



**Abstract**

This project focuses on predicting the functions of human genes from CRISPR genetic interaction screens using machine learning models, including a TensorFlow neural network, KNN classifier, and three tree-based classification models. The goal is to accurately predict the functions of 17,000 genes using an interaction profile and GO term labels as input data, while validating the models using a subset of known gene labels.

The neural network model demonstrated moderate predictive power despite the limited number of positive instances, as indicated by a weighted F1 score of 74.4%. The KNN classifier's performance was improved using oversampling and undersampling techniques, resulting in an average AUROC of 51.5%. The tree-based classification models, including the Random Forest, HistGradientBoosting, and AdaBoost Classifier, were evaluated using AUC values, precision-recall curves, and F1-score vs support plots.

The results of this study provide valuable insights into the strengths and weaknesses of each machine learning model, allowing for informed decisions on which model to use for future applications. It is worth noting that the rarity of positive classes makes the F1-score the primary metric of choice, with the goal being to achieve as many true positives as possible while accounting for the false positive rate.

**Introduction**

Genome-wide screens are a powerful experimental technique used to systematically perturb every gene in an organism to establish its relationship with a specific phenotype. The introduction of CRISPR-Cas9 technology has revolutionized genome-wide screens by enabling the targeted disruption of genes in mammalian cell lines. This technology has allowed researchers to identify essential genes for specific cancers and construct double mutants for gene-gene interaction analysis.

In this project, our goal is to predict human gene function from CRISPR genetic interaction screens using various machine learning models such as TensorFlow neural network, KNN classifier, and random forest classifier (2)(3). We will focus on developing models that can accurately predict the functions of the 17,000 genes from the 1x200 interaction profile while also validating them using a subset of known gene GO term labels. The performance of each machine learning model will be evaluated to determine the best fit for the dataset. Our approach involves using the interaction matrix and GO term labels for a subset of genes as input data to train the models.

**Methods**

The first classifier we used was a neural net consisting of seven layers of nodes: the input layer, two filter layers, three dropout layers, and the output layer. Specifically, each layer had 1,100 nodes, with the output layer having 1,106 nodes for each of the GO terms. The dropout layers were placed in between the input and filter layers. Binary-cross entropy was used as the loss function as this was suited for binary predictions. These parameters were carefully selected through hyperparameter tuning, resulting in the best performance on the testing data. To mitigate the effects of the class imbalance in the GO terms each node in the output layer was biased by the log of the number of positive cases divided by the number of negative cases. This was done to help the neural net with its initial training.

Due to the severe class imbalance in the positive GO annotations, we selected the weighted F1 score as our primary evaluation metric. The weighted F1 scores demonstrated that the neural net had moderate predictive power, despite the limited number of positive instances. Additionally, the binary F1 scores indicated a trend of increasing F1 score as the number of positive GO annotations increased.

The KNN classifier was the second machine learning model that we evaluated. To optimize its performance, we applied both undersampling and oversampling techniques to address the issue of imbalanced data. Specifically, we selected a smaller subset of the overrepresented class (the negative class) and removed any GO annotation for the gene, aiming to balance the dataset and improve the KNN classifier's performance.

To further address the imbalance, we utilized oversampling with SMOTE on the positive class (4). This approach generated additional cases for the model to train on, making it more sensitive to the positive class and improving the overall performance. As a result, we successfully developed a predictive model for the dataset despite its imbalanced nature. By training the KNN classifier on both undersampled and oversampled data, we achieved promising results that showed the effectiveness of our approach in dealing with imbalanced data in machine learning.

We conducted a comprehensive evaluation of the Tree-Based Classification approach, utilizing three distinct classifiers from scikit-learn: the RandomForestClassifier, the HistGradientBoostingClassifier, and the AdaBoost Classifier. Our Random Forest model employed an ensemble of trees without bagging, due to the sparsity of the positive class. In contrast, our AdaBoosted classifier adaptively grew each weak tree to correct the errors of the previous tree, using a weighted average of the weak trees to make predictions. The Gradient Boosted classifier is a more general boosting approach that also adaptively grows weak trees to correct previous errors, using an approximation of the loss function to enable gradient descent.

To assess model performance, we calculated the Area Under the ROC Curve (AUC) values for all GO terms, presented as histograms and averages. We also generated AUC vs support plots to investigate whether performance improved with more positive class instances. Given the rarity of the positive class, we produced precision-recall curves and overlaid them for all GO term classifiers in each model. Additionally, we plotted the F1-score vs support to enable a qualitative comparison between models. Our results provide valuable insights into the strengths and weaknesses of each of the three Tree-Based Classification models, allowing for informed decisions on which model to use for future applications.

It's worth noting that a larger support value should ideally enable a higher f1-score since there are more positive class examples to train on. The f1-score is used as the primary metric because of the rarity of positive classes, with the goal being to achieve as many true positives as possible while accounting for the false positive rate. Using a metric such as accuracy would be misleading because even an untrained classifier that only predicted the negative class would appear to perform well.

**Results**

The neural network classifier was evaluated using the AUROC metric, which produced a mean score of 58.8%. Interestingly, this score remained stable as the support increased, as evidenced by Figures 2 and 3. Due to the severe class imbalance in the positive go annotations, the weighted F1 score was selected as the performance metric, resulting in a mean score of 74.4%. This score indicates that the neural net possesses moderate predictive power despite the small number of positive instances, as seen in Figures 2 and 3. Additionally, the binary F1 score demonstrated a linear relationship with the number of positive go annotations, as depicted in Figure 4. Finally, the precision and recall curve indicated excellent performance for numerous go terms.

Table 1 indicates that GO:0070125, GO:0070126, GO:0006120, GO:0007029, and GO:0032981 are the top predicted GO terms, with binary F1 scores greater than 0.1. These GO terms are mainly linked to mitochondrial functions.

The KNN classifier from sci-kit learn was utilized to evaluate the data, with optimal parameters determined to be k equals 2, resulting in an average AUROC of 50.6%. This score suggests that the model performed no better than random guessing. To improve this outcome, a combination of oversampling and undersampling techniques were employed. The mean AUROC was slightly enhanced to 51.5% through undersampling, as seen in Figure 9. However, as demonstrated in Figure 11, the precision-recall curves indicate that the model was unable to accurately predict GO terms, and any optimization improvements were minimal.

On the other hand, Figure 10 demonstrates a positive correlation between support and the F1 score, indicating that increasing support results in a higher F1 score. Table 2 lists the highest F1 scores for the GO terms using the KNN classifier in ascending order.

The final machine learning model we evaluated was the Tree-Based Classification approach. The RandomForestClassifier generated a ROC AUC of 0.536. The AUC scores do not appear to improve with higher support (Figure 19). The precision-recall curves show a few outlier GO terms that are predicted well (Figure 7). The support vs f1-score plot does not show an increase in f1-score as the support increases (Figure 20). The AdaBoostClassifier uses a default max depth of 1. This approach generates a series of weak learners that are generated sequentially and focus on correcting the error made by the previous learner. The average ROC AUC was 0.520 and the AUC does not appear to improve with increasing support (Figure 14). The precision-recall curves show worse performance than the RandomForestClassifier. The support vs f1-score plot also does not show an increase in f1-score as the support increases.

The HistGradientBoosting Classifier was trained using 200 iterations, and its precision-recall curve revealed that it predicted fewer outlier GO terms accurately, as shown in Figure 6. The average ROC AUC obtained was 0.511, and there was no noticeable improvement in AUC with increasing support, as indicated by Figure 16. However, the support vs. f1-score plot was particularly intriguing, as there appears to be an improvement in the f1-score as support surpasses 50, indicating that the model's performance may improve as more representatives of each GO term are included, as demonstrated in Figure 18. Additionally, a larger number of GO terms showed a non-zero f1-score, indicating that the model was able to predict these terms with some degree of accuracy.

In Table 3, we present the GO terms that yielded non-zero f1-scores across the three models that were discussed. Moreover, we carefully examined the performance of the HistGradientBoosting Classifier and identified the GO terms with an f1-score of 0.2 or higher, which are listed in Table 4. Interestingly, we observed an overlap between the two tables for two GO terms (GO:0022900 and GO:0032981), indicating their importance in the classification process. Notably, the GO terms related to respiration, including mitochondrial functions, the electron transport chain, and glycolysis, were prominently featured in both lists.

**Conclusions**

In conclusion, the evaluation of multiple machine learning models for predicting GO terms demonstrated varied performance, with some models showing moderate predictive power while others performed no better than random guessing. The neural network classifier showed moderate predictive power despite the severe class imbalance in positive annotations, as evidenced by the weighted F1 score. The KNN classifier and Tree-Based Classification approach, including the RandomForestClassifier and AdaBoostClassifier, did not perform as well, with minimal improvement even after employing oversampling and undersampling techniques. However, the HistGradientBoosting Classifier demonstrated some promising results, especially with an improvement in the f1-score as support increased beyond 50. The GO terms related to respiration and mitochondrial functions were prominently featured in the highest F1 scores across multiple models, indicating their importance in the classification process. Overall, these findings highlight the potential of using machine learning models for predicting Gene Ontology terms, but also the challenges associated with severe class imbalance and the need for further optimization and exploration of different algorithms and techniques.

**Figures/Tables**

**Chart, histogram

Description automatically generated**Chart, scatter chart

Description automatically generated

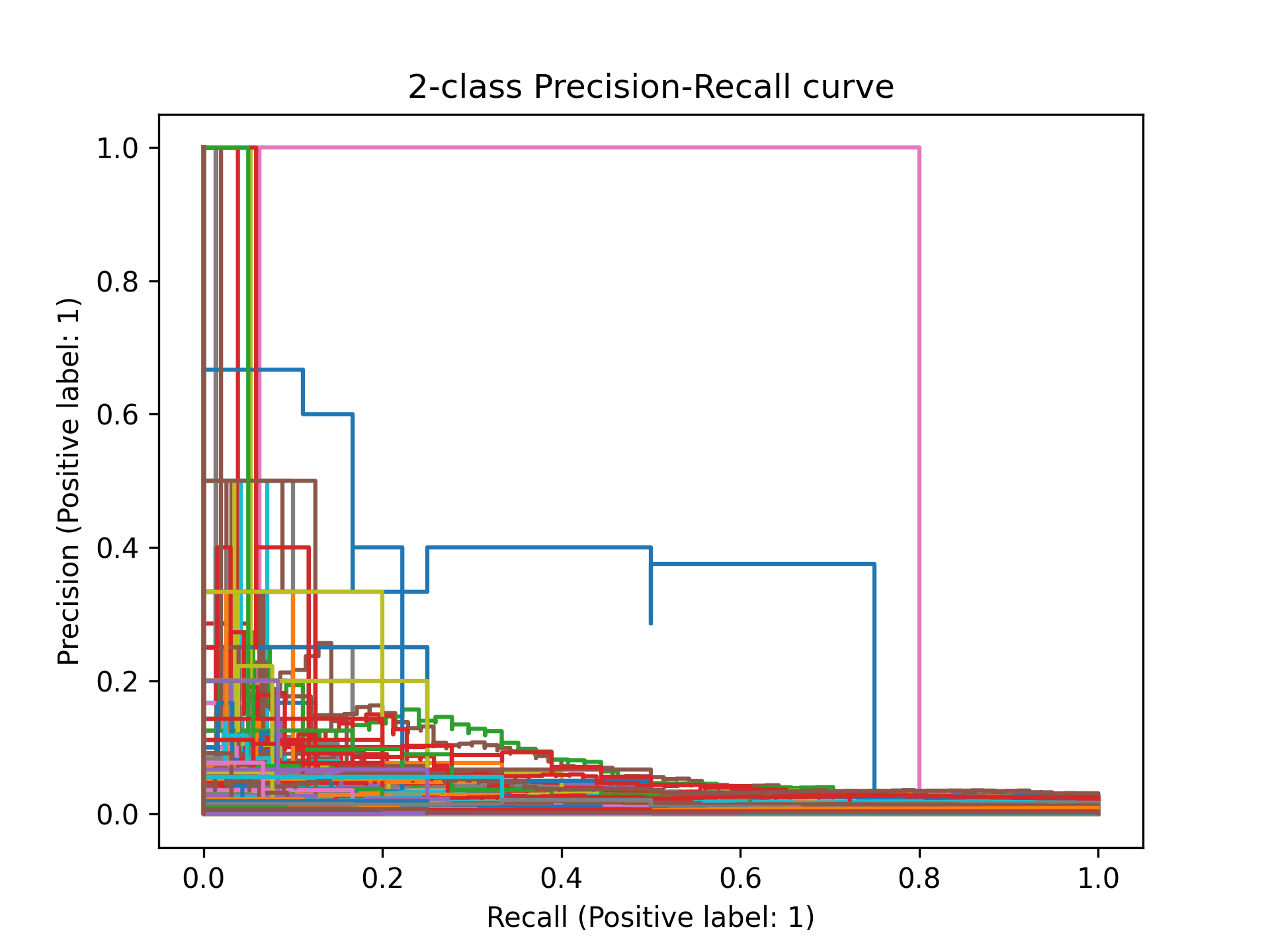
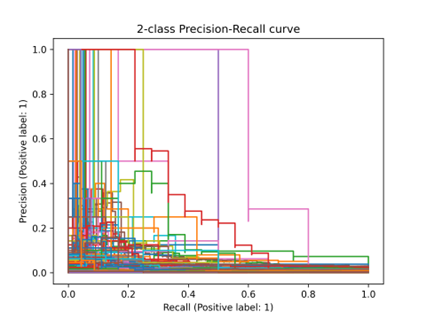
*Figure 1. Precision-Recall curve for TensorFlow Figure 2. Effect of data set size for TensorFlow*

Chart, scatter chart

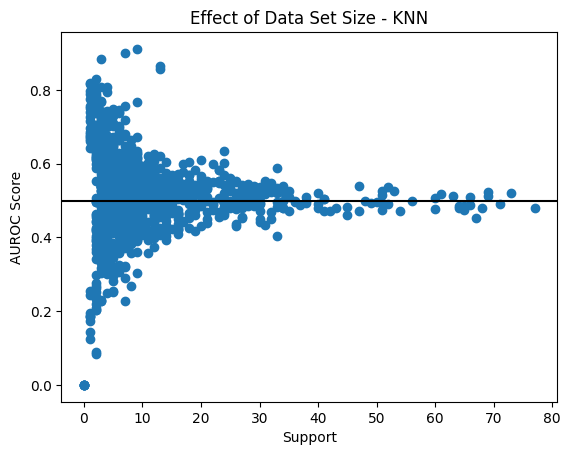
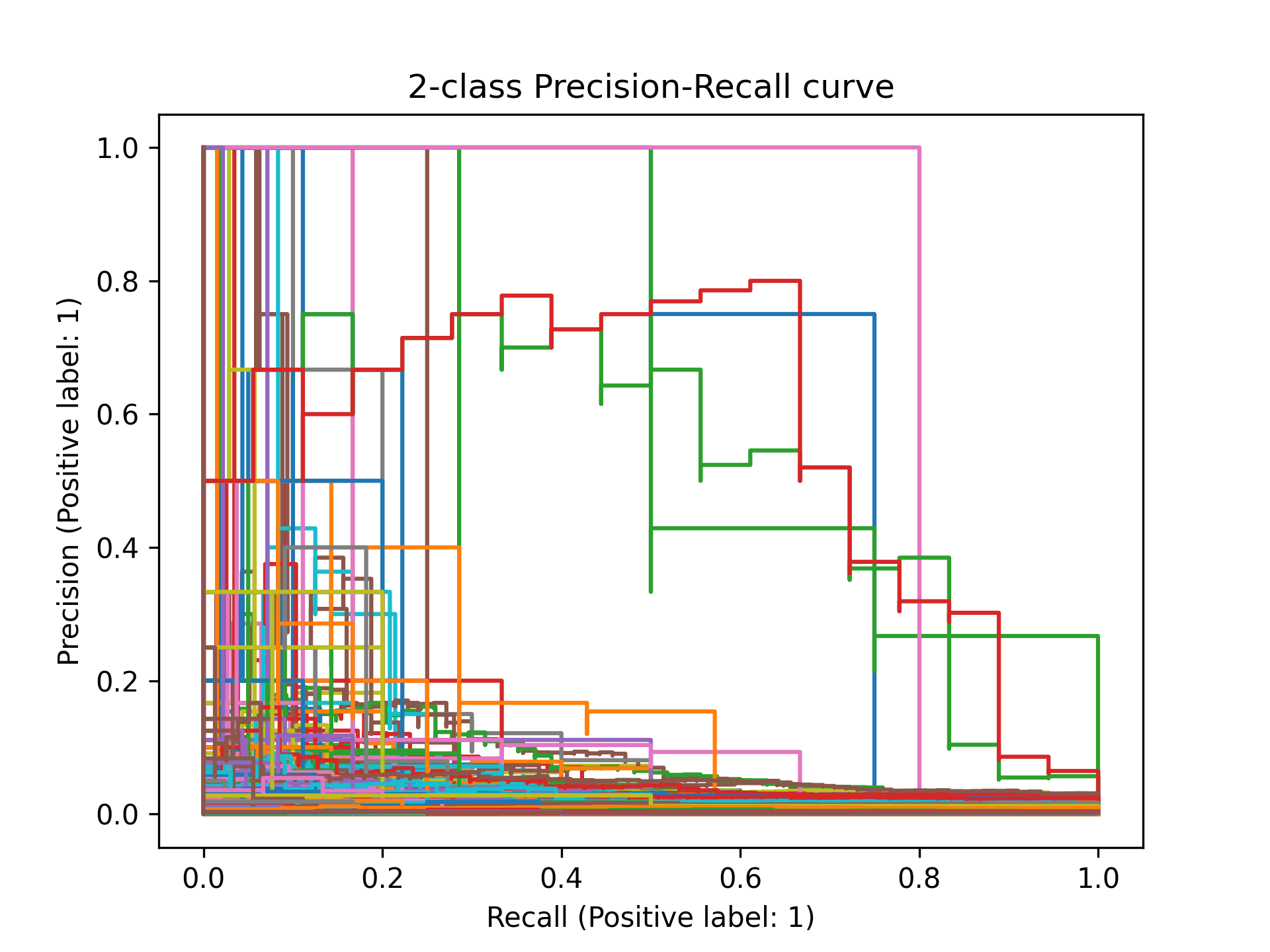
Description automatically generated**Chart, scatter chart

Description automatically generated**

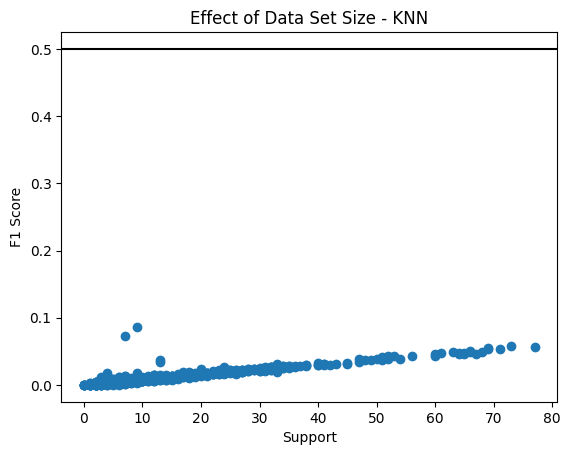
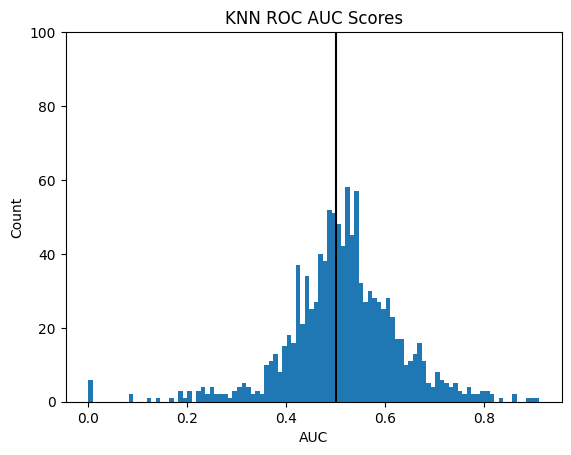
*Figure 3. Effect of data set size for TensorFlow**Figure 4. Binary F1 score vs Support for TensorFlow*



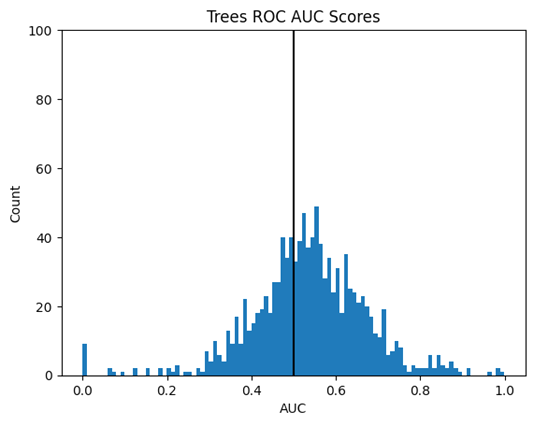
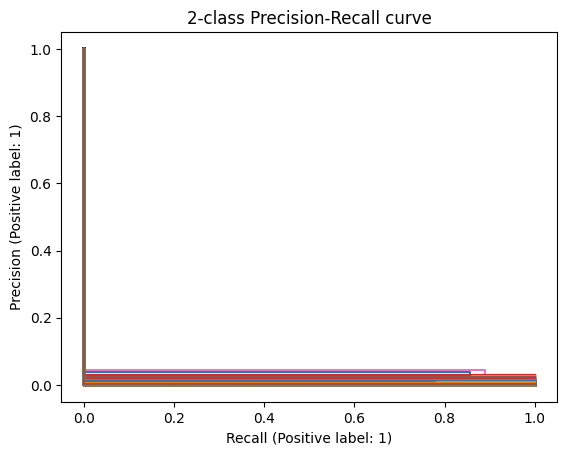
*Figure 5. ADA Precision-Recall Curve Figure 6. Boosted Precision-Recall Curve*



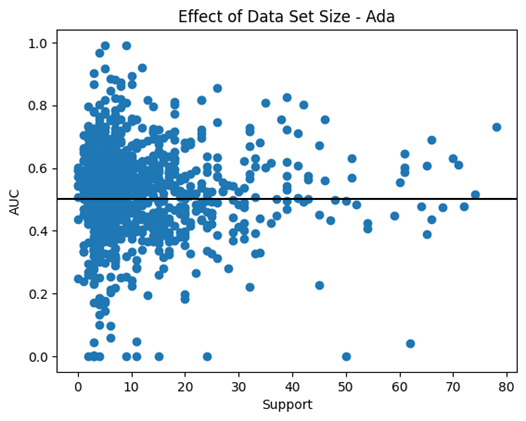
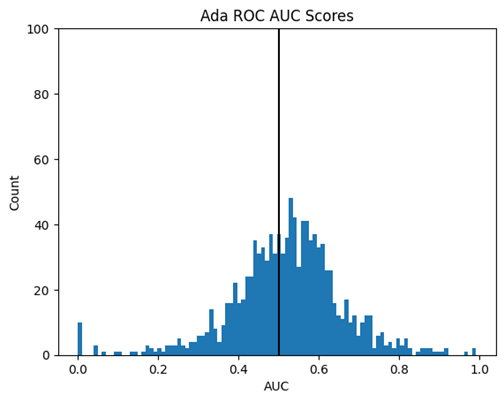
*Figure 7. Trees Precision-Recall Curve Figure 8. KNN support vs AUROC score*



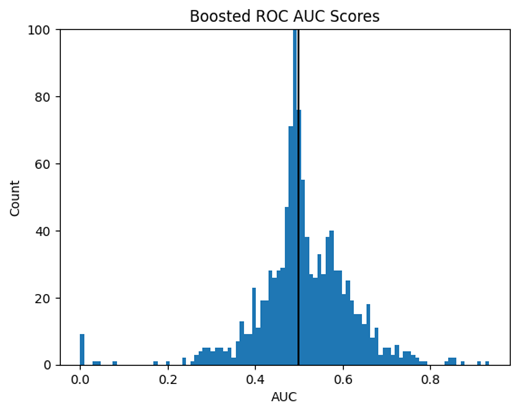
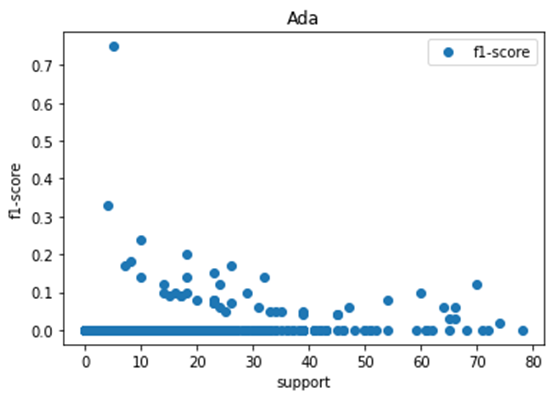
*Figure 9. KNN ROC AUC scores Figure 10. KNN effect of Data set size*



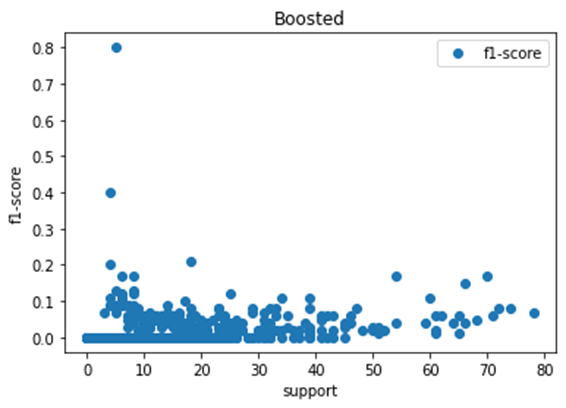
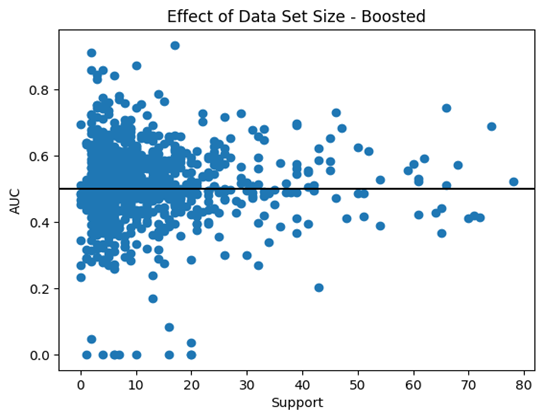
*Figure 11. KNN Precision-Recall Curve Figure 12. Trees ROC AUC scores*

**

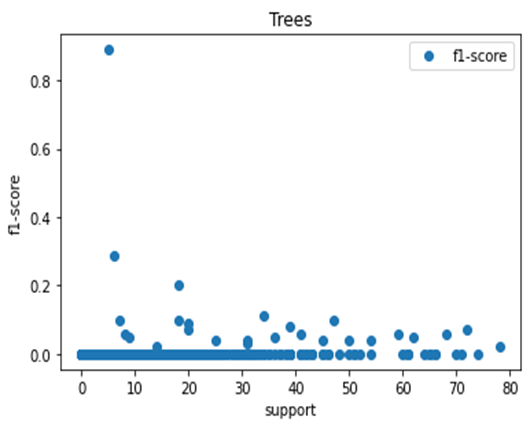
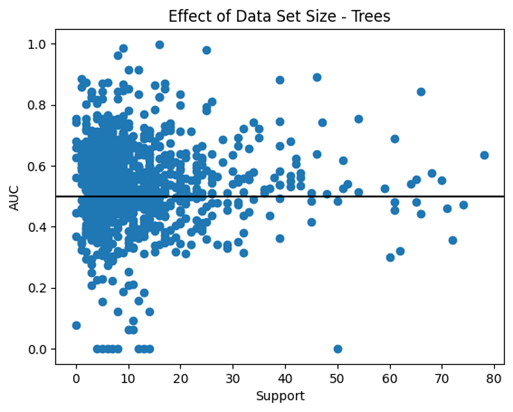
*Figure 13. ADA ROC AUC scores Figure 14. ADA effect of data set size*

**

*Figure 15. Ada support vs F1-score Figure 16. Boosted ROCF AUC scores*

**

*Figure 17. Boosted Support vs AUC Figure 18. Boosted support vs F1 Score*

**

*Figure 19. Random Forest effect of data set size Figure 20. Random Forest Classifier f1 score*

| **Binary F1 Score** | **GO Term** | **Function** |
| --- | --- | --- |
| 0.10030090270812438 | 0006397 | mRNA processing |
| 0.10122699386503069 | 0045786 |  |
| 0.11904761904761903 | *0038061* | NAD(P)+ binding |
| 0.1241565452091768 | 0006281 | DNA repair |
| 0.13197969543147206 | 0070125 | Mitotic spindle checkpoint |
| 0.15204678362573099 | 0070126 | Mitotic spindle organization |
| 0.1538461538461538 | 1900026 | Positive regulation of rRNA binding |
| 0.16 | 0050918 | Positive chemotaxis |
| 0.25 | 0042594 | Response to starvation |
| 0.3333333333333333 | 0007029 | Endoplasmic reticulum organization |
| 0.7999999999999999 | 0006120 | Mitochondrial electron transport |
| 0.9411764705882353 | 0032981 | Mitochondrial respiratory chain complex I assembly |

*Table 1: GO terms with Binary F1 score greater than 0.1. Go functions obtained from ref 1.*

| GO:0001819 | Positive regulation of cytokine production |
| --- | --- |
| GO:0006811 | Ion transport |
| GO:0006952 | Defense response |
| GO:0045184 | Establishment of protein localization |
| GO:0006511 | Ubiquitin-dependent protein catabolic process |

*Table 2. KNN GO terms with the highest F1 score.Go functions obtained from ref 1.*

| GO:0070126 | mitochondrial translational termination |
| --- | --- |
| GO:0032981 | mitochondrial respiratory chain complex I assembly |
| GO:0006091 | generation of precursor metabolites and energy |
| GO:0022900 | electron transport chain |
| GO:0006897 | endocytosis |
| GO:0050907 | detection of chemical stimulus involved in sensory perception |

*Table 3. RandomForest GO terms with non-zero f1-scores. Functions obtained from ref 1.*

| *GO:0032981* | *mitochondrial respiratory chain complex I assembly* |
| --- | --- |
| *GO:0006120* | *mitochondrial electron transport, NADH to ubiquinone* |
| *GO:0022900* | *electron transport chain* |
| *GO:0061621* | *canonical glycolysis* |

*Table 4. HistGradientBoosting Classifier GO terms with f1-score of 0.2 or better. Go functions obtained from ref 1.*

**References**

1. Binns D, Dimmer E, Huntley R, Barrell D, O'Donovan C, Apweiler R. (2009) QuickGO: a web-based tool for Gene Ontology searching. Bioinformatics. 2009; 25(22):3045-6.
2. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
3. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
4. Guillaume Lemaitre, Fernando Nogueira, & Christos K. Aridas (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*. Journal of Machine Learning Research, 18(17), 1-5.*