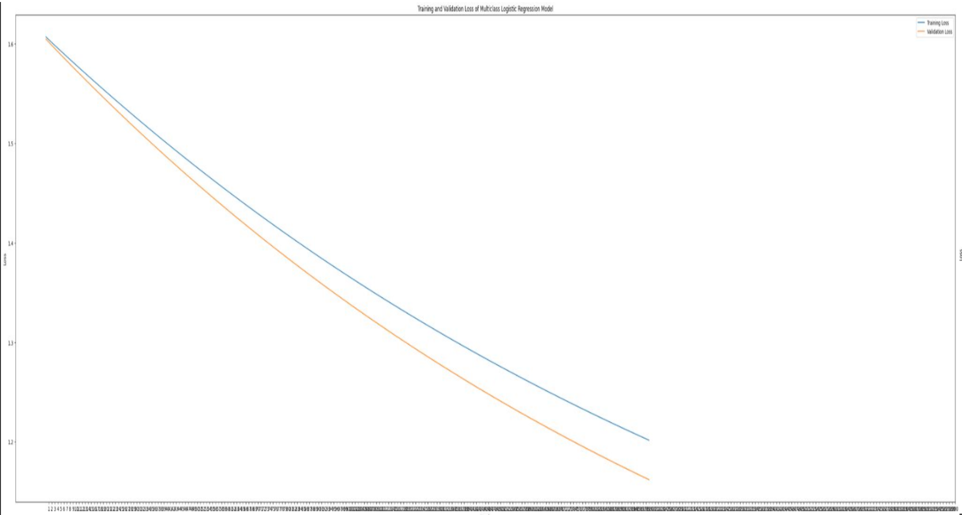
```
CLASS
0          9081
2          9026
4          8993
1          8953
3          8947
Name: count, dtype: int64
```

Attempt #1: Logistic Regression

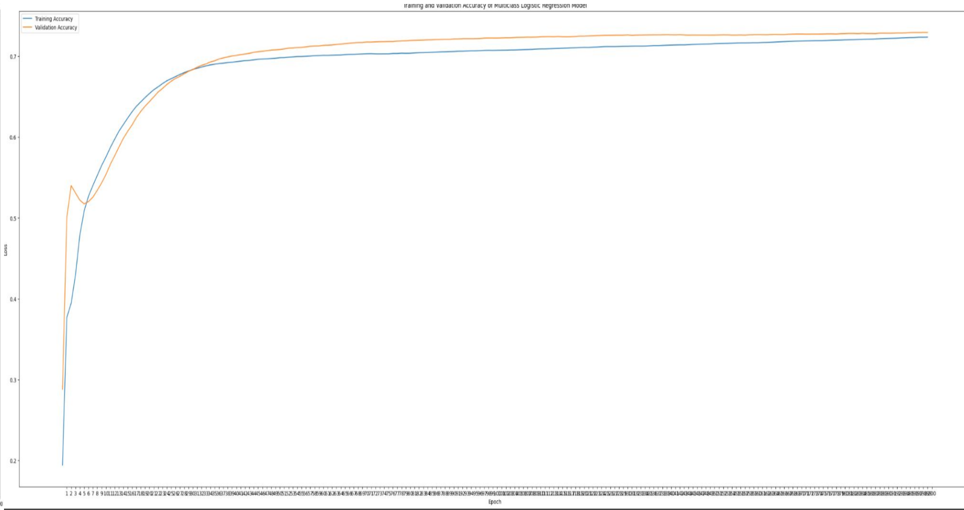
Hyperparameter Tuning: Keras Tuner

Optimal Learning Rate: ~9.7e-5
Optimal Epochs: 200
Optimal Batch Size: 32
Optimal Regularization Parameter: 1e-5 (Elastic Net)

Training and Validation Loss:



Training and Validation Accuracy:



Generalizability and Overfitting Fixes

Attempts to Increase Accuracy:
Reduce LR if Accuracy Doesn’t Increase
Use lower LR in case of complex loss function

Attempts to Reduce Overfitting:
Early Stopping if Loss Doesn’t Improve
Elastic Net Regularization (50/50 L1/L2)

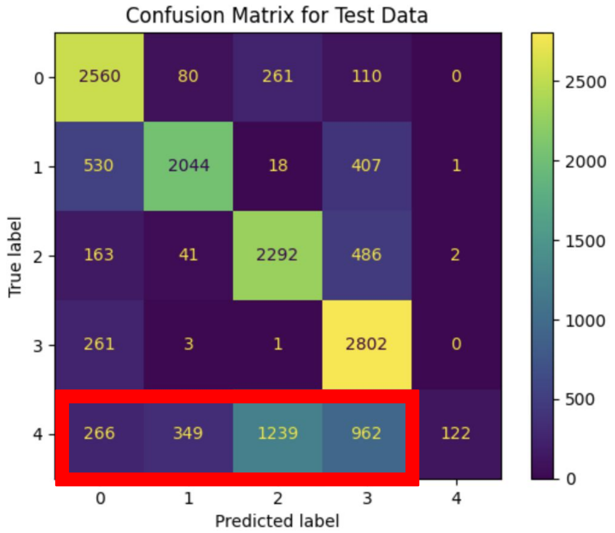
Training Accuracy	Validation Accuracy	Test Accuracy
0.724	0.729	0.654

Limitations

Why not 100% Accuracy?

- Pretty substantial improvement in accuracy, but still rather low

- Logistic Regression cannot model nonlinear relationships between features!



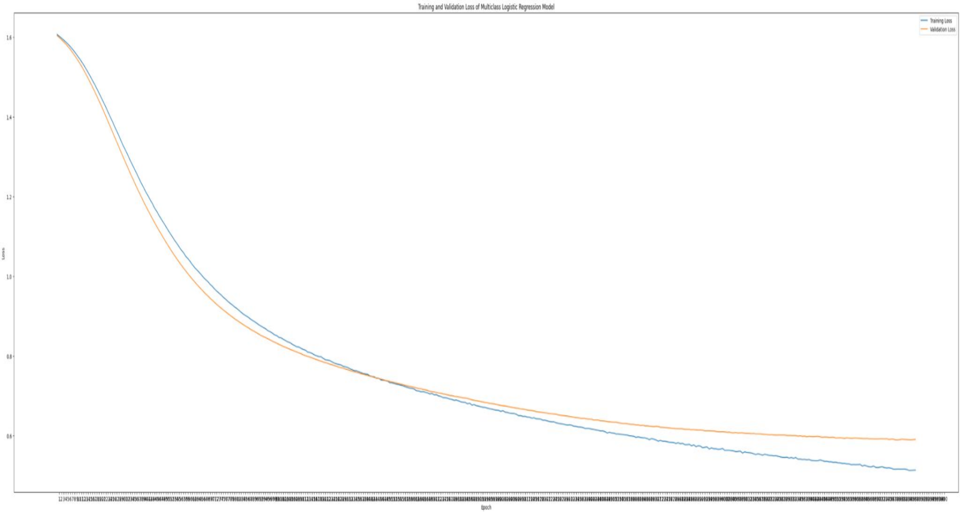
Attempt #2: Feed Forward Neural Network

Hyperparameter Tuning: Keras Tuner

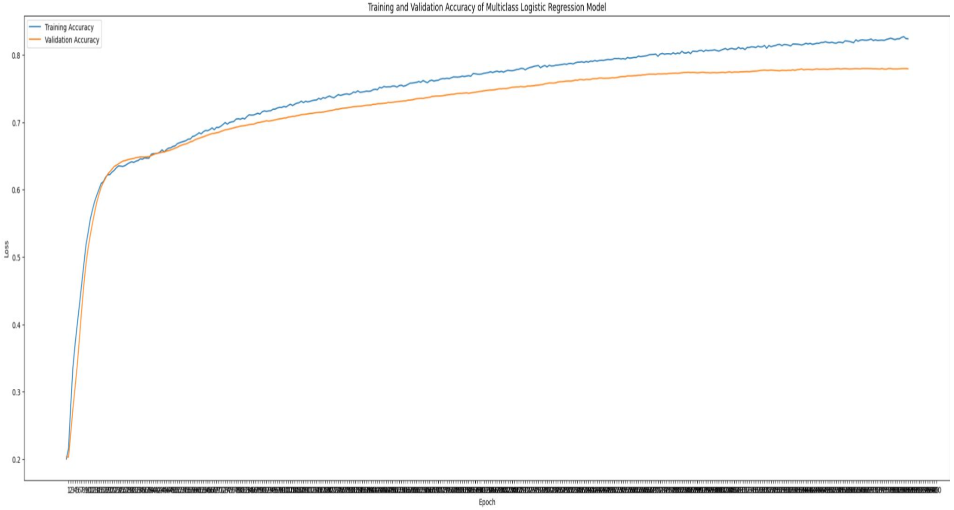
Optimal Learning Rate: ~9.7e-6
Optimal Epochs: 388
Optimal Batch Size: 32
Optimal Regularization Parameter: 1e-5 (Elastic Net)

Optimal Hidden Layers: 1 with 32 neurons
Optimal Dropout Rate: 0.0

Training and Validation Loss:



Training and Validation Accuracy:



Generalizability and Overfitting Fixes

Attempts to Increase Accuracy:
Reduce LR if Accuracy Doesn’t Increase
Use lower LR in case of complex loss function

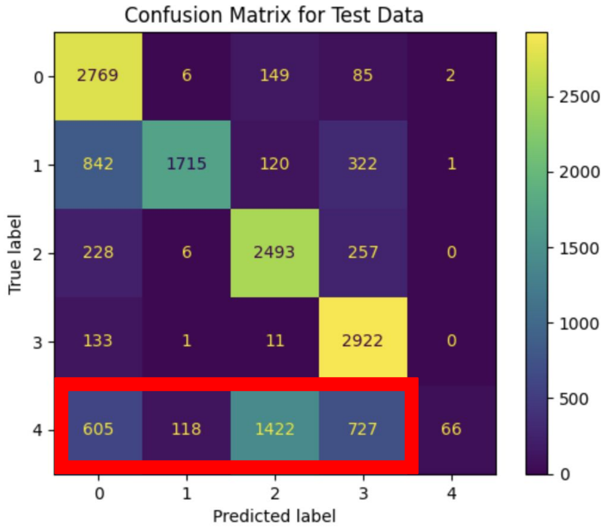
Attempts to Reduce Overfitting:
Early Stopping if Loss Doesn’t Improve
Elastic Net Regularization (50/50 L1/L2)
Implemented Dropout and Gaussian Noise (although tuning preferred no dropout)

Training Accuracy	Validation Accuracy	Test Accuracy
0.842	0.779	0.649

Limitations

Why not 100% Accuracy?

- Adding nonlinearity didn't make a huge difference…
- Could be high level of noise in data (relatively low number of observations, lot of features), curse of dimensionality, etc.



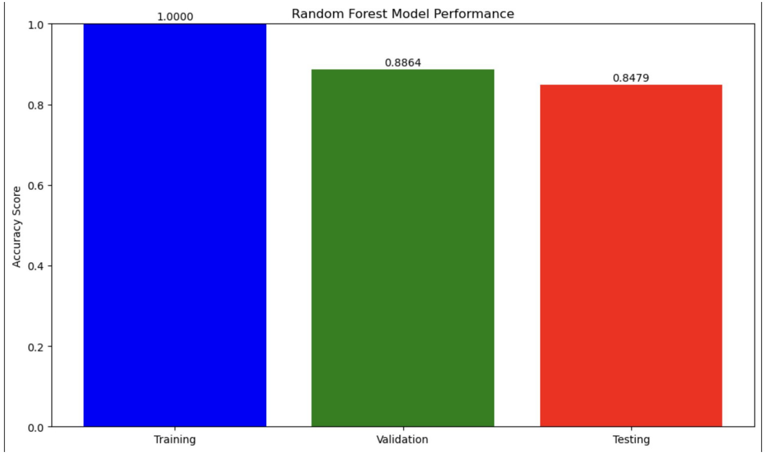
Attempt #3: Random Forest, XGBoost

Random Forest - Hyperparameter Tuning

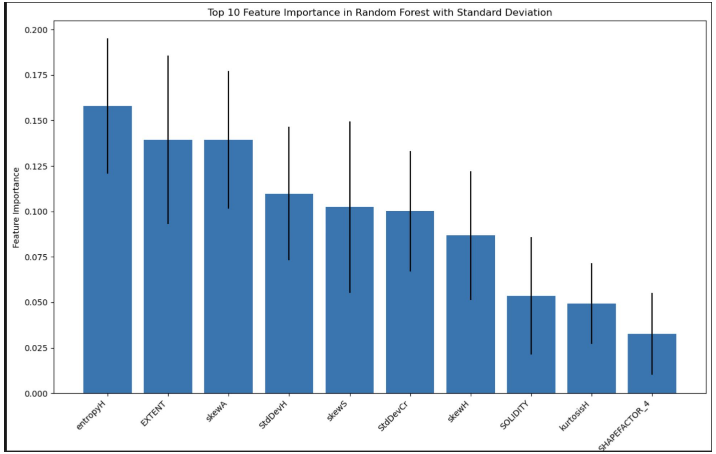
Number of Trees(n_estimators): 100
Max_Depth : None
Max_Features: sqrt
Random State: 1234

Class Weights: Balanced
N Jobs: -1

Training, Validation and Testing Accuracies:



Feature Importance:



Random Forest - Generalization

Attempts to Increase Accuracy (0.76 -> 0.88):
Increase Parameters (N_jobs, Max_Depth, N_Estimators)
Add Parameters (Max_Features, Bootstrap)

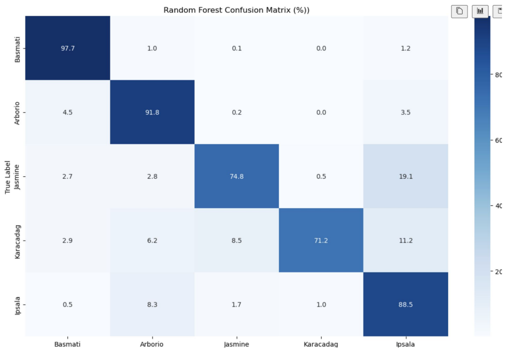
Attempts to Reduce Overfitting:
Large number of n_estimators and averaged results
Random number of features each time (different subsets of data)

Training Accuracy	Validation Accuracy	Test Accuracy
1.000	0.891	0.884

Limitations

Why not 100% Accuracy?

- Hard to understand the relationship between features and how random forest interprets them

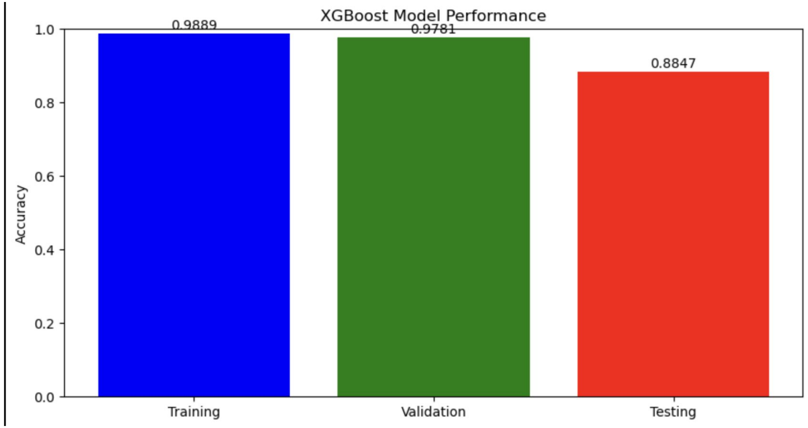


XGBoost - Hyperparameter Tuning

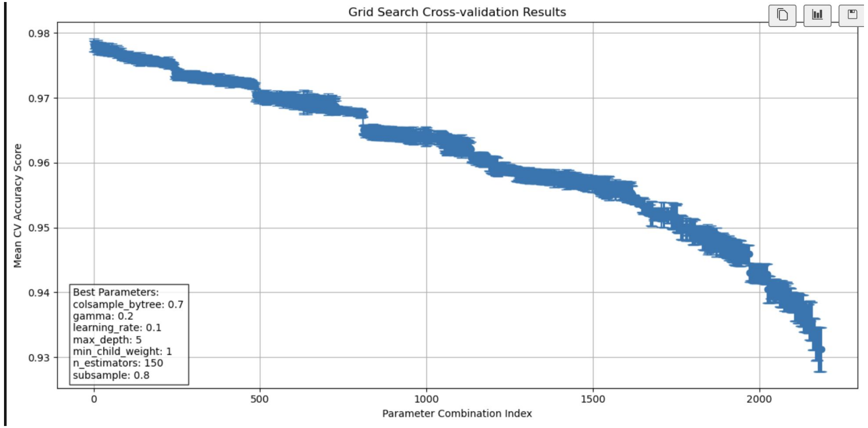
Number of Trees(n_estimators): 150
Max_Depth : 5
Learning_Rate: 0.1

Sub Sample : 0.8
Col_bytree: 0.7

Training, Validation and Testing Accuracies:



Cross Validation:



XGBoost - Generalization

Attempts to Increase Accuracy (0.73 -> 0.89):
Use GridSearch methods : n_estimators, max_depth, earning_rate, sub_sample, col_subtree

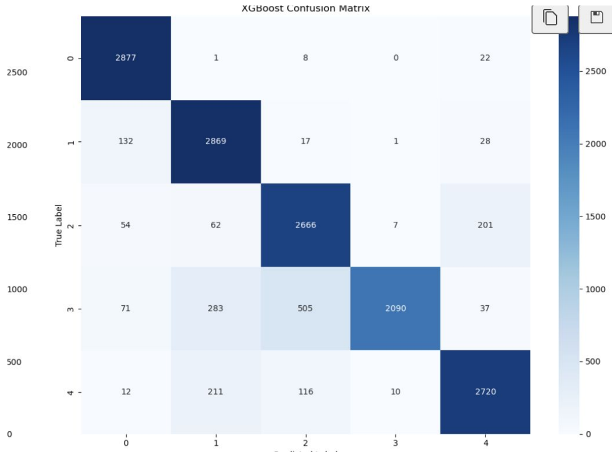
Attempts to Reduce Overfitting:
Large number of n_estimators and averaged results
Random number of features each time (different subsets of data)

Training Accuracy	Validation Accuracy	Test Accuracy
0.98	0.979	0.885

Limitations

Why not 100% Accuracy?

- Limited Parameters tested for Grid Search optimization



In general, where does our Derived Feature analysis fall short?

Limitations of using Derived Features

- Could have Explored and Generated more Features like Perimeter, Aspect_Ratio (Issue of Multicollinearity...)

- Couldn't Explore All the Feature Relationships

- Limited Domain Specific Knowledge

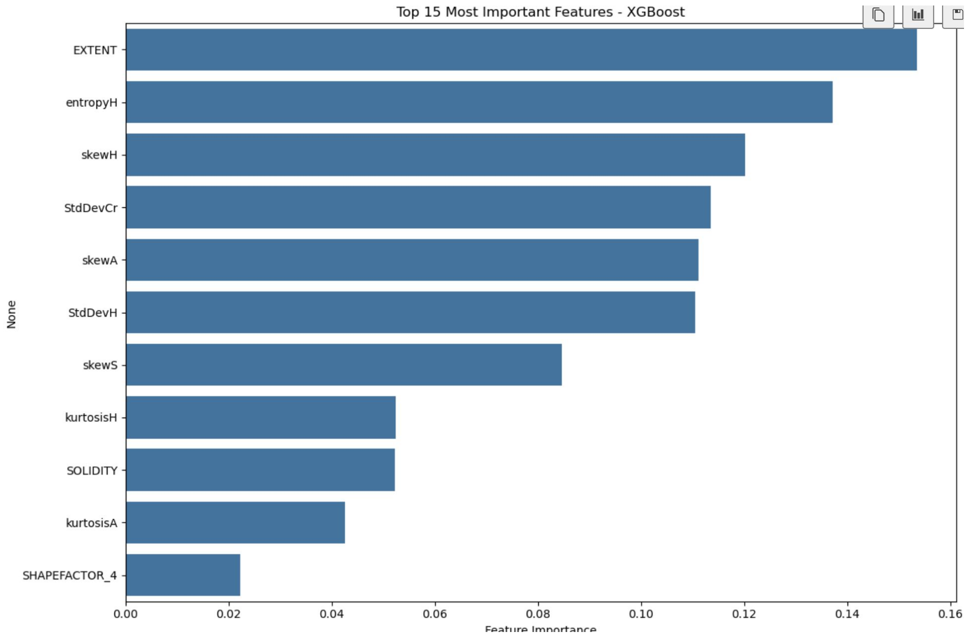


Image Data Analysis

Image Exploratory Data Analysis

75,000 images total

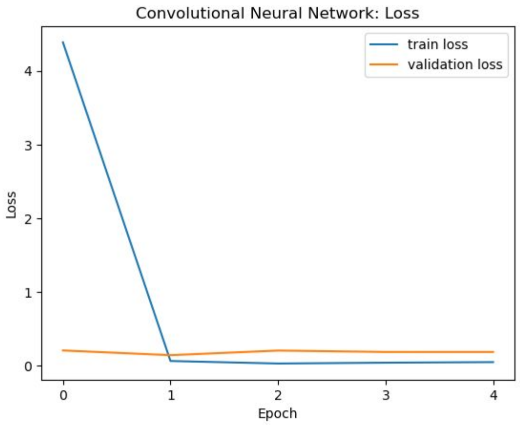
Balanced classes - 15,000 images per rice grain type

Attempt #4: Convolutional Neural Network

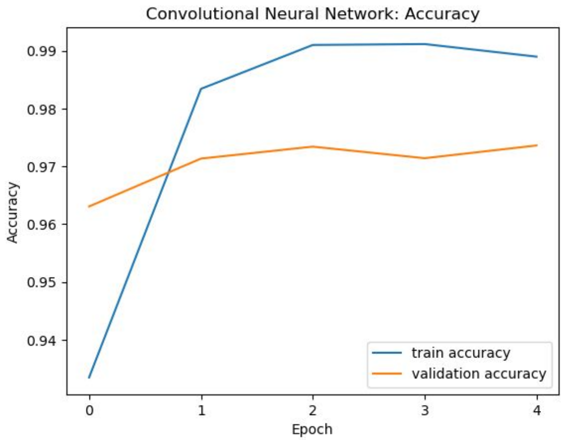
CNN - Baseline

Epochs: 5
Batch Size: 32
Number of Layers: 1
Pooling Method: Max Pooling 2D

Training and Validation Loss:



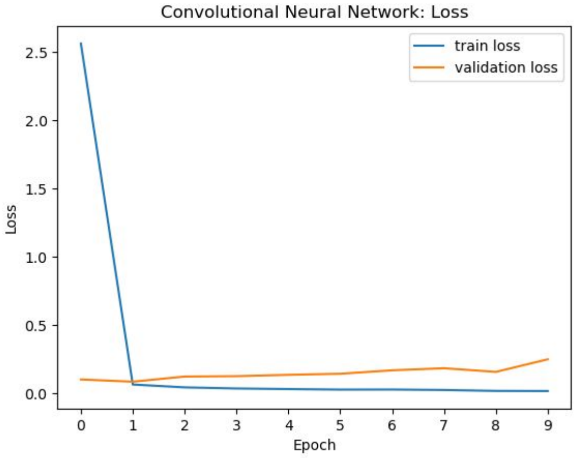
Training and Validation Accuracy:



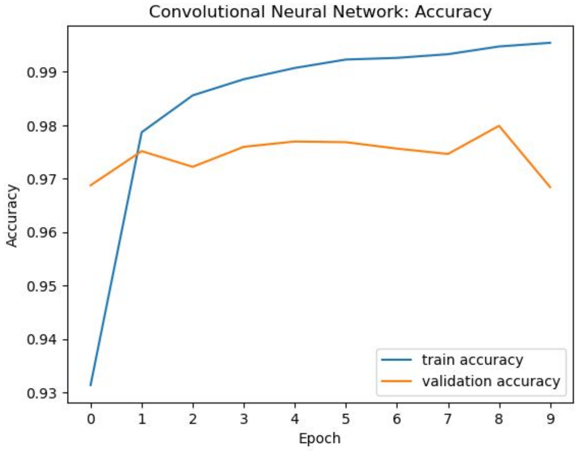
CNN - Hyperparameter Tuning

Optimal Epochs: 10
Optimal Batch Size: 32
Optimal Number of layers: 2 layers
Optimal Pooling Methods: Max Pooling

Training and Validation Loss:



Training and Validation Accuracy:



CNN - Generalization

Attempts to Increase Accuracy
Adding layers
Using different types of activation methods and pooling methods

Attempts to Reduce Overfitting
Using different types of activation methods and pooling methods
Early stop

Training Accuracy	Validation Accuracy	Test Accuracy
0.9948	0.9684	0.9686

Limitations

Why not 100% Accuracy?

- Computational cost - each model iteration consumes a lot of memory
- Model training time - Each iteration of the model takes a long time to run and is limited to your hardware (or borrowed hardware)

Concluding Remarks

Summary of Results

	Training Accuracy	Validation Accuracy	Test Accuracy
Logistic Regression	0.724	0.729	0.654
FFNN	0.842	0.779	0.649
Random Forest	1.000	0.891	0.884
XGBoost	0.98	0.979	0.885
CNN	0.994	0.968	0.968

Comparison of Methods

Method	Dataset	Advantages	Disadvantages
Logistic Regression	Derived Features CSV	Simple to implement, good baseline	Poor non-linear performance
FFNN	Derived Features CSV	Handles non-linear relationships	Prone to overfitting
Random Forest	Derived Features CSV	Improved performance on non-linear patterns	Less interpretable
XGBoost	Derived Features CSV	More tuning parameters than random forest, usually improves performance	Complex tuning
CNN	Rice Grain Images	Captures spatial hierarchies and features from raw images	Computationally intensive

Further Research

Where do we go from here?

- Continue tuning CNN models to improve generalization
- Adding more types of rice like bomba, black, red or wild rice
- How would this model do with a different background with more noise?
  - How can this be used to make the rice production process more efficient?

Thank You!

Contributions

Nick: Motivation, EDA/Data Cleaning, Logistic Regression, Feed Forward Neural Network

Suvass: Random Forest, XGBoost, Limitations of Derived Feature Analysis

Stephanie: Image Data Analysis (Convolutional Neural Network), Results Summary, Room for Improvement + Future Research

GitHub Repository

CLICK HERE