Target Shift and Model Performance

We used EvidentlyAI to find out the concept shift in our image dataset of plant species, and we also evaluated classification model performance on a multi-class classification problem. Before diving deep into either of these shifts, it's essential to prepare similar structured reference and production datasets in pandas for the input. In the case of the output, we have an option to choose between either a visual dashboard or a JSON profile containing all statistical tests.

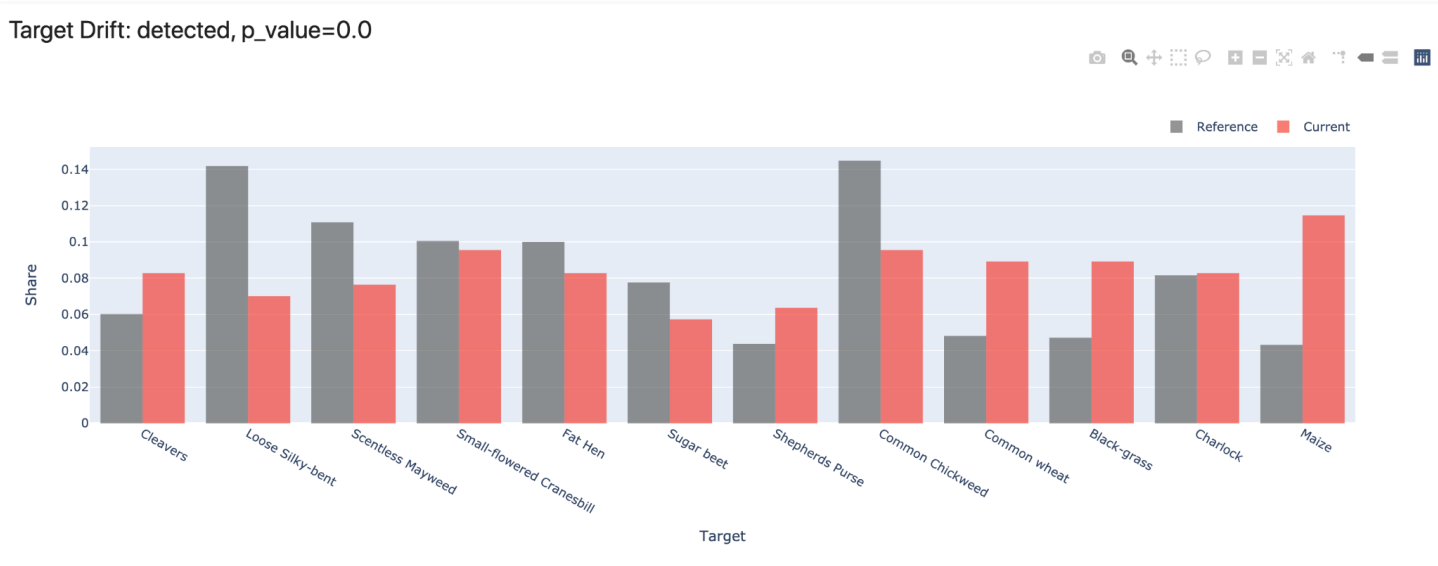
1 - Categorical Target drift

To evaluate the concept shift for the categorical variable, EvidentlyAI uses chi-square tests which are applied to assess whether a variable is coming from a similar distribution or not. In terms of our experience with the tool for concept shift, we admired the simplicity of the code needed to get the result. The output for the dashboard contains p values of the test, along with proportions for reference and the current dataset with target drift once you hover on the graphs. These graphs are created from the plotly python library, and you can read more about the code in their Github repository. This feature could be greatly valuable to address a drop in model performance.

Code Snippet:

```
#understanding the target shift
np.seterr(divide='ignore', invalid='ignore')
seedling_report = Dashboard(tabs=[CatTargetDriftTab])
seedling_report.calculate(pd.DataFrame(training_img_list.iloc[:,1]), pd.DataFrame(testing_img_list.iloc[:,1]), column_mapping = None)
seedling_report.save("reports/seedling_report.html")
```

Output:



2- Model Performance

The Classification model performance feature supports both binary and multi-class classification. To run this report, we need input features with actual and predicted target variables. The tool currently supports tabular data intending to add more features for it. Moreover, the team also plans to extend the support on image and text data as well. Therefore, for our model, we were able to work metrics only for the labels. We got model metrics like F1 Score, Precision/Recall, and Accuracy, along with class representation, Confusion metrics, and quality metrics for each class. In our specific case of multi-class classification, it would be effective to also have an option for micro and macro f1 scores, beta metric to evaluate the model performance. Similar to the case above, the interactive dashboard is created using plotly. A report like this would be extremely helpful to analyze the model run, run A/B tests, find appropriate thresholds, and debug

Code Snippet:

```
reference=pd.DataFrame(columns=['target','prediction'])

production=pd.DataFrame(columns=['target','prediction'])


reference['target'] = training_img_list['target']
reference['prediction'] = target_list

production['target'] = testing_img_list['target']
production['prediction'] = predictions_list
```

```
data_dict = {}

data_dict['target'] = 'target'
data_dict['prediction'] = 'prediction'
```

```
classification_performance_report = Dashboard(tabs=[ClassificationPerformanceTab])
classification_performance_report.calculate(reference, production, column_mapping=data_dict)
classification_performance_report.save("reports/classification_performance_report.html")
```

Output:

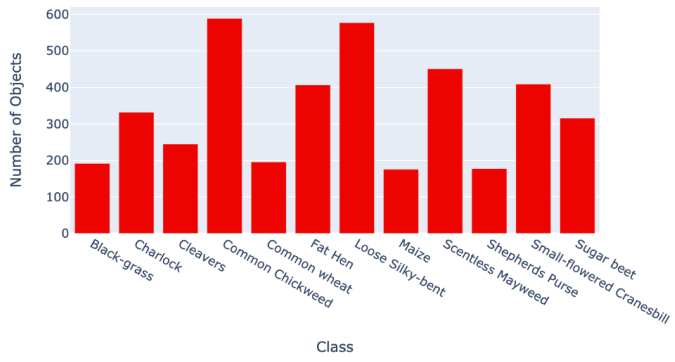
Reference: Model Quality With Macro-average Metrics

0.437	0.406	0.344	0.325
Accuracy	Precision	Recall	F1

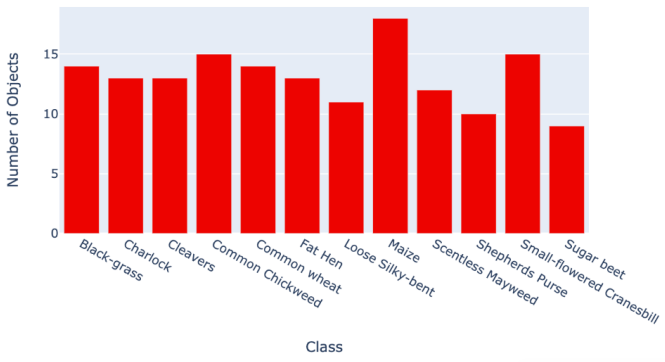
Current: Model Quality With Macro-average Metrics

0.325	0.444	0.334	0.289
Accuracy	Precision	Recall	F1

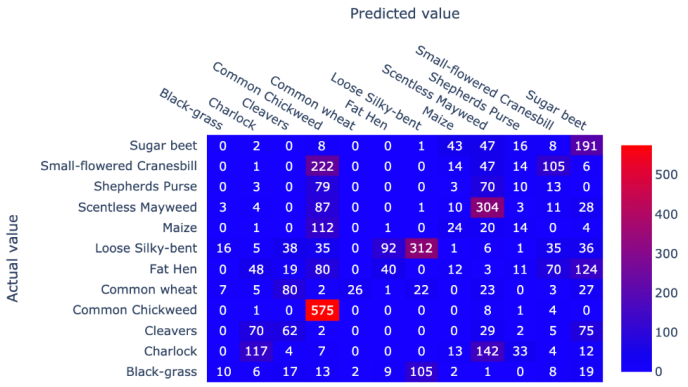
Reference: Class Representation



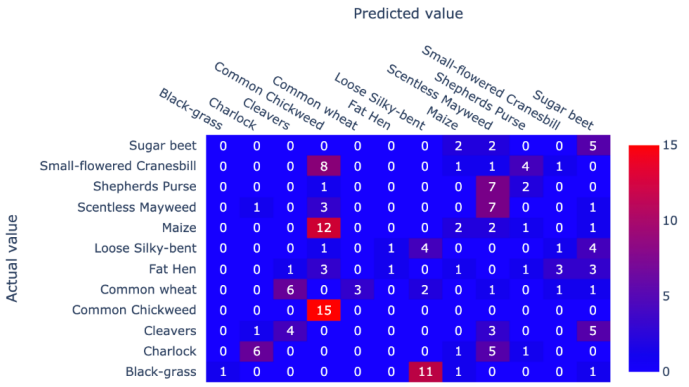
Current: Class Representation



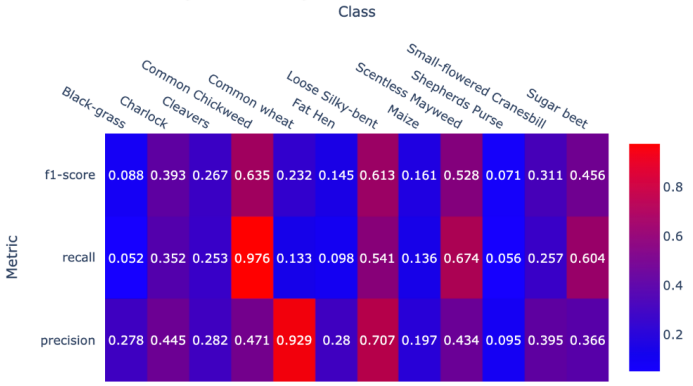
Reference: Confusion Matrix



Current: Confusion Matrix



Reference: Quality Metrics by Class



Current: Quality Metrics by Class