Presentation Title

Seedling Project - PGAIML

Date: June 21, 2024

Robert Mullins

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- EDA Results
- Data Preprocessing
- Model Performance Summary
- Conclusion
- Appendix

Executive Summary

Final Model had the following results:

- Accuracy: 0.80
- Macro avg: Precision (0.79), Recall (0.75), F1-score (0.76)
- Weighted avg: Precision (0.80), Recall (0.80), F1-score (0.79)

Well-performing classes:

- Charlock
- Fat Hen
- Loose Silky-bent
- Scentless Mayweed
- Small-flowered Cranesbill

Business Problem Overview and Solution Approach

- Please define the problem:
  - We are looking for a way that Data Science can help the field of agriculture as they are still reliant predominately on manual labor. The idea of modernizing here is to find a way that a convolutional neural network may help in identifying/classifying plant seedlings.
  - This work could aid in identifying or sorting between seedlings we would like to have, over seedlings we would like to get rid of.
- Use Cases:
  - 1. Crawler Application: We could make use of a 'crawler' outfitted with GPS, and input parameters of a field. Armed with several cameras pointing down, the crawler could slowly move over hundreds of acres planted with soybean in the space of a weekend, and continuously take in images of the ground. Adding specialized CNN models could make quicker work of a field and advise agricultural caretakers when to spray (if necessary) certain herbicides to deal with weed invasions.

Business Problem Overview and Solution Approach

- Use Cases Continued:
  - 2. Drone Application: The world of Drones has exploded offering hundreds of job opportunities. Unmaned drones can be outfitted with downward-facing cameras to film
  - 3. Other Use Cases: Additionally other use cases could be used for the identification of insect pests, water utilization, and plant development studies to identify optimum harvesting times.

Business Problem Overview and Solution Approach

- The current List of seedlings we would like to identify is as follows;

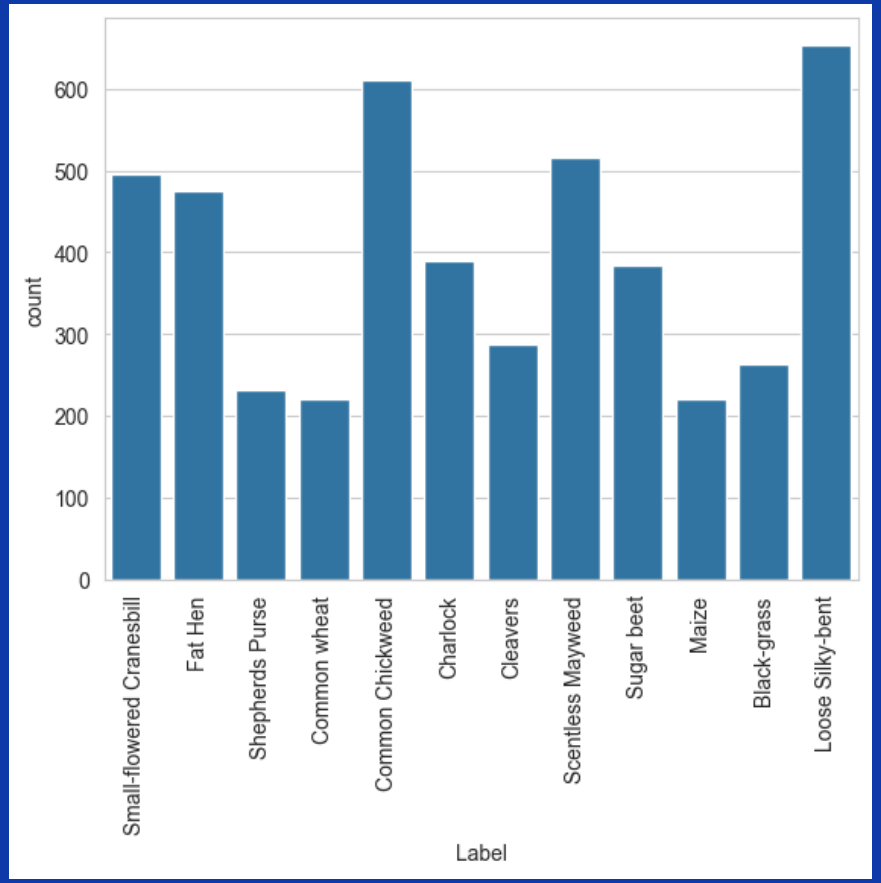
  Black-grass
  Charlock
  Cleavers
  Common Chickweed
  Common Wheat
  Fat Hen
  Loose Silky-bent
  Maize
  Scentless Mayweed
  Shepherds Purse
  Small-flowered Cranesbill
  Sugar beet

Business Problem Overview and Solution Approach

- Please mention the solution approach/methodology
- The approach to this problem is to use a base model with sequential.
- The secondary model is to utilize convelutional neural networks with augmentation to get better reliability. Which the graphs presented clearly show.
- We also want to identify what seedlings are readily trainable.

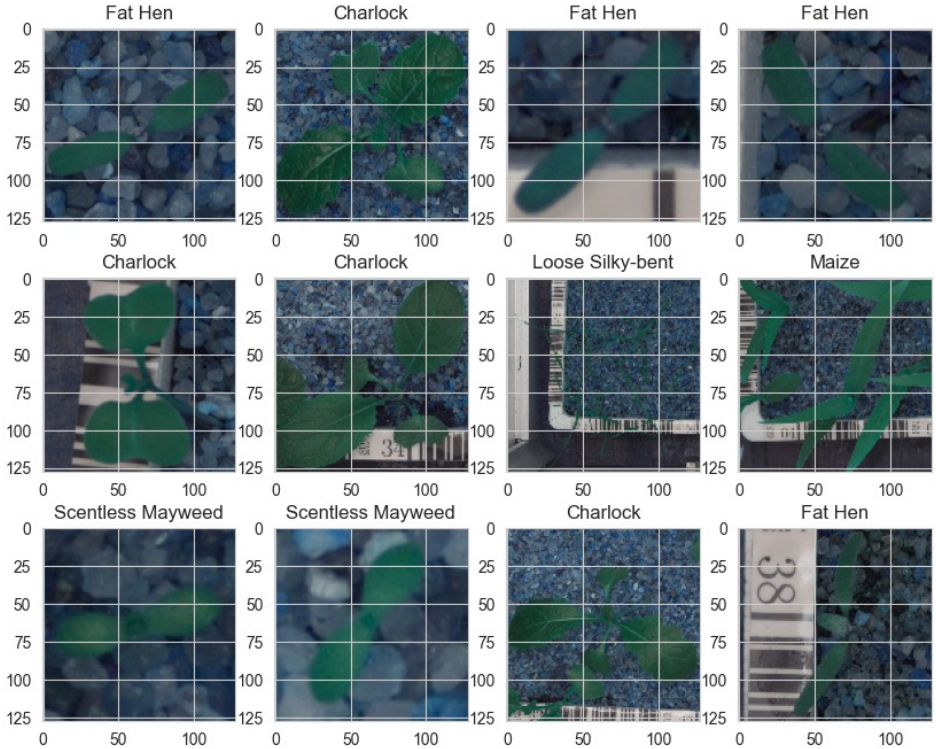
EDA Results

- Our data comes in the form of images pulled from the images.npy file.
- There are a number of different plants for classification.
- While Loose Silky-bent seems to comprise the most it could be safe to assume that we would want a more equal distribution across plant species. Especially where data is quite a bit smaller, Shepherds Purse and Common Wheat.



EDA Results

- Current Image of Seedlings:



There are a number of different seedling types with a variety of leaf structures.
Some exhibit grassy characteristics where as others seem to be more ‘broadleaf’.

Data Preprocessing

- Resizing Images

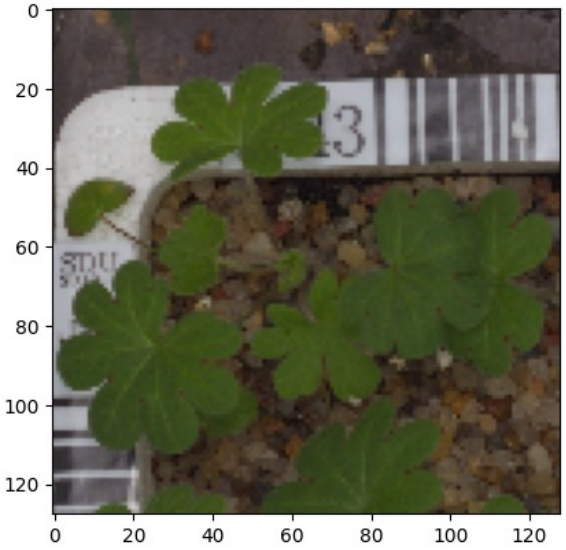


Image Size at 128 X 128

After Image Resizing

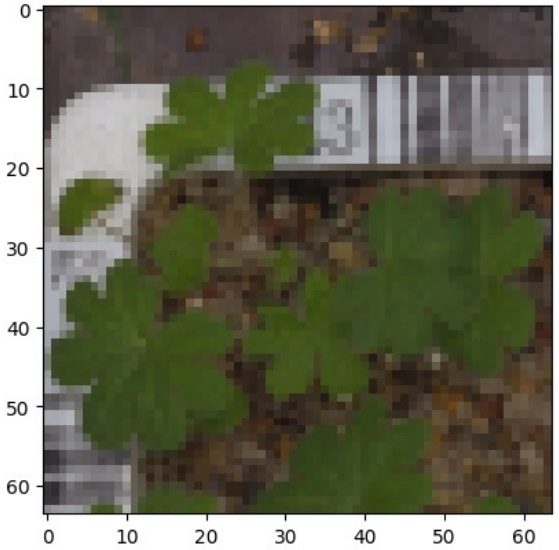


Image Size at 64 X 64

Data Preprocessing

- Encoding the target class: Was performed by utilizing the `LabelBinarizer()` class. The data as follows.

```
enc = LabelBinarizer()                                          #
y_train_encoded = enc.fit_transform(y_train)          # Complete
y_val_encoded=enc.transform(y_val)                    # Complete
y_test_encoded=enc.transform(y_test)                  # Complete
```

```
y_train_encoded.shape,y_val_encoded.shape,y_test_encoded.shape
```

```
((3847, 12), (428, 12), (475, 12))
```

Data Normalization

Data shape explains the training set, validation set, and test set, specifically for the encoded target variables (Y). Break down is as follows

1.Y_train_encoded.shape: (3847, 12)

1. Training set
2. 3847 samples
3. Each sample has 12 features or dimensions

2.Y_val_encoded.shape: (428, 12)

1. Validation set
2. 428 samples
3. Each sample has 12 features or dimensions

3.Y_test_encoded.shape: (475, 12)

1. Test set
2. 475 samples
3. Each sample has 12 features or dimensions

Data Preprocessing

Data Normalization

```
# Complete the code to normalize the image pixels of t
X_train_normalized = X_train.astype('float32')/255.0
X_val_normalized = X_val.astype('float32')/255.0
X_test_normalized = X_test.astype('float32')/255.0
```

Normalizing the data by dividing by 255 to get images to Be between 0 and 1. Speeds processing time.

Model Performance Summary

- Overview of model and its parameters
- Summary of the final model for prediction
- Summary of key performance metrics for training and test data in tabular format for comparison

***Note**: You can use more than one slide if needed*

Model Performance Summary

- Overview of model and its parameters
- Summary of the final model for prediction
- Summary of key performance metrics for training and test data in tabular format for comparison

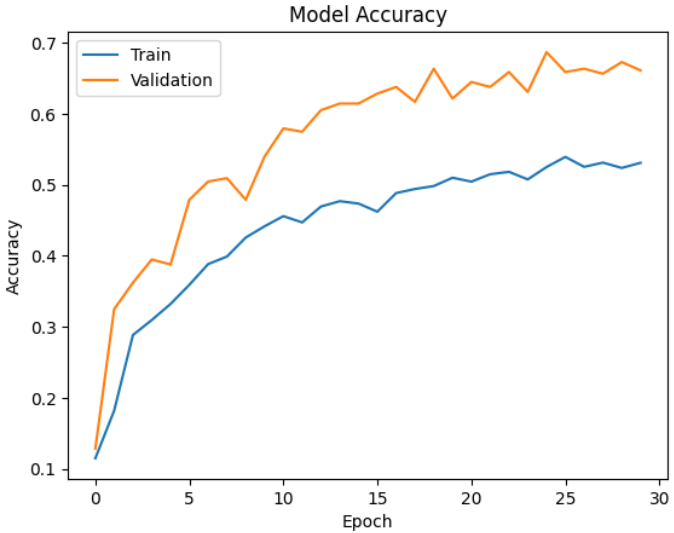
***Note**: You can use more than one slide if needed*

Model Performance Summary

- Model 1 → Sequential

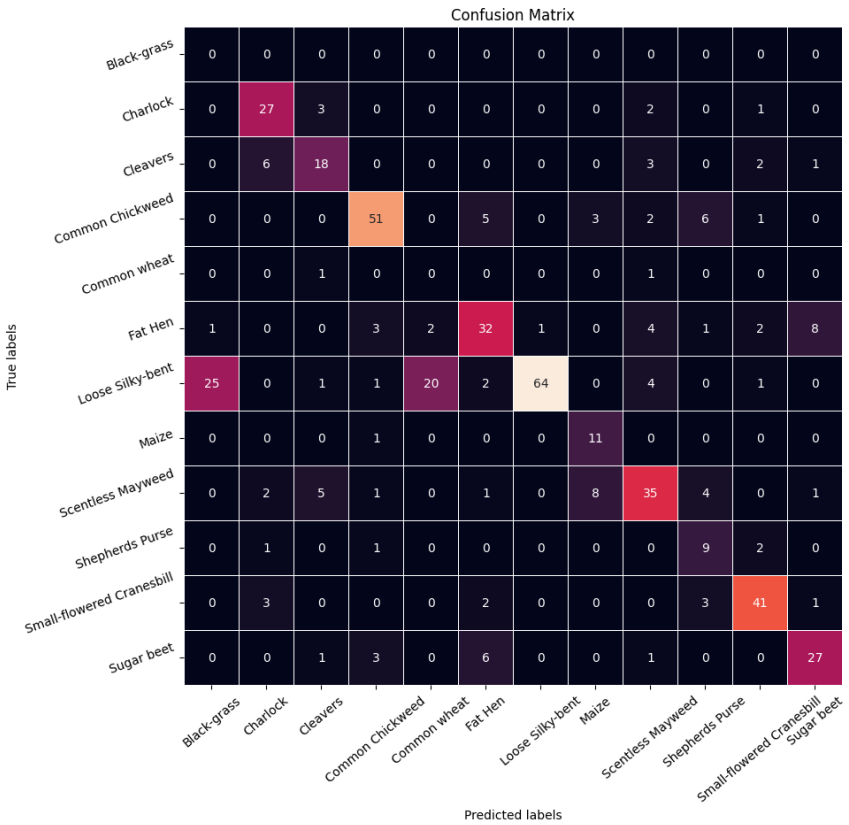
```
accuracy = model1.evaluate(X_test_normalized, y_test_encoded, verbose=2)
```

```
15/15 - 0s - loss: 1.0461 - accuracy: 0.6632 - 189ms/epoch - 13ms/step
```



- Final Accuracy: By the end of training (epoch 30), the model achieves: Training accuracy of about 55%
- Validation accuracy of about 67%
- Overfitting: There's no clear sign of overfitting, as the validation accuracy remains higher than the training accuracy throughout the training process.
- Potential Issues: The fact that validation accuracy is consistently higher than training accuracy might indicate some peculiarities in the data split or potential data leakage. This pattern is atypical and might warrant further investigation.

Model Performance Summary



- Off-diagonal elements represent misclassifications.
- Class-specific observations:Loose Silky-bent is most accurately classified (64 correct predictions).
- Common Chickweed also performs well (51 correct predictions).
- Black-grass performs poorly (0 correct predictions, often misclassified as Loose Silky-bent).
- Common wheat also performs poorly (0 correct predictions).
- Common misclassifications:Loose Silky-bent is often misclassified as Black-grass (25 instances).
- Fat Hen is sometimes misclassified as Sugar beet (8 instances).
- Cleavers is sometimes confused with Charlock (6 instances) and Scentless Mayweed (5 instances).
- Overall accuracy:While not perfect, the model seems to perform reasonably well for most classes, with some notable exceptions.
- Imbalanced classes:Some classes have many more samples than others (e.g., Loose Silky-bent vs. Common wheat), which might affect the model's performance.
- Potential improvements:The model could benefit from focusing on improving its ability to distinguish between commonly confused classes, especially for poorly performing classes like Black-grass and Common wheat.

Model Performance Summary

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.69	0.82	0.75	33
2	0.62	0.60	0.61	30
3	0.84	0.75	0.79	68
4	0.00	0.00	0.00	2
5	0.67	0.59	0.63	54
6	0.98	0.54	0.70	118
7	0.50	0.92	0.65	12
8	0.67	0.61	0.64	57
9	0.39	0.69	0.50	13
10	0.82	0.82	0.82	50
11	0.71	0.71	0.71	38
accuracy			0.66	475
macro avg	0.57	0.59	0.57	475
weighted avg	0.77	0.66	0.70	475

Overall Performance:
Accuracy: 0.66 (66% of all predictions were correct)

Class-specific Observations:
1) Class 0: Performs poorly with 0.00 for all metrics
2) Class 6: Highest precision (0.98) but lower recall (0.54)
3) Class 7: Highest recall (0.92) but lower precision (0.50)
4) Class 10: Best balanced performance with 0.82 for all metrics

Imbalanced Classes:
Support column shows varying sample sizes (e.g., 118 for class 6, only 2 for class 4). This imbalance likely affects the model's performance across classes.

Overall Model Performance:
Weighted avg F1-score of 0.70 suggests moderate performance. There's room for improvement, especially for poorly performing classes.

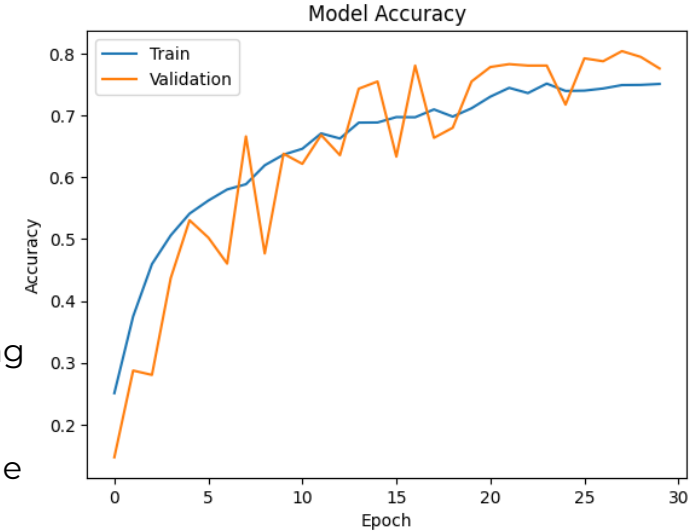
Potential Issues:
Class 0 and 4 have extremely poor performance, requiring investigation. Some classes show high precision but low recall (e.g., class 6) or vice versa (e.g., class 7)

Model Performance Summary

- Model 1 → Sequential

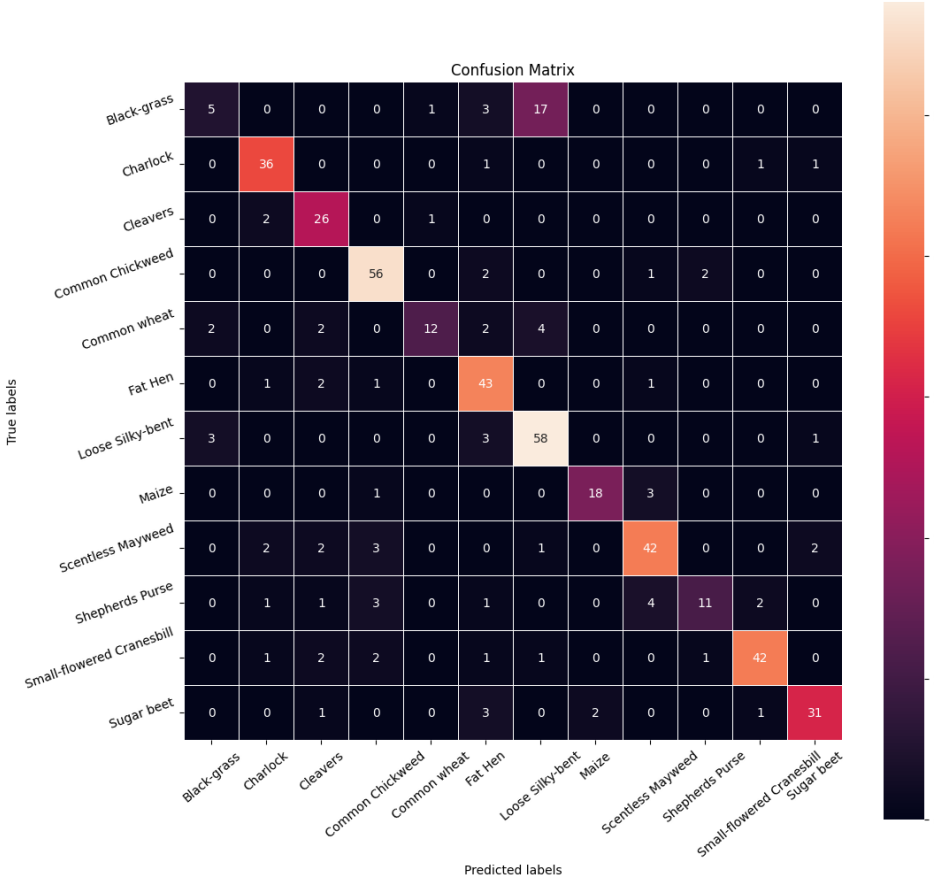
```
accuracy = model2.evaluate(X_test_normalized, y_test_encoded, verbose=2)
```

```
15/15 - 0s - loss: 0.6790 - accuracy: 0.8000 - 113ms/epoch - 8ms/step
```



- Learning Progress: Both lines show an overall upward trend, indicating that the model's accuracy improves as training progresses.
- Fluctuations: The validation accuracy (orange line) shows more fluctuations compared to the training accuracy, which is common due to the smaller size of validation sets.
- Convergence: The curves start to level off towards the later epochs, suggesting the model may be approaching convergence.
- Final Accuracy: By the end of training (epoch 30), both the training and validation accuracies are around 75-80%.
- Overfitting: There's no clear sign of overfitting, as the validation accuracy generally keeps pace with or slightly exceeds the training accuracy.
- Model Behavior: The model shows healthy learning behavior, with validation accuracy closely tracking or slightly exceeding training accuracy, indicating good generalization.

Model Performance Summary



Class-specific observations:

- Loose Silky-bent has the highest number of correct predictions (58).
- Common Chickweed also performs well (56 correct predictions).
- Black-grass performs poorly (only 5 correct predictions, often misclassified as Loose Silky-bent).
- Common wheat has relatively low accuracy (12 correct out of 22 total).

Common misclassifications:

- Black-grass is often misclassified as Loose Silky-bent (17 instances).
- Some confusion between Scentless Mayweed and Shepherds Purse.
- Fat Hen is occasionally misclassified as other classes.

Well-performing classes:

- Charlock, Fat Hen, Loose Silky-bent, Scentless Mayweed, and Small-flowered Cranesbill all show good performance with high numbers on the diagonal.

Poorly-performing classes:

- Black-grass and Common wheat show the most room for improvement.

Class balance: Some classes have more samples than others (e.g., Loose Silky-bent vs. Shepherds Purse), which might affect the model's performance.

Overall accuracy: While not perfect, the model seems to perform reasonably well for most classes, with a few exceptions.

Model Performance Summary

	precision	recall	f1-score	support
0	0.50	0.19	0.28	26
1	0.84	0.92	0.88	39
2	0.72	0.90	0.80	29
3	0.85	0.92	0.88	61
4	0.86	0.55	0.67	22
5	0.73	0.90	0.80	48
6	0.72	0.89	0.79	65
7	0.90	0.82	0.86	22
8	0.82	0.81	0.82	52
9	0.79	0.48	0.59	23
10	0.91	0.84	0.87	50
11	0.89	0.82	0.85	38
accuracy			0.80	475
macro avg	0.79	0.75	0.76	475
weighted avg	0.80	0.80	0.79	475

- Overall Performance: At the bottom, there are summary statistics: Accuracy: 0.80
- Macro avg: Precision (0.79), Recall (0.75), F1-score (0.76)
- Weighted avg: Precision (0.80), Recall (0.80), F1-score (0.79)
- Class-wise Performance:Best performing classes: 1 and 3 (F1-score: 0.88)
- Worst performing class: 7 (F1-score: 0.59)
- Highest precision: Class 10 (0.91)
- Highest recall: Classes 1 and 3 (0.92)
- Support:Total instances: 475
- Largest class: Class 6 (65 instances)
- Smallest class: Class 4 (22 instances)
- Observations:The model performs well on most classes, with F1-scores above 0.80 for many.
- There's some class imbalance, as evidenced by the varying support numbers.
- The overall accuracy of 0.80 suggests good general performance.

Conclusion

CNN Model with augmentation (Model 2) performs quite a bit better than the base sequential model (Model 1) across all dimensions.

APPENDIX

Data Background and Contents

- Additional Notes:

I first attempted this project by setting up a PyCharm project utilizing a virtual environment (.venv) to keep all my installations via 'pip' separate from the rest of my local system. I wanted to attempt this in a local environment. While it did indeed work configuring the IDE, and making use of a GitHub repository posed some challenges.

1) Learning to use the IDE (PyCharm) and setting it up with both Jupyter AND a *.venv was problematic using the IDE's internal capabilities. It required starting a new project and utilizing onboard Python as the interpreter, and THEN adding the Jupyter capabilities for the .venv to work.

2) Setting up to GitHub was an issue and I could not 'pull' the project, but instead had to initialize the project from the local environment to push to a new GitHub project. These two steps required about four days of trying different things before I could Find a set-up that could work.

Data Background and Contents

- Additional Notes:

Due to time-constraints I eventually had to abandoned the ‘local attempt’ part of the project, however, I was able to connect my Google Colab to GitHub and pull everything I had from the repository.

Additionally there is a semi-complete ‘readmy.md’ in the repository as well, that outlines the basics of the project.

I have the GitHub project still working and made for public use, so if anyone needs access be sure to let me know.

Final note, I took additional time and made this PowerPoint compliant with ’Accessibility’ standards, particularly with the images added.

Great Learning

Happy Learning !