Predicting Plant Traits with Boosting, ViTs and Model Stacking

Srihari Vishnu
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1
svishnu@uwaterloo.ca
report due: August 12

Abstract

This project aims to predict a given set of plant properties (such as plant height, seed mass etc.) from citizen science plant photographs, and their corresponding ancillary information (such as local climate, and soil information). The goal was to achieve a baseline score of 0.43902. By utilizing feature engineering, Pretrained Vision Transformers (ViTs), gradient boosting and combining various models, we were able to obtain an R2 score of 0.54968 on the public dataset. The code can be found here: $https : //github.com/sriharivishnu/CS480_{P}roject$.

1 Introduction

For image analysis tasks, CNNs and ViTs (as described in Dosovitskiy et al. 2020) are state-of-the-art techniques for building high performing models. For tabular data, Gradient Boosting algorithms such as XGBoost (Chen and Guestrin 2016) and Catboost (Prokhorenkova et al. 2017) are well-known for achieving top scores in Kaggle machine learning competitions. However, in the given dataset, we have both images as well as tabular data. To create an accurate model, it was desired to obtain image embeddings that could be easily combined with tabular data.

Due to computational resource restrictions, generic pretrained models that output image embeddings to be used for downstream tasks were preferred over training a model from scratch. Out of the models that were tested (CLIP, ResNet Transfer Learning, DinoV2), DinoV2 (Oquab et al. 2024) far outperformed the others and was selected to obtain image embeddings.

The goal was to improve the accuracy of the model by training various regressors on the whole dataset, and combining the predictions using a meta-regressor. This procedure is known as Model Stacking. As seen in Figure 1, each model outputs various predictions for each data point, and by leveraging this diversity, we show that it is possible to achieve far higher scores than what is possible with each individual model.

2 Related Works

Vision Transformers (DinoV2): The notion of using models that require minimal supervision to produce image embeddings is not new. As mentioned in the work done by Oquab et al. 2024, DinoV2 performs extremely well with images it has not seen before, with a simple linear layer is all that is needed for downstream learning tasks. Although the work also mentions potentially finetuning DinoV2 to achieve even higher accuracy, we did not pursue this approach due to computational restrictions. The work itself is focused on images, whereas we have images complemented with tabular data as well.

Submitted to CS480/680, 2024 Spring. Do not distribute.

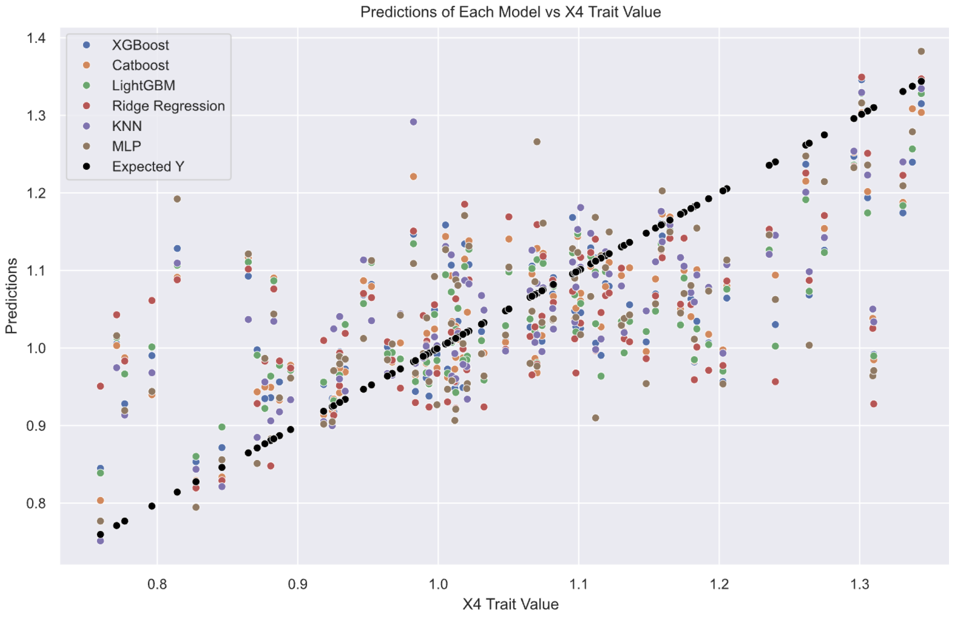

Figure 1: Diverse set of models differ in their predictions for Trait X4

Model Stacking: The process of stacking models involves training layers of models, and using the predictions of the models to train a meta-regressor on top of it. The idea of stacking various was inspired from the work done by Daza et al. 2023, which utilized model stacking to detect diabetes given a tabular dataset. In our work, we utilize popular algorithms such as XGBoost and Catboost as our base models, as well as weaker models such as ridge regression .

3 Main Results

There are 2 phases in training the final model to obtain predictions:

1. *Base Regressors* (B): Train each base regressor $r \in B$ on the entire dataset. These classifiers can be arbitrary, whose internals can be treated as a black box.
2. *Meta Regressor* (M): Train a meta regressor M, who takes the predictions of each $r \in B$, and outputs a final prediction. Care has to be taken to train the meta regressor on samples that the base regressors have not seen yet.

The goal of model stacking is to obtain a meta classifier M that outperforms any individual $r \in B$. We describe the algorithm given by Algorithm 1 in detail below.

3.1 Data Preprocessing & Feature Engineering

Both the features and the labels are normalized utilizing sklearn's $StandardScaler$. The features are scaled for algorithms such as KNN & linear regression, where data normalization improves the accuracy of the model. The labels are scaled to fix convergence issues with the MLP model (since we utilize average MSE over the 6 target labels).

In addition, 1000 polynomial features of at most degree 2 were added to the CSV by randomly choosing a subset of sklearn's $PolynomialFeatures$ output, as described in this article. This offered slight improvements in the final $R2$ score (≈ 0.005). However, only the gradient boosting regressors (described later) benefited from these features, thus only those models utilize them.

Images were transformed according to DinoV2 preprocessing pipelines (channels normalized to mean $[0.485, 0.456, 0.406]$ and images resized to 224×224 described $here$). These images were then fed into the DinoV2 model (specifically, dinov2_vitg14_reg) to generate embeddings of size 1536 for each image.

Thus, the resulting number of features is: $163 + 1536\ (+1000) = 1699\ (2699)$.

```
Algorithm 1: Model Stacking of Regressors with Cross-Validation
Input: $X, Y, X_{test}$
$Y_{test} \leftarrow$ zeros($len(X_{test}), 6$)
for $k = 1, 2, \ldots K$ do
    $X_{train}, Y_{train}, X_{val}, Y_{val} \leftarrow$ random_split($X, Y$)
    $X_{train}, Y_{train} \leftarrow$ Preprocess($X_{train}$), Preprocess($Y_{train}$)
    $X_{val}, Y_{val} \leftarrow$ Preprocess($X_{val}$), Preprocess($Y_{val}$)
    $X_{meta-train} \leftarrow []$                // will have shape $(len(X_{val}), |B| \times 6)$
    $X_{meta-test} \leftarrow []$                 // will have shape $(len(X_{test}), |B| \times 6)$
    Initialize $\forall r \in B, M$
    for $r \in B$ do
        train(r, $X_{train}$, $Y_{train}$)
        Report score(r, $X_{val}$, $Y_{val}$)
        $X_{meta-train} \leftarrow X_{meta-train} \bigcup$ predict($X_{val}$)       // column-wise union
        $X_{meta-test} \leftarrow X_{meta-test} \bigcup$ predict($X_{test}$)        // column-wise union
    train($M, X_{meta-train}, Y_{val}$)
    Report score($M, X_{meta-train}, Y_{val}$)
    $Y_{test} \leftarrow Y_{test} +$ predict($M, X_{meta-test}$)$/K$            // Result is averaged
Return $Y_{test}$
```

3.2 Training Base Regressors

A train-test split of $0.9/0.1$ was found to be optimal. Each base regressor is trained to output each of the 6 plant traits. In our experiment, we have $|B| = 6$ and we describe the chosen base regressors below.

3.2.1 Gradient Boosted Trees: XGBoost, CatBoost, Light GBM

This family of models rely on gradient boosting of weak decision trees in order to learn. Although the algorithms are quite similar to each other, there were some subtle performance gains in R2 score in utilizing all 3. Catboost was the highest individual performer; removing any of these regressors results in marginal decreases in final R2 scores (≈ 0.005), while removing all 3 decreases the final R2 score by (≈ 0.035).

These 3 models had their hyperparameters tuned utilizing a bayesian optimization framework called Optuna.

3.2.2 K-Nearest Neighbours

Another interesting find was the strong performance of the classical K-Nearest Neighbours algorithm. Optimal performance of this regressor was attained using Manhattan distance, values weighted by distance, and $k = 7$. The removal of this base classifier results in ≈ 0.015 decrease in R2 score.

3.2.3 Ridge Regression

An interesting find was that even using simple linear regression gave subtle performance gains. The addition of regularized linear regression provided gains of ≈ 0.03 in $R2$ score when first introduced, however, its efficacy declined as base regressors were improved. Nevertheless, it still achieves 0.002 improvement in R2 score.

3.2.4 Multilayer Perceptron (MLP)

The tendency for the MLP to overfit was very high, so a lower number of parameters was required to achieve strong performance. After some trials, it was determined that 2 hidden layers of size 1024 and 256 performed the best. Removing the MLP base model would result in a decrease of ≈ 0.008.

3.3 Training Meta Classifier

3 models were tested as a meta classifier: A linear model (Lasso with $\alpha = 0.0006$), DecisionTreeRegressor, and a simple Neural Network. Out of the 3, the linear model outperformed the other two, providing an $R2$ result of 0.5434 (vs 0.5134).

3.4 Experimental Results

One of the issues of Model Stacking is the availability of data. Data is required to train the base classifiers, along with the meta classifier. To ensure accurate, representative results, cross validation strategy is used, in which output values are computed as the average across the rounds.

The results of running the model stacking algorithm are shown in Table 1. The embeddings took approximately 1.5 hours to produce on 1 $A100$ GPU with 40 GB of RAM, and the main training loop as described in Algorithm 1 with $K = 5$ rounds took 15 hours on an M1 Max Macbook Pro 2021.

In all rounds, the stacked regressor outperforms any individual base regressor by a large margin.

Table 1: Model Comparison Between Validation Rounds

Model	Validation Round R2 Score					
	1	2	3	4	5	Average
XGBoost	0.4791	0.4757	0.4669	0.4625	0.4852	0.4739
CatBoost	0.5038	0.5025	0.4984	0.4884	0.5072	0.5001
LightGBM	0.4785	0.4744	0.4692	0.4623	0.4874	0.4744
Ridge Regression	0.4067	0.4059	0.3963	0.3919	0.4121	0.4026
KNN	0.4851	0.4911	0.4801	0.4753	0.4923	0.4848
MLP (1024x256)	0.4630	0.4652	0.4451	0.4389	0.4572	0.4539
Stacked	**0.5434**	**0.5392**	**0.5300**	**0.5227**	**0.5460**	**0.5363**

4 Conclusion

The results show that it is possible to combine several base learners to achieve a meta regressor that far outperforms each individual model. This shows the positive benefits that the diversity of different models has on improving the accuracy of predictions. Compared to many boosting algorithms, model stacking allows training various heterogenous models (different frameworks, etc.) in parallel.

In the future, it may be valuable to explore meta learners that use the context of the features to select which model would be the best to use. Further, it may also be desirable to explore whether fine-tuning DinoV2 would improve accuracy scores.

Acknowledgement

Thank you to Yaoliang Yu for the advice: "When you can't choose between models, choose both"! Further, thank you to Jeannie Quach for helping brainstorm ideas.

References

Brownlee, J. (2020). "How to Use Polynomial Feature Transforms for Machine Learning". URL: https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/ (visited on 08/09/2024).

– (2021). "Stacking Ensemble Machine Learning With Python". URL: https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/ (visited on 08/03/2024).

Chen, T. and C. Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". URL: https://arxiv.org/abs/1603.02754 (visited on 08/01/2024).

Daza, A., C. F. P. Sanchez, G. Apaza-Perez, J. Pinto, and K. Z. Ramos (2023). "Stacking ensemble approach to diagnosing the disease of diabetes". URL: https://www.sciencedirect.com/science/article/pii/S2352914823002733 (visited on 07/25/2024).

Dosovitskiy, A. et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". URL: https://arxiv.org/abs/2010.11929 (visited on 07/24/2024).

Oquab, M. et al. (2024). "DINOv2: Learning Robust Visual Features without Supervision". URL: https://arxiv.org/abs/2304.07193 (visited on 07/25/2024).

Prokhorenkova, L., G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin (2017). "CatBoost: unbiased boosting with categorical features". URL: https://arxiv.org/abs/1706.09516 (visited on 08/04/2024).