Derek Eppinger

Logan Groth

Efficient Model Selection Using Dataset Metadata

Table of Contents

Introduction, Background, and Related Work

Deep Learning has become the superior solution for inferencing in domains such as audio, visual, and text largely due to ever-increasing availability in computational resources as well as large, labeled datasets. However, even when given a high volume of data in the tabular format, traditional machine learning algorithms such as XGBoost and Random Forest are frequently observed outperforming deep learning methods due to tabular data's structured and well-defined nature [1]. Deep learning's strength lies in its ability to learn the unstructured features within data and most tabular datasets already represent extracted features, so oftentimes there isn't an advantage to using deep learning. Additionally, in domains such as vision or speech, since the data is completely homogenous in format, it is easier for deep learning models to discover a greater correlation among the features in the data compared to the features found in tabular data which include a diversity of both numerical and categorical features, making it more challenging for deep learning models to process and learn effectively [2].

However, there still is a need to discover how best to leverage deep learning neural networks for tabular data. Tabular data is the most commonly used form of data, and is ubiquitous in many different domains such as medical diagnosis, climate science, financial applications, user recommendation systems, cyber security, and many more. If deep learning performance is maximized in these industries, there will be tremendous benefits for both industry, research, and end users alike.

Current research on deep learning for tabular data recognizes this problem deeply. Many research papers attempt to survey the different state-of-the-art deep learning algorithms available and compare their performance to traditional machine

learning algorithms, exploring why the ML methods may often be outperforming DL methods and how to extract optimal performance from the DL methods whether through data preprocessing or hyperparameter tuning methods [1, 2, 3]. According to Grinsztajn, Oyallon, & Varoquaux [1], the best tabular data methods usually share two properties. One is that they are ensemble methods, which are less prone to overfitting, and that for the ensemble method the weak learner is a decision tree. In contrast to decision trees, MLP's are rotationally invariant, meaning they perform the exact same after a linear transformation has been applied to the input data. Moreover, [1] also reveals that any rotationally invariant learning procedure shows that any rotationally invariant learning procedure has a worst-case sample complexity that grows at least linearly in the number of irrelevant features.

A majority of research also focuses on creating new deep learning architectures optimized for tabular data, often taking advantage of current innovations in deep learning intended for domains such as text or vision. Google's TabNet architecture uses sequential attention to choose which features to reason from at each instance of data, enabling interpretability and efficient learning [4]. Transformers have also been adapted for use in tabular data including the FT-Transformer (Feature Tokenizer + Transformer) architecture. This model works by transforming all features both numerical and categorical to embeddings and applies a stack of Transformer layers to the embeddings. Therefore each transformer layer will operate on the feature level of one object. It then passes this CLS token to a final linear layer. One flaw reported of FT-Transformer is that the architecture requires a lot more training resources than architectures such as ResNet, and is oftentimes not as easily scalable to datasets with a large number of features [5]. FT-Transformer is slightly different from another popular model,

TabTransformer, that only passes the categorical features through the transformer and concatenates the raw numerical input with its contextual embeddings to the final linear layer [6]. Both FT-Transformers, TabTransformer, and TabNet have shown to be competitive against ML algorithms by their respective creators.

Even CNNs have been adapted to be used within the tabular domain with the usage of the novel algorithm, IGTD, or, Image Generation for Tabular Data, which transforms one row of data into a grid of pixels. If there were nine features for each row of data, the generated image would be nine pixels large, with each pixel representing its corresponding features' relative values in their brightness and related features being located close together on the generated square [7].

Our project is similar to the existing research in that we are surveying deep learning architectures for tabular data and comparing their performance across different datasets. However, in the existing research there are usually many different architectures being compared over only a few data sets. In contrast, we are only comparing a few architectures over many different datasets. This is because unlike the existing research, we are trying to identify relationships between the characteristics within the datasets and prediction performance, while other research is focused on the characteristics of the algorithms themselves relative to the performance. While we are focused on comparing performance of an algorithm relative to a dataset's size, ratio of categorical to numerical variables, and majority class ratio (for binary classification), current research tends to focus on performance of an algorithm relative to its training time and model complexity, two aspects we are choosing to ignore.

If there is a relationship between aspects of a dataset and algorithm performance, this research could help researchers identify when to use which methodology. We could

also create a machine learning model (albeit a very simple one given our training size) that predicts which ML or DL method will perform best given a dataset's metadata. This model could help save computer scientists time and computational resources in training multiple AI models by helping them decide which model will work best with their current dataset.

Efficient model selection is also another emerging branch of research within deep learning and specifically computer vision, a field where model training can be especially expensive and time consuming. In a paper released by Stanford titled "LOVM: Language-Only Vision Model Selection", Vision-Language Models were able to make classifications on images but only given text-descriptions of the images created by LLMs. These classifications were comparable to ground-truth evaluation data assembled prior, revealing that descriptions of the image data was enough to make an efficient model selection [8]. While our paper does not use LLMs or text descriptions of the data, our research is also trying to achieve efficient model selection given only a small amount of information about the dataset, certainly in the same spirit as the LVOM paper.

Data Collection Methodology

We assembled 22 binary classification datasets from Kaggle of diverse size and use cases including predicting smokers based on health characteristics, predicting credit card approvals, and classifying species of raisins, fruits, and penguins based on their various dimensions.

In the preprocessing phase of each test, for numerical features, the mean was subtracted and they were divided by their standard deviation. For the deep learning

architectures that do not apply their own embedding process for categorical features, categorical features were converted by sklearn's LabelEncoder to an integer on the scale from zero to number of classes-1. We relied on the definition of numerical features versus categorical features taken from UCLA's OARC website, where categorical variables have no intrinsic ordering to the categories. This therefore would also include binary variables. No other data preprocessing or standardization was performed in an attempt to evaluate the algorithms as equally as possible. We also consistently used an 80% split between training data and validation data for each algorithm.

We then ran Random Forest, XGBoost, a Multi-Layer Perceptron (MLP), FT-Transformer, and TabNet on each of the datasets. We choose Random Forest and XGBoost because they are commonly stated as top machine learning performers for tabular data as stated above. We chose FT-Transformer and TabNet as hybrid architectures that both operate very differently from each other, therefore presenting a greater chance of showing different results. We also have an MLP with a single hidden layer as the middle ground between the hybrid deep learning architectures and the machine learning algorithms. It should be noted that well tuned MLP's have still been shown to perform well on tabular data [9].

To represent the deep learning architectures best potential, Python's Optuna package was used to find the best hyperparameters for each architecture. Optuna still requires an input range of hyperparameters, and given our limited time to test every architecture on every dataset, there is a possibility we still may have missed the best combination of hyperparameters for a given dataset and architecture. Each deep learning architecture was trained for ten epochs.

After achieving what we deem to be the best performance possible for a model we recorded the best validation accuracy and validation F1 score and used these as key metrics for how well each model did.

Algorithm Performances

Dataset Name	RF F1	RF Acc	XGB F1	XGB Acc	MLP F1	MLP Acc	FT F1	FT Acc	TabN F1	TabN Acc
Loan Approval Prediction	0.973	0.979	0.978	**0.984**	0.938	0.953	0.978	0.973	0.915	0.932
Heart Disease Health	0.179	0.902	0.097	0.907	0.158	0.908	0.152	**0.910**	0.107	0.908
Palmer Penguin	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	0.977	0.979	1.000	**1.000**
Breast Cancer	0.949	0.962	0.933	0.953	0.982	**0.977**	0.960	0.918	0.814	0.861
Body Signs of Smoking	0.776	**0.831**	0.682	0.759	0.664	0.757	0.713	0.765	0.684	0.756
Income >50K	0.657	0.849	0.635	0.855	0.589	0.844	0.679	**0.860**	0.630	0.843
Credit Card Approval	0.529	0.926	0.150	0.891	1.000	**1.000**	0.069	0.900	0.000	0.888
Bank Marketing Response	0.141	0.840	0.067	0.844	0.397	0.895	0.566	**0.913**	0.534	0.882
Raisin Classification	0.871	0.868	0.871	0.862	0.866	0.870	0.823	0.832	0.881	**0.875**
Rock or Metal	0.796	**0.821**	0.750	0.762	0.786	0.709	0.765	0.805	0.733	0.757
Orange or Grapefruit	0.948	**0.947**	0.937	0.936	0.947	0.945	0.936	0.936	0.930	0.931
Trojan Detection	1.000	**1.000**	1.000	**1.000**	0.816	0.800	1.000	**1.000**	0.991	0.991
Rain Tomorrow	0.628	**0.865**	0.589	0.855	0.558	0.848	0.678	0.857	0.623	0.846
PharynGitis Antigen	0.557	0.660	0.593	0.666	0.725	**0.757**	0.706	0.755	0.611	0.680
Water Quality	0.763	0.954	0.812	0.961	0.975	**0.956**	0.772	0.946	0.617	0.923
Rice Classification	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**
Customer Churn	0.556	0.859	0.547	0.860	0.770	**0.928**	0.828	0.531	0.405	0.839
Airline Satisfaction	0.957	**0.964**	0.933	0.942	0.943	0.955	0.943	0.953	0.612	0.679
Surgical Complication	0.689	**0.866**	0.893	0.744	0.463	0.816	0.685	0.854	0.450	0.810
Titantic Survival	0.804	0.738	0.825	0.751	0.098	0.615	0.844	**0.800**	0.820	0.783
Diabetes Prediction	0.604	0.749	0.616	0.750	0.653	0.753	0.708	0.801	0.597	**0.789**
Stroke Prediction	0.000	0.958	0.008	0.954	0.000	0.946	0.000	**0.961**	0.000	0.946

Figure 1: Model F1 Score and Accuracy over 22 datasets. The best accuracy is bolded and the best f1 score is underlined.

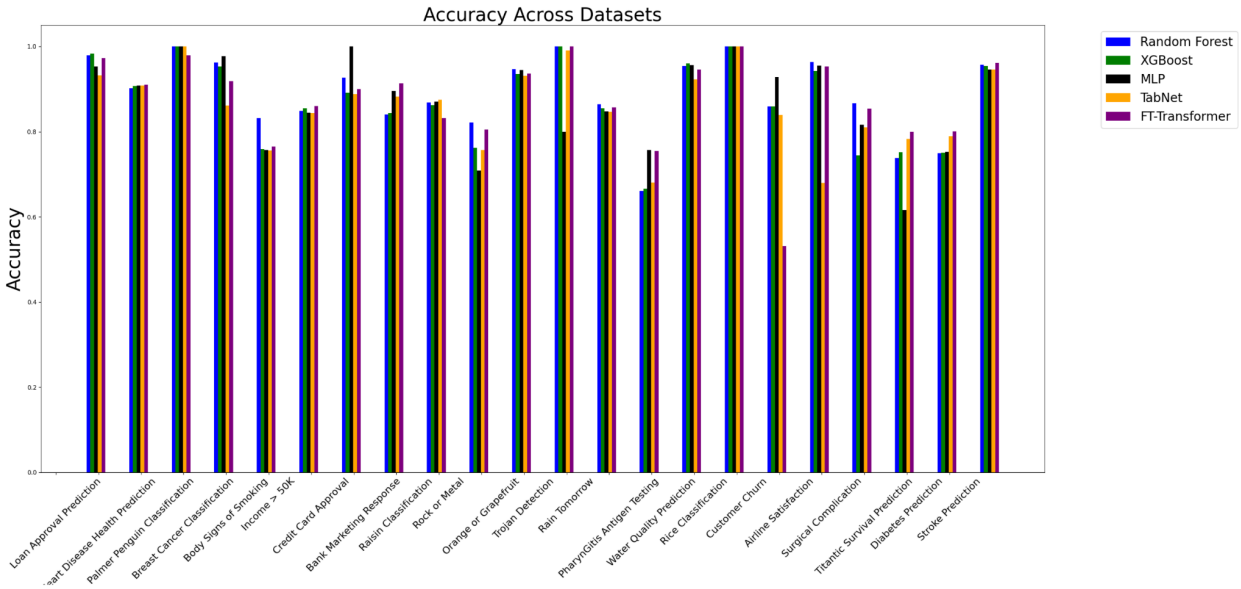

Figure 2: Bar graph displaying model accuracy over 22 datasets

	Random Forest Accuracy	XGBoost Accuracy	MLP Accuracy	FT-Transformer Accuracy	TabNet Accuracy	Random Forest F1 Score	XGBoost F1 Score	MLP F1 Score	FT-Transformer F1 Score	TabNet F1 Score
Training Size	0.302271	0.332882	0.182007	0.097912	0.214261	-0.064533	-0.077225	-0.085454	-0.010845	-0.073667
Majority Class Ratio	0.169182	0.175982	0.390858	0.069131	0.296960	-0.664311	-0.665887	-0.362153	-0.666410	-0.698937
No. of Categorical Features	-0.013443	0.083335	-0.012945	0.042166	0.040526	-0.508007	-0.567824	-0.513787	-0.466875	-0.453351
No. of Continous Features	0.218518	0.149797	-0.256101	0.160677	0.042857	0.365977	0.333703	0.240670	0.321811	0.332637
Total Number of Features	0.129367	0.054950	-0.301384	0.119068	-0.024174	0.200314	0.165125	0.106609	0.151681	0.159296
Average Correlation	0.047414	0.040340	0.090458	0.013517	0.126503	0.353117	0.353643	0.277018	0.277927	0.399379
Ratio of Continous Features to Total Features	0.373959	0.350811	0.091625	0.212179	0.200793	0.741766	0.726735	0.627095	0.715823	0.722839

Figure 3: Correlation Table

As shown with both Fig. 1 and Fig. 2, the algorithms perform similarly for several of the datasets. In fact, we ended up having to remove the results completely for three of the datasets later in the model prediction process as multiple of the algorithms were shown to perform perfectly. In addition, for several other datasets, while there may have been a best performer, the differences in performance could be considered almost negligent, and we could not have attributed the differences in performances to any specific dataset characteristics. Regarding top performers, Random Forest had the top accuracy a total of nine times including ties for first place, MLP and FT-Transformer had the top accuracy seven times, XGBoost four and just one for TabNet. This was consistent with the benchmark set by Borisov et al. [2] in that the machine learning models ultimately performed the best, but inconsistent in the fact that XGBoost was not our top performer as seen in the paper.

In Fig. 3 we can see the correlations between dataset characteristics and algorithm performance. Contrary to our expectations of the deep learning architectures

performing better given more data, the machine learning algorithms saw the biggest correlation between training size and accuracy. As always with correlations, there could have been other factors at play leading to them to perform well on larger datasets. The MLP also saw a uniquely negative correlation between total features and accuracy, similar to the idea mentioned prior that rotationally invariant algorithms do not perform well given more uninformative features [1]. Ratio of Continuous Features to Categorical Features was definitely correlated with algorithm accuracy and strongly correlated with algorithm F1 Score. Unsupringly, the majority class ratio was negatively correlated with F1 score but positively correlated with accuracy as random guesses were more likely to be correct.

Model Prediction and Project Progress

Since the first report we finished our meta-dataset bringing it up to 22 different datasets and adding average correlation as a predictor and total features. Multiple other predictors were considered but in the end we chose to exclude them for the experiment. Given the limited size of our dataset adding too many features could have caused overfitting and because one of our goals is to create an efficient algorithm selection process adding more in depth meta-features may require exploring the data so much that it may not really save any time. As such we have also combined categorical features with binary features. We also created several graphical representations of the relationships between our variables and accuracy/F1 score.

We also created a ranking system for the models. This ranking system utilizes leave one out cross validation (LOOCV) to compare 2 models and the model that was

predicted to be better has its rank incremented by one. One of the consequences of this is that our algorithm could output that multiple algorithms share the same rank as each other. Because of this and the difficulty in interpreting how correct or incorrect our algorithm is we decided to change our methodology.

Instead of ranking them we used LOOCV to compare only two models at a time utilizing Random Forest. We did this leaving each datapoint out once to act as a test set and building the model on the rest of the data. We then predicted the value of the test datapoint and compared that to the actual result. We repeated this ten times for each datapoint and output the accuracy and F1 score. Repeating the process ten times lowered the chance of accuracy/F1 score being high due to pure chance. This process was repeated twice predicting algorithm accuracy and F1 score.

One major change in model prediction that was attempted but deemed not viable was treating similar results as the same. The results seen in fig. 5 and fig. 6 are done in a naive way where the results are compared as which one is better; this does not take into account the variance of each algorithm's results caused by random sampling. Due to this it would have been better to treat results within 2 standard deviations of each other as equal. This was not possible to do and will be further explained in project challenges. This was also attempted using various static cutoffs. While this did occasionally improve results, (note that random guess would be ⅓ not ½ which had to be accounted for) we realized that while the better results could have been because the model was better, it could have also been because of running the algorithm with different running inputs until desirable results were achieved. Thus we deemed the process too biased and this adjustment to model prediction was shelved.

Model Prediction Results

	left	right	average acc
0	Random Forest Accuracy	XGBoost Accuracy	0.464444
1	Random Forest Accuracy	MLP Accuracy	0.538889
2	Random Forest Accuracy	FT-Transformer Accuracy	0.576667
3	XGBoost Accuracy	MLP Accuracy	0.527222
4	XGBoost Accuracy	FT-Transformer Accuracy	0.575556
5	MLP Accuracy	FT-Transformer Accuracy	0.452778

Figure 5: average acc indicates accuracy when predicting better accuracy over left and right models

left	right	average acc
Random Forest F1 Score	XGBoost F1 Score	0.600556
Random Forest F1 Score	MLP F1 Score	0.492778
Random Forest F1 Score	FT-Transformer F1 Score	0.430556
XGBoost F1 Score	MLP F1 Score	0.478333
XGBoost F1 Score	FT-Transformer F1 Score	0.704444
MLP F1 Score	FT-Transformer F1 Score	0.653889

Figure 6: average acc indicates accuracy when predicting better F1 Score over left and right models

As Fig. 5 and Fig 6. reveal, our results predicting the best model for both accuracy and F1 score did not yield results much better than the 50% average probability expected with a model with two options to choose from. Predicting the model with the better F1 score did see a slightly larger range of accuracies of 0.274 versus 0.125 for predicting the model with better accuracy. Our method of predicting the better model between XGBoost and FT-Transformer saw some of the highest accuracies in both categories (0.575 for accuracy and 0.704 for F1 score), suggesting that there is a small chance there were slightly distinguishable trends in performance between the two models.

Although there was a notable correlation between some of the input features and model performance, there was likely not a significant enough difference in those correlations to correctly predict one model over another consistently. A similar process was repeated using alternative classifiers such as Decision Trees and XGBoost but better performances were not seen.

Project Challenges

On the way we have met with several challenges. This included resource restraints including the fluctuating GPU access time in Google Colab even when upgrading to Google Colab+.

When creating the meta-dataset, we realized that having in-depth meta features, features that would require every row in a dataset to be looked at to be calculated, not only could cause overfitting with our small meta-dataset size but would also require a significant amount of time and resources when processing every row in the dataset. This in turn makes our goal of an efficient model selection process less efficient. If we were to include a meta-feature such as average standard deviation of the dataset columns, we would have also needed to formulate a method to reckon with existing the categorical and binary variables. As such, our only in-depth meta features were average correlation and majority class ratio.

An obstacle we encountered during the data collection was that occasionally all of the algorithms all scored perfectly. We chose to remove these datasets from training the prediction model (though they still appear in the dataset for correlation analysis). We also found that TabNet tended to perform far worse than any other model consistently

to the point that LOOCV could not be done as it required TabNet to do better than the model it is being compared to at least twice and it was unable to do this. As such it was excluded from LOOCV.

A problem during prediction evaluation was dealing with our model being correct by chance. Early on it looked like our model was good at predicting the best model with a smaller dataset. However, when the dataset was expanded to 19, the accuracy decreased drastically. We improved the accuracy slightly again by decreasing the number of trees in our random forest and running it multiple times to get the average.

One challenge we were unable to deal with was that when the algorithms performed very well they did so as a group. Referring back to Fig 2. there is little difference in performance when all the algorithms perform well (average accuracy above 80%) though there are exceptions like in the Credit Card Approval dataset. But when algorithms perform worse the differences are more pronounced. This could have been fixed by collecting more data where the predictions are harder to make accurately or choosing a different task where a more specific error can be calculated like Regression. Sadly this was not viable due to time constraints as the only way to know if the algorithms do well is to run them on the new dataset. The other possible solution was making similar be considered equal. This led to two new problems where the overfitting became even worse with the addition of a new output category which would be solved with more data points, and determining what the cutoff point to be considered equal would be. To get a cutoff point the most logical thing to do would be to run a model 30 times getting the standard deviation of the accuracies and if the accuracies were within

two standard deviations of each other saying they were the same. Both issues could have been solved if not for the limited time and limited gpu usage.

Conclusion and Future Work

While we were able to conclude that correlations do exist between certain dataset features and model performances, the dataset characteristics that we chose were not good enough predictors for which model would perform best. This can at least be said for a binary classification task where all algorithms and models were seen to perform and react similarly to the different changes in datasets.

If we were to continue this experiment further we would have the option of expanding the dataset to include more entries, but we would be more inclined to first make a few other choices. We would likely need to expand the task from only binary classification to also regression where a more specific loss can be calculated, and where we are more likely to see a larger range of results considering the complexity of the architectures. We could also include more algorithms, with the potential to see solidified trends in model performances within architecture subgroups (pair MLP with ResNet, FT-Transformer with Tab Transformer, TabNet with SAINT, etc). Lastly, we could also formulate new meta-features that wouldn't require us overanalyzing the dataset. For example, one possible new meta-feature could be the category of the dataset (financial, medical, species classification, etc).

Responsibility Breakdown

To finish this project, Derek was responsible for running the deep learning algorithms and Logan was responsible for running the machine learning algorithms. Derek then performed most of the initial analyses between the dataset variables and the algorithm performances, whereas Logan performed the model prediction section of the project.

References

[1] Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data?. *Advances in neural information processing systems*, *35*, 507-520.

[2] Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

[3] Hwang Y, Song J. Recent deep learning methods for tabular data. CSAM 2023;30:215-226. https://doi.org/10.29220/CSAM.2023.30.2.215

[4] Arik, S. Ö., & Pfister, T. (2021, May). Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, No. 8, pp. 6679-6687).

[5] Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, *34*, 18932-18943.

[6] Huang, X., Khetan, A., Cvitkovic, M., & Karnin, Z. (2020). Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

[7] Zhu, Y., Brettin, T., Xia, F. *et al.* Converting tabular data into images for deep learning with convolutional neural networks. *Sci Rep* 11, 11325 (2021). https://doi.org/10.1038/s41598-021-90923-y

[8] Zohar, O., Huang, S. C., Wang, K. C., & Yeung, S. (2024). LOVM: Language-only vision model selection. *Advances in Neural Information Processing Systems*, *36*.11

[9] Kadra, A., Lindauer, M., Hutter, F., & Grabocka, J. (2021). Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, *34*, 23928-23941.