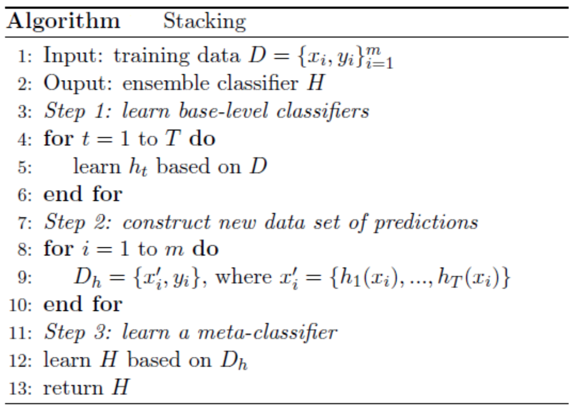
An Analysis of a Stacking Method with Neural Networks for Classification

Lauren Benson
Luddy School of Informatics
Indiana University
Bloomington, IN
Email: lvbenson@iu.edu

Daniel Kaiser
Luddy School of Informatics
Bloomington, IN
Indiana University
Email: kaiserd@iu.edu

Joanna Li
Kelley School of Business
Indiana University
Bloomington, IN
Email: joli@iu.edu

***Abstract*—Neural networks are a well-known tool for machine learning regression and classification tasks. Ensemble methods, such as boosting and bagging are also popular algorithms for harnessing the combinative power of multiple classifiers to optimize the accuracy of predictions as much as possible. Another such ensemble method, called stacked generalization, undergoes similar logic, but has the advantage of not requiring the use of weak classifiers, and can instead handle robust, complex classifiers to form the ensemble. Stacking methods are becoming more popular, but are not as often used in the literature as other ensemble methods. The following paper is a rigorous investigation of one such stacking method that utilizes deep neural networks to form both the ensemble, and the overall stacked model. This investigation employs both real-world breast cancer classification data, as well as assorted simulated data to provide robust analysis from multiple perspectives, and to determine potential applicable usages.**



```
Algorithm    Stacking
Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
Ouput: ensemble classifier $H$
Step 1: learn base-level classifiers
for $t = 1$ to $T$ do
    learn $h_t$ based on $D$
end for
Step 2: construct new data set of predictions
for $i = 1$ to $m$ do
    $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
end for
Step 3: learn a meta-classifier
learn $H$ based on $D_h$
return $H$
```

Fig. 1: Stacking Algorithm Pseudocode

I. Introduction

Ensemble methods are popular techniques in machine learning for enhancing the predictions, decreasing the variance, or decreasing the bias of multiple classifiers. Bagging and Boosting are the ensemble methods associated with the latter two, while stacking seeks to enhance the predictions of the base learners. Another notable difference is that bagging and boosting typically draw on the same type of classifier for their sub-models (such as with a random forest), while stacking can have a diverse ensemble of submodels. The stacking ensemble method follows the general logic of combining multiple sub-models for either classification or regression tasks via a meta-learner, which is trained on the outputs of the sub-models. These sub-models can take many forms, either with different learning algorithms, or different parameters and variations of the same learning algorithms. This heterogeneity of stacking models can lead to an improved combined prediction, while preserving the complexities of the sub-models. Stacking is often used to investigate nuanced questions on difficult data, where no single type of classifier is necessarily superior. The following is pseudocode for the stacking algorithm (adapted from: https://blog.statsbot.co/ensemble-learning-d1dcd548e936):

Researchers and data scientists have found success using stacking methods on a myriad of problems such as parallelizing learning algorithms for deep architectures, [2] and classifying events in particle physics [1]. One such area of machine learning that has gained traction in recent years is that of medical diagnosis analysis and classification, and in particular, cancer diagnosis and predictive analysis. While a variety of machine learning methods and algorithms are employed for cancer-related research, this paper focuses on neural network methods, as recently reviewed in Zhu et al.'s work [6]. Neural networks as applied to cancer diagnosis can be explored extensively in the literature, but specifically neural network stacking models are underdeveloped in the cancer diagnosis and prognosis field. Rather than designing increasingly complex neural network architectures for this purpose, a stacking method applied to an ensemble of neural network submodels allows increased flexibility for submodel design. In 2019, Kwon et al developed a stacking ensemble method for cancer classification [4] but to their knowledge, the authors did not find literature describing such methods on specifically and exclusively neural networks. The following paper investigates a stacked neural network ensemble method on two cancer prognosis datasets, as well as a set of simulated data for comparison.

II. Methods

A. Data

1) Diagnostic Data: The dataset for the diagnostic classification data was collected by Dr. Wolberg from the University of Wisconsin [3]. This data classifies 569 patients with potentially cancerous breast tumors as either malignant or benign. The features for these diagnoses were extracted from images of a fine needle aspirate of a breast mass, and describe characteristics such as the radius, texture, perimeter, area, smoothness, and compactness of the cell nuclei. The authors chose to use this dataset as the first in a three-part analysis, following a linear timeline of a patient's cancer diagnosis and prognosis. Upon initial exploration of this classification data, the authors found that many of the univariate distributions of the features were highly skewed; much of this skewness can be attributed to the high co-linearity of the data. To accommodate this, feature selection was applied to the data to reduce the co-linearity, eliminating certain highly-correlated features (for example, features like perimeter and area are highly co-linear), followed by eliminating unimportant features below a certain threshold. The final set of features was narrowed to ten features, for which the following analysis was applied.

2) Prognosis Data: The dataset for the prognosis classification data was also collected by Dr. Wolberg, as part of an overarching data collection effort that was later pruned into both the diagnostic data and the prognosis data [3]. This data is based on 198 individuals, each of who had previously been diagnosed with breast cancer. The outcome for these individuals is either R=Recurrence (of their cancer) or N=Non-Recurrence, based on 34 features, also extracted with a FNA to describe the cell nucleus of the cancer cells. The significance of using both the diagnostic and prognostic datasets are that they are quite different sizes while maintaining similar numbers of features. The authors performed the same feature selection methods on the prognosis data as for the diagnostic data, again narrowing the feature pool to ten of the most important, non-correlated features.

3) Simulated Data: Given the highly skewed nature of the datasets mentioned above (to be discussed), the authors decided to analyze the stacking model approach on simulated, linearly-separated data. This simulated data was divided into eight subsets, with the focus being on one that has two classifications, eight features, and 200 samples. This subset was selected for further analysis given its similarity in parameters to the prognosis cancer data.

4) Sub-Models: Given that the stacking method is an ensemble method, it is necessary to intelligently select particular submodels that are in line with the classification problem. The framework discussed in this paper is based on deep neural networks for the submodels. Because this framework restricts the submodel classifiers to deep neural networks, the authors decided to vary the parameters of the submodels, as well as the stacking meta-learner in order to rigorously investigate it's performance. When building a neural network classifier, one is faced with choosing from a large set of parameters, such as the hidden layer activation, the output layer activation, the number of hidden layers, and the number of nodes (neurons) in each hidden layer. Changing these parameters affects the performance of the classifier in different ways, and it is not always obvious how to construct a neural network that is appropriate for the problem at hand. In particular, choosing an activation function is critical for optimal neural network construction, because without it, the neural network's output signal would merely be a linear function. An activation function elevates the classifier from a simple linear regression model to a neural network that can learn, recognize, and classify complex mappings from a variety of data types. This non-linearity of the neural network's mappings from inputs to outputs is dependent upon the differentiability of the activation function to allow back propogation, and eventually to apply an optimization technique to the node's weights. While there are a myriad of activation functions to choose from, for both the hidden and output layers, some popular ones and the ones which are implemented in this framework are: sigmoid, tanh, ReLU, and softplus. These activation functions offer certain benefits in certain situations, such as sigmoid functions tend to offer better results in classification problems [5].

Therefore, in constructing the neural network submodels for the stacking framework, the submodel activation functions were selected from a parameter grid with sigmoid, tanh, and ReLU for the hidden layer activations. The output activations for all of the submodels were restricted to softplus. Other parameters that were varied among the submodels were the number of hidden layers, which included nine options, ranging from two to ten hidden layers. The number of nodes per hidden layer was also varied, with seven options ranging from five to thirty-five in increments of five. Finally, there are two options for the loss function in the submodels, either categorical crossentropy or poisson. For all of the submodels, the optimizer was set to Adam optimization, and the metric to Accuracy. Each submodel was trained on the training data for its corresponding dataset, and the total pool of submodels for each dataset totalled to $3x1x2x9x7 = 378$. All of the submodels for all of the experiments were generated with this identical schematic. The authors constructed large pools of submodels for each of the three datasets described above, from which specific models were pulled for each experiment.

5) Stacking Model: In order to investigate the robustness of the stacking method specifically designed with neural networks, the authors also elected to use a neural network for the stacking model, or the meta-learner. This meta-learner is functionally similar to the architecture of the submodels, but differs in a few key ways: first, its input is a compilation of the predictions from the submodels, rather than the training data. Then, the meta-learner learns a new set of of predictions with the goal of finding the best way to combine the contributions from each submodel; this process is based on assigning weights to the submodels depending on the importance on their influence. The stacking model's new set of predictions are the final classifications whose accuracy is, hopefully, better than the individual classifiers. However, the purpose of this investigation is partially to determine under which circumstances does this improved accuracy actually occur. The authors conducted this rigorous investigation in a similar manner to that of the submodels: with varied parameters and comparisons of multiple constructions. For the stacking model, the varied parameters were again the number of layers, the number of nodes, the hidden activation functions, and the loss function. The parameters for these models differed slightly (to compensate for computation time) than those of the submodels, for a total of $3x3x3x3 = 81$ different stacking models. The parameters that were used specifically were the stacking models were layer numbers of 2, 5, and 10, layer sizes also of 2, 5, and 10, hidden activations of relu, sigmoid, and tanh (and for one of the datasets, selu as well), and loss functions of categorical crossentropy, poisson, and binary crossentropy.

III. Results

In the following descriptions of the results for each dataset, the author's investigated the performance of both the submodels and the stacking models in the context of a myriad of experiments, for which the most influential and interesting are reported here.

A. *Diagnostic Cancer Data Results*

The analysis on the stacking model framework on the diagnostic cancer data reveals a few key insights, namely, that the dataset is particularly well-suited for this method. The goal for this, and the following two datasets, is to characterize the stacking method in relation to well-performing and poor-performing submodels. The driving question here is whether or not the stacking method is capable of classifying a dataset with a good accuracy, even if it's submodels offer poor predictions. To test this, a pool of 378 submodels were generated using training data (using a traditional training and testing split) from the diagnostic dataset. These submodels were then categorized as either high-performers or low-performers, corresponding to their accuracies according to a low-performing threshold of less than 0.5 accuracy, and a high-peforming threshold of greater than a 0.8 accuracy. With these thresholds, 180 models were found to be low performers, and 105 to be high performers. Once the submodels were categorized, the stacking models were applied, and their behavior observed. As visualized in Fig. 2, despite the clustering of the low-performing submodels at around 3.9, the stacking model was able to improve their predictions significantly, with some stacking model architectures reaching 100 percent accuracy. Kernel density estimation plots were used to visualize estimates of the trends of accuracies for the submodels as well as for the stacking models. For this experiment and the following ones, the authors opted for kernel density estimation plots over other techniques due to the visual ease of a continuous distribution for overlapping data trends.

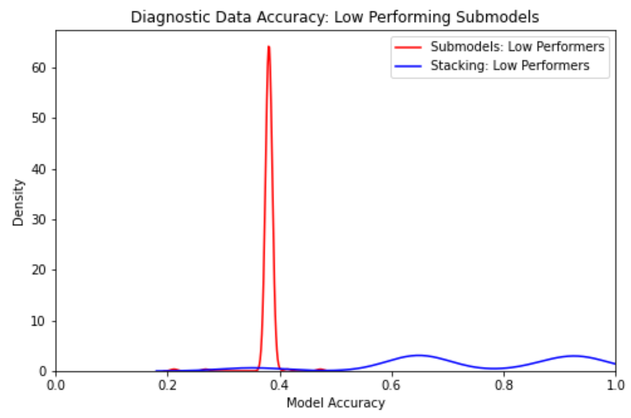


Fig. 2: A Kernel Density Estimation of the stacking method applied to Poor performing submodels. 81 different stacking models are applied to 105 well-performing submodels.

The next step was to explore the conditions in which certain stacking models were more successful at vastly improving the submodel predictions. This step is investigated more thoroughly in the following two sections, but the reader can refer to figures in the project's github for more explicit results regarding this dataset. The main results of from the analysis of this dataset show that the stacking method is quite adept to improving the accuracy of submodels of varying quality, as well as displaying a preference for using relu as a hidden activation function, and fewer numbers of hidden layers specifically for the stacking model.

B. *Prognosis Cancer Data Results*

The prognosis dataset for cancer classification proved to be far more unwieldy than the diagnosis dataset, which can be partially attributed to both the size of the data and its skewedness. However, in some ways, a difficult dataset is an ideal way to stress-test the neural network stacking model framework. For this dataset, a similar experimental process to the diagnostic dataset was carried out, but with a finer-grained analysis. The first experiment was to evaluate 81 stacking models on the prognosis data submodel pool, again dividing the submodels into a high-performing group (21 models), and a low-performing group (200 models), both with the same thresholds as mentioned in the previous section. The results of this experiment are displayed in Figure 3, which shows density plots of the accuracies over both pools of submodels, and their corresponding pools of stacking models. According to these results, the stacking model is quite successful at improving the predictive accuracy of the low-performing submodel group. What is perhaps more surprising are the stacking models' relative lower performance on the high-performing submodels. The plot corresponding to the stacking models with the high performers shows that while some stacking models are indeed able to improve the accuracy of their classifiers, where some reach almost 100 percent accuracy, many of them exhibit slight under-performance. Furthermore, there are a small number of stacking models for this group that only reach about a 30 percent accuracy, which is far lower than the threshold for the submodels in this group. Therefore, this initial experiment reveals that perhaps stacking models are better suited to improving the performance of lower performing classifiers, but that there might be some conditions under which the stacking framework is superior even with high performing classifiers.

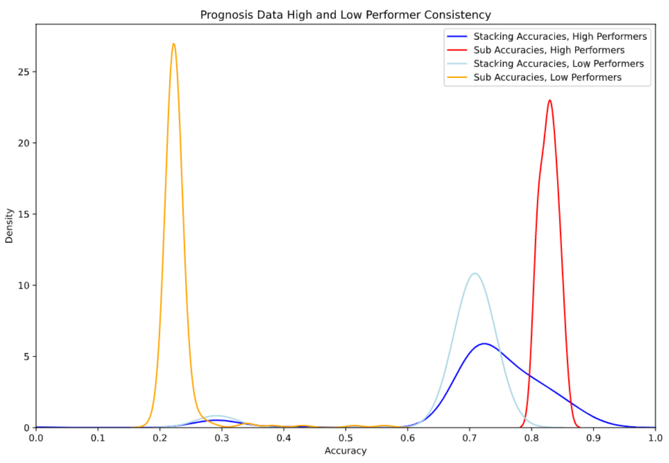


Fig. 3: Stacking Algorithm Pseudocode

The next step of this analysis was to investigate the stacking framework with a more fine-grained perspective, specifically that of the hidden activation function, and the loss function for the stacking models over the well-performing submodels. Figure 4 reveals the impacts of the hidden activation and the loss function on the performance of these stacking models. The general trends of this analysis lean towards relu and tanh as the preferred activation functions. The notable trends of the loss functions are that there is a distinct difference in bias and variance between the three functions: across the board, categorical crossentropy has the lowest variance and the highest bias, while binary crossentropy is the opposite. The poisson function has low variance when paired with relu, but high variance when paired with tanh and selu hidden activations. Other experiments in this dataset include those for stacking models on an ensemble of 9 poor-performing submodels and 1 well-performing submodel (refer to Figure 7 in the appendix), as well as stacking models on the entire corpus of submodels for this dataset (100 poor + 21 good). Both experiments support the earlier analysis where the stacking models appear to have a very positive effect on improving the poorly-performing submodels, but less so on the well-performing submodels.

C. Simulated Data Results

The final dataset the authors analyzed with this framework was generated with simulated data, the purpose being to have a version of a null model comparison, controlled for noise, to explore the possible scalability of the interpretations of the results from the two real-world datasets. To do this, the authors generated eight different simulated datasets, each differing in terms of their size, features, and number of classifiers. The reader is directed to the project's github for initial analysis of all eight experiments. After generating sets of stacking models for each of the eight experiments, further analysis was narrowed to one particular simulated dataset, characterized as previously discussed. This dataset is similar to the prognosis cancer dataset as it has approximately the same number of instances, and 8 features. The first experiment on this dataset was performed similarly to the initial experiment in the prognosis dataset, where 81 stacking models were each trained on corpuses of well-performing and poor-performing submodels. The thresholds for well-performing and poor-performing submodels were set to be accuracies of 0.8 and 0.5 respectively, to acquire 126 well-performing submodels and 47 poor-performing submodels. The results of this experiment were also behaviorally similar, as can be seen in Figure 5, where there are distinct differences in the performances of the groups of submodels, but very little difference in the performances of the corresponding stacking models. Note that these are again kernel density estimations, and therefore the density plots occasionally estimate distributions outside of the thresholds for which they were set. Overall, the stacking models were able to improve the predictions of the low-performing submodels almost to the point of the stacking models that were trained on the high-performing submodels. Unlike in the prognostic data analysis of the same experiment however, the stacking models on the high-performers never underfit their classifiers.

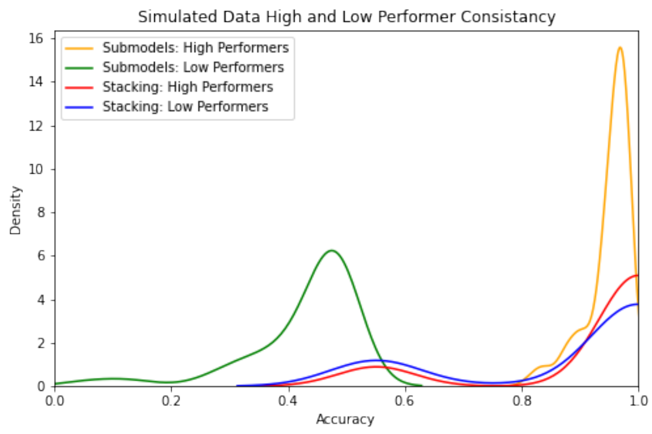


Fig. 5: Stacking Algorithm Pseudocode

Given that the simulated data has no noise and is easily classified, the focus of this dataset was to closely inspect the impacts of the different parameters on both the submodels *and* the stacking models. Given that both types of classifiers are neural networks, the question arises of whether or not the same optimization logic that is applied to the submodels can be applied to the stacking models as well. To answer this question, accuracies for the four sub-experiments explained earlier were extracted as a function of the parameters that were varied across the corpuses of both sub and stacking models. Figure 6 shows this analysis for the stacking models trained

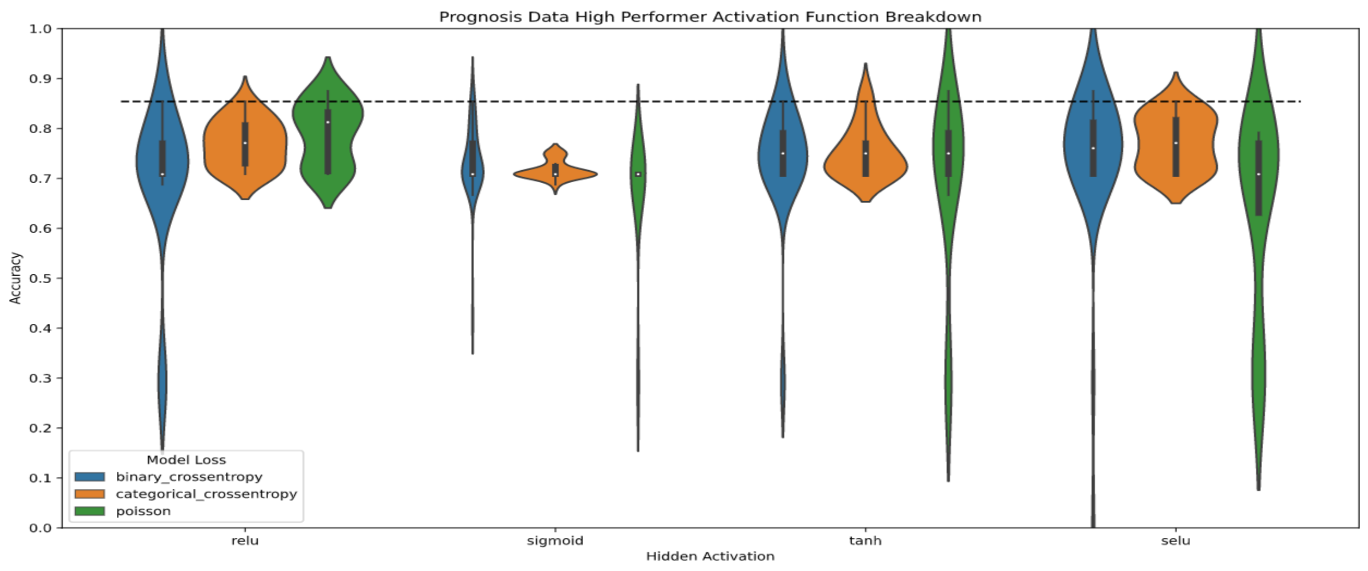


Fig. 4: Violin plots corresponding to the impacts of the hidden activation functions and the loss functions on the 81 stacking models, each trained on 21 well-performing submodels. The black dotted line corresponds to the maximum accuracy of the submodels, thus showing the approximate number of stacking models that perform better than the maximum submodel performance. Firstly, the combinations of categorical crossentropy and relu, poisson and relu, categorical crossentropy and sigmoid, categorical crossentropy and tanh, and categorical crossentropy and selu have the lowest bias in their influence on the stacking model accuracies. However, for the combination with sigmoid, the models are always underperforming the classifiers. On the other hand, the remaining combinations (binary crossentropy and relu, binary crossentropy and sigmoid, poisson and sigmoid, binary crossentropy and tanh, poisson and tanh, binary crossentropy and selu, and poisson and selu) have higher variance, but more often exhibit higher accuracies. Overall, there is no one overarching combination of these two parameters that produces overwhelmingly better-performing stacking models, but in general, relu and tanh seem to be the preferred activation functions, and categorical crossentropy has a slightly higher average accuracy than the other loss functions

on the poor-performing submodels, and the reader can refer to Figure 10 in the Appendix for the parallel analysis in the poor-performing submodels, and the stacking models trained on the well-performing submodels. Note that patterns in the dual slopes in the density plots can most likely be attributed to the parameters in the parameter grid for the submodels and the stacking models that were controlled for, such as the output activation function, which was set to be softmax for both sets of model generation. This figure reveals that the parameters contributing most often to high accuracies for the stacking models that were trained on poor-performing submodels were a hidden activation function of relu, a model loss function of categorical crossentropy, a hidden layer number of 2, and a hidden layer size of 10. On the other hand, the parameters that contribute most often to good performance on the submodel classifiers (refer to Figure 10 in the Appendix) are a hidden activation of tanh, a loss function of poisson, a hidden layer number of 6, and a hidden layer size of 20 or 25.

The main takeaway from the experiments for this dataset are that many of the trends observed in the real-world datasets are indeed scalable to certain simulated data, and that stacking models continue to vastly improve the predictive performance of poor-performing ensembles of neural network submodels.

IV. Discussion

The discussion of this project can be summarized with a few key points. Firstly, that when using such nuanced classifiers as neural networks for classification problems, it is useful to explore the effects of the parameters on the classification accuracy. Secondly, a stacking ensemble framework can indeed be implemented with exclusively neural networks, for both the submodels and the stacking model, but that the selection of these submodels impacts the degree of usefulness of the stacking models. Finally the logic that applies to optimizing deep neural networks, which are in this case used for our ensemble of classifiers, cannot necessarily be applied to optimizing the stacking model. From the experiments conducted on the three different datasets, the main conclusions centered around the interesting finding that when applied to poor-performing submodels, an overarching stacking model is phenomenal at improving the accuracy of the classification problem. In most cases, the improvement is so large that the differences are almost indistinguishable from the stacking models that are trained on originally well-performing classifiers.

The experiments discussed also reveal some nuances in the construction of the submodels and the stacking models that are specific to the types of data that they are trained on. For the diagnostic data, the were a far larger number of samples

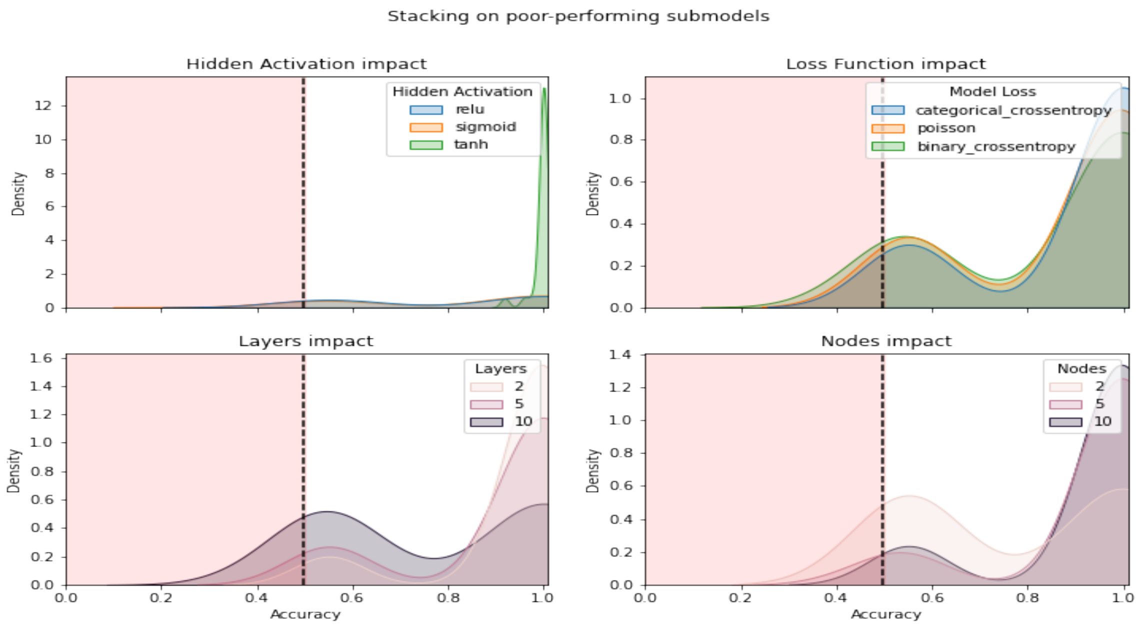


Fig. 6: Stacking model performance, as measured by the accuracy, on a corpus of low-performing submodels. The shaded blocks refer to the accuracy threshold for which the submodels are contained. The stacking models are clearly far outperforming the submodel with the maximum accuracy, as indicated by the black dotted line. The stacking models that perform the best are those with the parameters of tanh, categorical crossentropy, 2 hidden layers, and 10 nodes per hidden layer.

than in either the prognosis data or the simulated dataset, and therefore the submodels were more easily able to classify the data with a high accuracy. However, as shown with the other two datasets, having this high accuracy among the submodels might not even be necessary. Another noteworthy conclusion regarding the specific conditions for both submodels and the stacking models perform optimally is that the submodels tend to perform more optimally under slightly different conditions than the stacking models.

V. Limitations and Future Directions

Despite the rigorous analysis of the stacking model framework discussed in this project, there are a few notable limitations that, if implemented, may improve the validity results. While the purpose of this particular framework was to investigate the nuances of deep neural networks on a stacking ensemble method, incorporating other types of classifiers may reveal different results. An extended analysis of this project would include both stacking models performed on neural networks, and also on a heterogeneous pool of classifiers. Also, because they were limited in computation time, the parameter grid for the corpuses of both the submodel and the stacking model were restricted, but having a more complete set of parameters would reveal even more fine-grained results. Finally, an interesting future direction of this project might be to repeat this analysis on different types of real-world datasets and for different tasks, namely, regression rather than classification tasks.

Acknowledgment

The author of this paper is exclusively Lauren Benson, but she acknowledges the equal contribution to the implementation of the project of all listed authors. The code for the stacking model was inspired by the following: https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/. The code for this project can be found at: https://github.com/FA20-BL-CSCI-P556/final-project-group-7

VI. Appendix

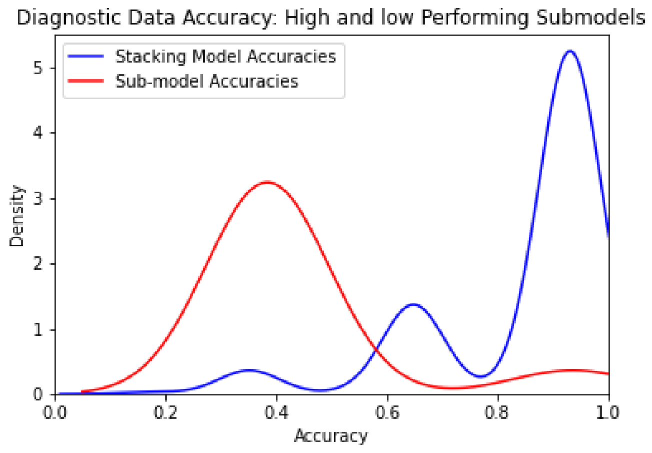


Fig. 7: Diagnostic dataset showing stacking model performance on a set of 9 low-performing submodels, and 1 well-performing submodel. Despite the number of poor submodels, the stacking model outperforms nearly all of them.

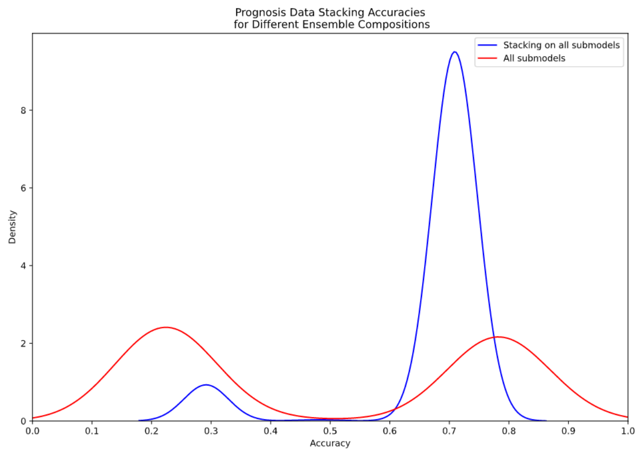


Fig. 8: Stacking methods applied to the entire corpus of submodels, for the prognosis dataset.

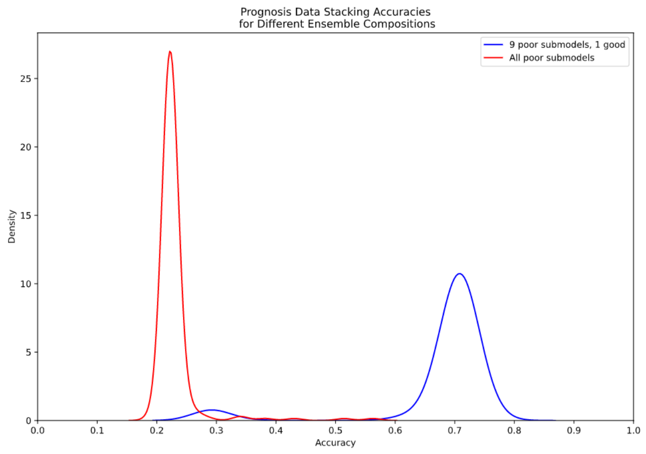


Fig. 9: Diagnostic dataset showing stacking model performance on a set of 9 low-performing submodels, and 1 well-performing submodel. Despite the number of poor submodels, the stacking model outperforms nearly all of them.

References

[1] A. Alves. “Stacking machine learning classifiers to identify Higgs bosons at the LHC”. In: *Journal of Instrumentation* 12.05 (May 2017), T05005–T05005. ISSN: 1748-0221. DOI: 10.1088/1748-0221/12/05/T05005. URL: https://iopscience.iop.org/article/10.1088/1748-0221/12/05/T05005 (visited on 12/16/2020).

[2] Li Deng, Dong Yu, and John Platt. “Scalable stacking and learning for building deep architectures”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan: IEEE, Mar. 2012, pp. 2133–2136. ISBN: 978-1-4673-0046-9 978-1-4673-0045-2 978-1-4673-0044-5. DOI: 10.1109/ICASSP.2012.6288333. URL: http://ieeexplore.ieee.org/document/6288333/ (visited on 12/16/2020).

[3] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[4] Hyunjin Kwon, Jinhyeok Park, and Youngho Lee. “Stacking Ensemble Technique for Classifying Breast Cancer”. In: *Healthcare Informatics Research* 25.4 (Oct. 2019), pp. 283–288. ISSN: 2093-3681. DOI: 10.4258/hir.2019.25.4.283. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6859259/ (visited on 11/02/2020).

[5] Siddharth Sharma et al. “ACTIVATION FUNCTIONS IN NEURAL NETWORKS”. en. In: 4.12 (2020), p. 7.

[6] Wan Zhu et al. “The Application of Deep Learning in Cancer Prognosis Prediction”. In: *Cancers* 12.3 (Mar. 2020). ISSN: 2072-6694. DOI: 10.3390/cancers12030603. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7139576/ (visited on 10/26/2020).

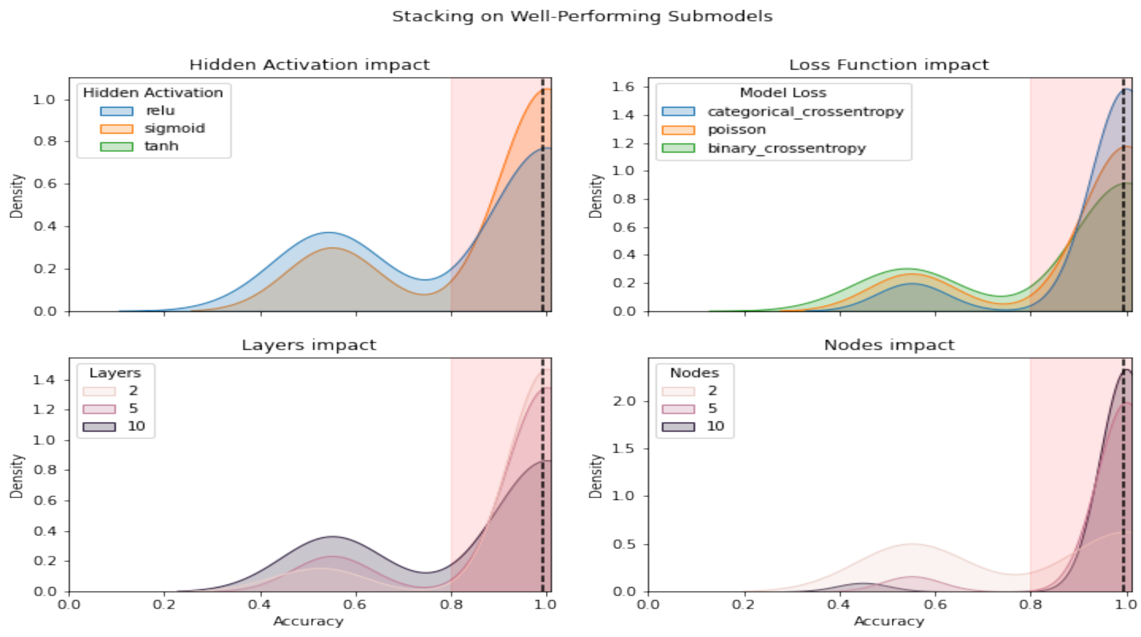


Fig. 10: Stacking models on well-performing submodels.

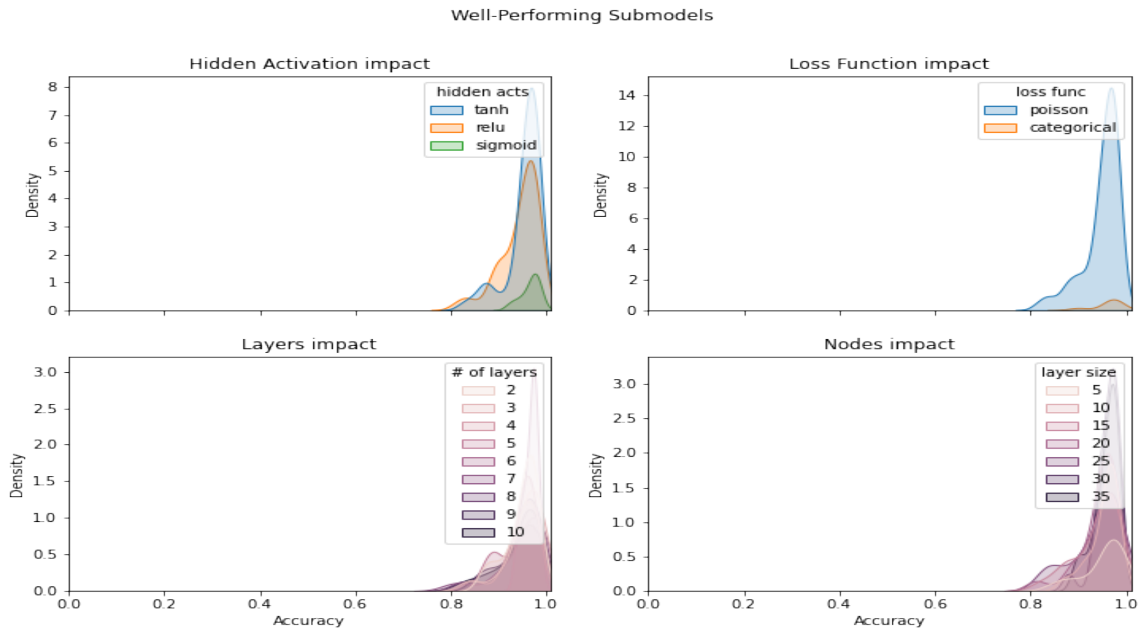


Fig. 11: Well-Performing submodels over different parameters.