PREDICTING FLIGHT DELAY USING RANDOM FOREST ALGORITHM, XGBOOST ALGORITHM, AND STACKING ENSEMBLE METHOD

[1]Ferrey Adinarta, [2]Yulianto Tejo Putranto
[1,2]Faculty of Computer Science, Soegijapranata Catholic University
[1]21k10007@unika.ac.id, [2]yulianto@unika.ac.id

ABSTRACT

Flight delays are problematic for both passengers and airlines. With the increasing amount of flight traffic volume, time punctuality is important since it significantly influences passengers' satisfaction and airline companies' financial performance. Many studies have been conducted to predict these delays by using machine learning algorithms. In some research, it was found that combining more than one machine learning algorithm can improve the prediction results. Therefore, in this research, a comparison of machine learning ensemble methods like bagging, boosting, and stacking to predict flight delays is compared. The objective of this research is to find the best-performing ensemble method for flight delay prediction. A dataset from Kaggle named 'Flight Status Prediction' is used as the dataset for this research. Then, the dataset is cleaned and modified using the preprocessing steps. After that, the dataset is fitted to each ensemble model using the Random Forest algorithm as the bagging method, the Extreme Gradient Boosting (XGBoost) algorithm as the boosting method, and combining both algorithms using the stacking method with Random Forest as the first learner, and the results are evaluated based on the accuracy, recall, and precision values. The results are gotten from two different dimensional reduction methods, which are feature selection and principal component analysis (PCA). The results obtained from this study are that the XGBoost model performs best on predicting flight delays with a mean average accuracy of above 95% in both dimensionality reduction methods, while the Stacking Ensemble method performs the worst with a mean accuracy of less than 92% in both dimensionality reduction methods.

Keywords: Flight delay prediction, Ensemble method comparison, Random Forest, Extreme Gradient Boosting (XGBoost), Stacking Ensemble Method, Principal Component Analysis, Feature Selection

INTRODUCTION

Background

The global increase in flight traffic and passengers, despite the COVID-19 pandemic, has led to increased flight delays, disrupting airports, increasing airline costs, and causing time loss for passengers. Machine Learning classification models have been used to predict flight delays, with tree-based models performing better than others. The Random Forest algorithm and XGBoost algorithm have shown the best results in predicting 24-hour flight delays. This paper aims to compare results using different ensemble methods, such as Bagging, Boosting, and Stacking, and analyze the results based on their theories. The goal is to find the best ensemble method using XGBoost and Random Forest algorithms.

Problem Formulation

Here are the key problems emphasized in this paper:

1. Which ensemble method is the best at predicting flight delay based on its accuracy, precision, and recall values?
2. Which ensemble method the best at predicting flight delay based on its cross validation accuracy values?
3. Which dimensional reduction methods works best on each ensemble method when predicting flight delay?

Scope

This study uses the "Flight Status Prediction, Combined Flights 2022" dataset from Kaggle, a compiled dataset taken from the Bureau of Transportation Statistics, to compare the accuracy, precision, and recall of bagging, boosting, and stacking ensemble methods on predicting flight delay. Each ensemble method will only use one model except for the stacking ensemble method. The task that is focused on this study is a classification type of task, so the results are a yes or no classification representing the flights' total delay status. In this study, the author will only note and analyze the utility usage performance and does not improve or optimize the utility usage performance of each ensemble method like speed, memory usage, etc.

Objective

The objective of this study is to compare and analyze the results from three different ensemble methods to find which ensemble method is the best to predict and classify flight delays using multiple dimensional reduction methods.

LITERATURE STUDY

Yuemin Tang's [3] study on machine learning classification algorithms for predicting flight delays found that decision tree-based models performed better than Random Forest, with an accuracy of 92.4%. Seongeun Kim and Eunil Park's [4] research on weather factors found Random Forest as the best model for all airports, with an accuracy of 84.3% for predicting the 8-hour difference. Mingdao Lu et al.'s [5] research on Chinese flights from 2015 to 2017 also found the best boosting models for predicting flight delays. The authors suggest using ensemble methods for mixing multiple models and XGBoost for the boosting ensemble method. These studies provide valuable insights into the effectiveness of machine learning algorithms in predicting flight delays and suggest potential future research using larger datasets and ensemble methods.

Yiheng Li and Weidong Chen's [6] research compared ensemble methods like bagging, boosting, and stacking on scoring credits using the Lending Club credit card dataset. The results showed Random Forest as the best bagging method in five performance criteria, while XGBoost was best for time and hardware limits. This paper can provide a good fundamental outlook on how different ensemble methods produce unique results. The stacking model's results were dynamic

based on the base models used. Xunuo Wang et al.'s [7] study compared ensemble methods for predicting flight delays using PEK flight records, showing a 95.7% accuracy rate and reduced MAE and RMSE of predictions.

Rosalin Sahoo et al.'s paper [8] uses bagging and stacking ensemble methods to predict air cargo delays, with the stacking method outperforming standalone models with a 96.44% accuracy. Ibomoiye Domor Mienye and Yanxia Sun's [5] survey covers ensemble methods in fraud detection, medical diagnosis, and sentiment analysis classification. Yuji Horiguchi et al.'s study compares machine learning classification models for fuel consumption and flight delays using big data. While Yuzhen Zhang et al. [10] and Miguel Lambelho et al. [11] evaluate ensemble methods for remote sensing applications, focusing on Random Forest for bagging and ADABoost, GBM, XGBoost, and LightGBM. They highlight differences in boosting algorithms and feature selection, and experiment with dimensionality reduction and feature number to observe performance changes.

Based on the literature above, this research is conducted to compare and find which ensemble method has the best performance in predicting flight delays. XGBoost is used as the boosting method, Random Forest is used as the bagging method, and both algorithms are combined using the stacking method. Theories and formulas about the three ensemble methods are taken from the paper conducted by Ibomoiye Domor Mienye and Yanxia Sun [5]. This paper differs from previous research by using a different dataset and not using machine learning models for flight delays prediction, unlike Kristanto et al.'s [14] and Soesantio et al.'s [15], which use Random Forest but not compare dimensional reduction methods.

RESEARCH METHODOLOGY

Dataset Collection

The dataset used in this paper is taken from Kaggle.com, titled “Flight Status Prediction." The dataset contains data compiled from the Bureau of Transportation Statistics, specifically the on-time departure data from 2022 in the United States. This dataset contains more than two hundred thousand million rows and columns served in the form of a CSV file. “Flight Status Prediction” dataset can be downloaded from this Kaggle link[1].

[1] https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022?select=Combined_Flights_2022.csv (accessed on April 20, 2024)

Dataset Preprocessing

The dataset is cleaned by removing null and duplicate values, then the Interquartile Range (IQR) method is used to detect outliers in the target variable (ArrDelay, DepDelay). A new column called 'Total Delay' is created by adding departure and arrival delay values, which are then binarized as true or false Boolean variables. Columns not significantly affecting the target variable are removed, such as 'DepTimeBlk', 'ArrTimeBlk', 'Cancelled', 'Diverted', and 'FlightDate'. The categorical data is split based on cardinality, with columns with unique values less than 15 considered low cardinality. The data is split into different percentages of training and testing data to be fitted into the Random Forest and XGBoost model. Two dimension reduction methods are used: feature selection using a correlation matrix for numeric columns, and principal component analysis to reduce the dimension. The results from both dimension reduction methods will be evaluated in the three models.

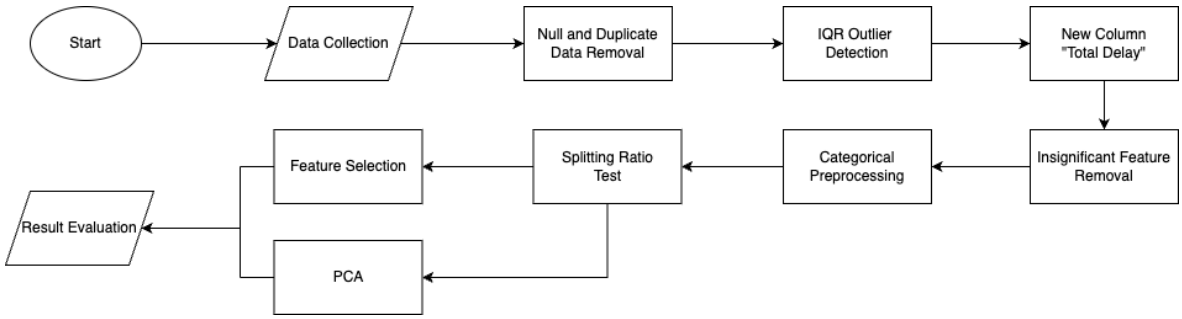


Figure 1. Preprocessing Flowchart

Synthetic Minority Oversampling Technique (SMOTE)

Synthetic Minority Oversampling Technique (SMOTE) is an oversampling method that is used to balance the dataset used in this project. SMOTE works by creating synthetic data in the minority class to make the dataset more balanced. First, SMOTE identifies the minority class data, and then it finds the k nearest neighbor in the minority class. In this study, the k value of the SMOTE is set to default, which is five. After that, SMOTE creates new synthetic samples using interpolation. Lastly, the process is repeated until the minority class has reached the same amount of data as the majority class [12].

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimension reduction technique that reduces multivariable data into principal components. It standardizes the dataset, creates a covariance matrix, and sorts the principal components based on their correlation strength. In this study, the author selects the principal components with 95% of the total variance and transforms the data into a new dataset. [13]

Hyperparameter Tuning

Random Forest

The Random Forest model is imported from a library called Scikit Learn. The author used three of the most commonly used and impactful parameters to tune. The first parameter to tune is called ‘n_estimators’; this parameter is used to set the number of trees in the model, the bigger the value the more complex the model will be. The author used the value one hundred which is the default value, then two hundred, and three hundred node trees to try. The second parameter to tune is called ‘max_depth’; this parameter is used to set the maximum depth of trees in the model, the bigger the value the model will be prone to overfit. The author used the value three, six, nine, and twelve maximum tree depth to try. Lastly, the third parameter to tune is called ‘min_samples_split’; this parameter is used to set the minimum number of samples before the node splits, the author used the value two, four, and eight minimum samples per split to try.

XGBoost

The author used three of the most commonly used and impactful parameters to tune. The first parameter to tune is called ‘n_estimators’; this parameter is used to set the number of trees in the model, the author used the value one hundred which is the default value, then two hundred, and three hundred trees to try. The second parameter to tune is called ‘max_depth’; this parameter is used to set the maximum depth of trees in the model, the author used the value three, six, nine, and twelve maximum depth to try. The third parameter to tune is called ‘learning_rate’; this parameter is used to 0, the higher the value the faster the model will learn but more probability to overshooting and vice versa for lower values. The author used the value 0.01, 0.05, 0.1, and 0.2 learning rate to try.

After that, the fourth parameter is called ‘subsample’; this parameter is used to set the sample ratio of the data used for each tree. The author used 0.5, 0.6, 0.7, and 0.8 sample ratio values to try. Lastly the final parameter is called ‘gamma’; this parameter is used to set the minimum loss reduction before the split happens. The author used 0.1, 0.3, and 0.5 as the gamma value.

Ensemble Methods and Models

Bagging Ensemble Method

Bagging is one of the ensemble methods that works by creating a number of training sets for multiple base learners. Then, the results from the base learners are aggregated by the combination rule of that model. In this paper, the bagging ensemble model that is used is Random Forest algorithm. Random Forest has its own bagging method that creates multiple decision tree models as the base learner, and then each of the results is combined using the majority voting method as shown in 0 [3].

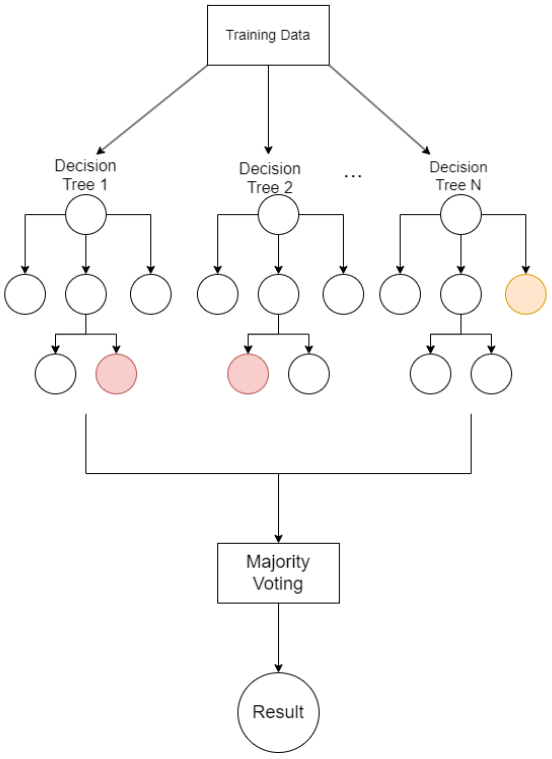


Figure 2. Random Forest Flowchart

Boosting Ensemble Method

The boosting ensemble method is an iterative method that boosts the base learner algorithm's performance by adjusting the input data samples and re-training the base learner with the newly adjusted input data. In this paper, the XGBoost Classification method is used as the representative of the boosting ensemble method. XGBoost works by iteratively creating decision trees and training them sequentially; the newest decision tree will learn based on the performance of the last decision tree by using residual values. The final result of the model will be calculated based on the weighted sum of each tree prediction, as shown in Figure 3 [5].

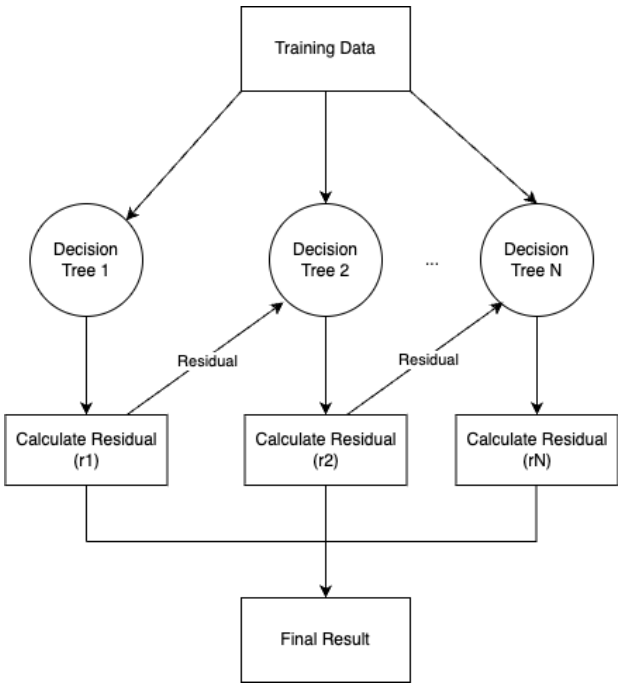


Figure 3. XGBoost Flowchart

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \quad (1)$$

Function $L^{(t)}$ is an output equation of XGBoost Classificaton model. The equation sums the decision tree outputs for each iteration n. $l(x)$ is a function of decision trees learner that finds the loss of each data point i, while y_i is the true label or target prediction and $\hat{y}_i^{(t-1)}$ is the predicted value from the last decision tree. After that, $f_t(x_i)$ is a function prediction of the new decision tree based on the results and residuals of the last tree. Then, XGBoost also facilitates regularization, which is represented with the $\Omega(f_t)$ equation. Lastly, the results are summed to get the final prediction result.

Stacking Ensemble Method

Stacking is another ensemble method that stacks two or more machine learning models and combines the results sequentially. Stacking requires two layers of stacks; the first layer is called the base learner, and the second layer is called the meta learner [5]. In this paper, Random Forest is used as the base learner model since it works best to minimize overfitting. After that, a new set of data consisting of its predictions is created by the Random Forest model. Then, the XGBoost model is used as the meta-learner to learn the newly created dataset and perfect the results from the Random Forest model as shown in Figure 4.

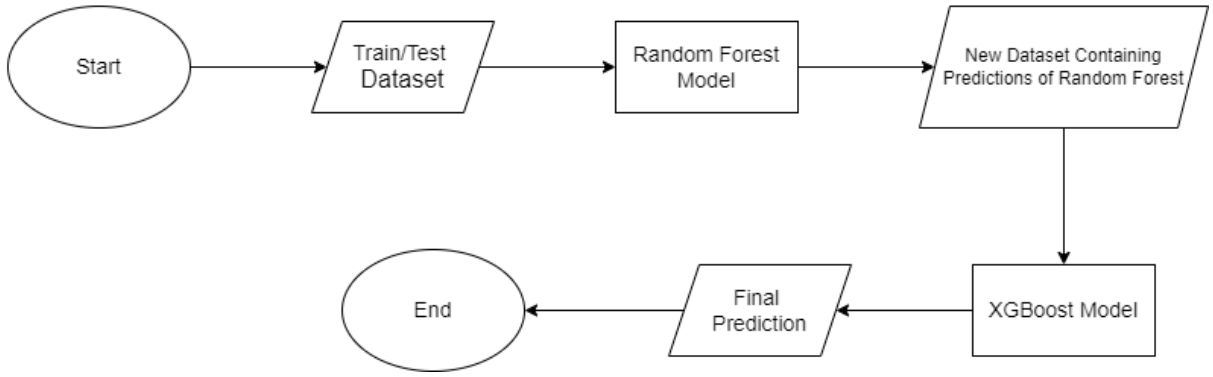


Figure 4. Stacking Ensemble Method Flowchart

Result Evaluation and Comparison

After the three ensemble methods are trained, the next step is to make an evaluation based on some evaluation values like true positive (TP), true negative (TN), false negative (FN), and false positive (FP). TP is the number of correct delayed flight predictions, TN is the number of correct non-delayed flight predictions, FN is the number of false non-delayed flight predictions, and FP is the number of false delayed flight predictions. The first metric is Accuracy, which is obtained by adding the total correct prediction (TP and TN) and then dividing it by the whole number of tests (TP, TN, FN, and FP). The second metric is Precision, which is obtained by dividing the number of TP with the sum of TP and FP. The last metric is Recall, which is obtained by dividing the number of TP with the sum of TP and FN

IMPLEMENTATION AND RESULTS

Experiment Setup

The experiment of this study was conducted using Python version 3.10 in Google Colab with a free 11.7 GB RAM and a free 71.2 GB disk space. The device used to run this experiment is an Apple Macbook Air with an Apple M1 chip. Multiple libraries from Python version 3.10

were used in this study, such as Pandas to read csv, Numpy to manipulate arrays and matrixes, Sklearn to use multiple machine learning helpers, Imblearn to use SMOTE, and XGBoost for the boosting ensemble method model, and an additional library was downloaded from Google Colab named category encoders to preprocess categorical data.

Results

The author has tried to predict flight delays by using three different ensemble method models. The results are then evaluated and compared based on each algorithm's accuracy, precision, recall, and computational speed values based on the test set and validation set. The author also compares the cross-validation accuracy results for the train and test set for the overfitting test.

Accuracy, Precision, and Recall Values On Test Set (Feature Selection)

Table 1. Result Comparison On Test Set

Evaluation Metrics	Random Forest	XGBoost	Stacking Ensemble
Accuracy	99.74%	99.98%	99.55%
Precision	99.45%	99.98%	99.24%
Recall	99.7%	99.97%	99.29%

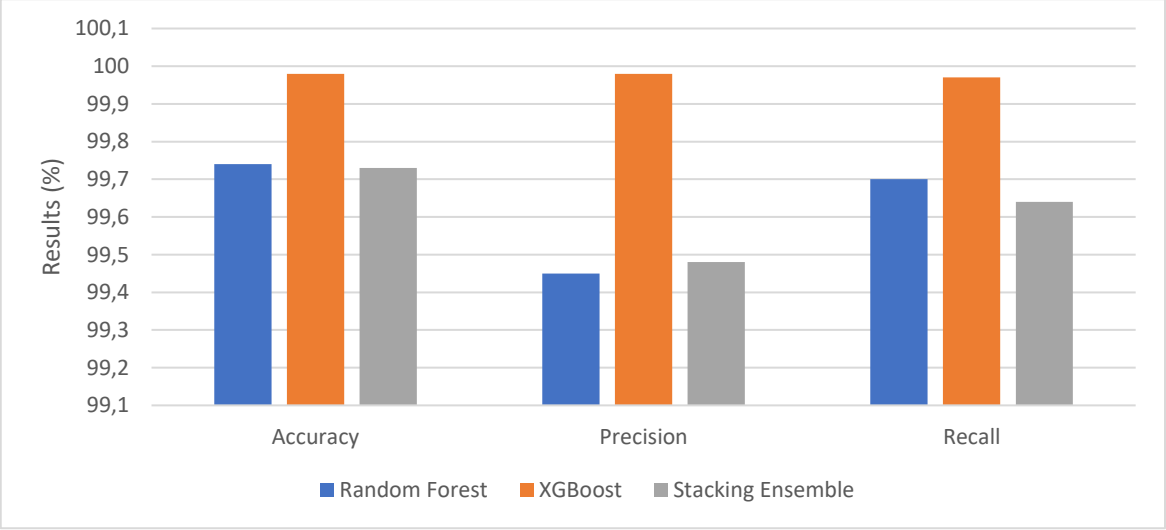


Figure 5. Test Set Results Graph

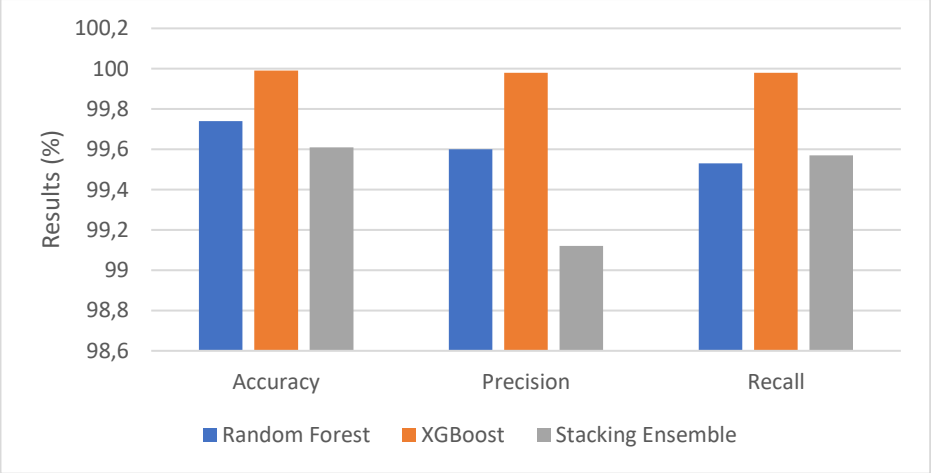
In Table 1, the author compares each evaluation metric on the ensemble method models for predicting flight delays using the test set. In these results, the XGBoost has the best accuracy,

precision, and recall values than other ensemble methods. While the Random Forest model has better accuracy and recall values than the stacking ensemble method. The Stacking Ensemble method has better precision value than the Random Forest model.

Accuracy, Precision, and Recall Values On Validation Set (Feature Selection)

Table 2. Result Comparison On Validation Set

Evaluation Metrics	Random Forest	XGBoost	Stacking Ensemble
Accuracy	99.74%	99.99%	99.61%
Precision	99.6%	99.98%	99.12%
Recall	99.53%	99.98%	99.57%
Speed	10 minutes 12 seconds	1 minutes 12 seconds	28 minutes 48 seconds



In figure 6, it is clear that the XGBoost model outperforms other ensemble methods on predicting flight delays using the validation set. Then, both the Random Forest model and the Stacking Ensemble model perform similarly on the three metrics. In table 2, the Random Forest model is slightly better than the stacking ensemble method, but because of the faster computational speed, Random Forest is better both at prediction and speed than the stacking ensemble method.

Cross Validation Values On Train and Test Set (Feature Selection)

Table 3. Cross Validation Accuracy Comparison

Split Type	Random Forest	XGBoost	Stacking Ensemble
Train Set	99.76%	99.98%	99.67%
Test Set	99.64%	99.97%	99.54%

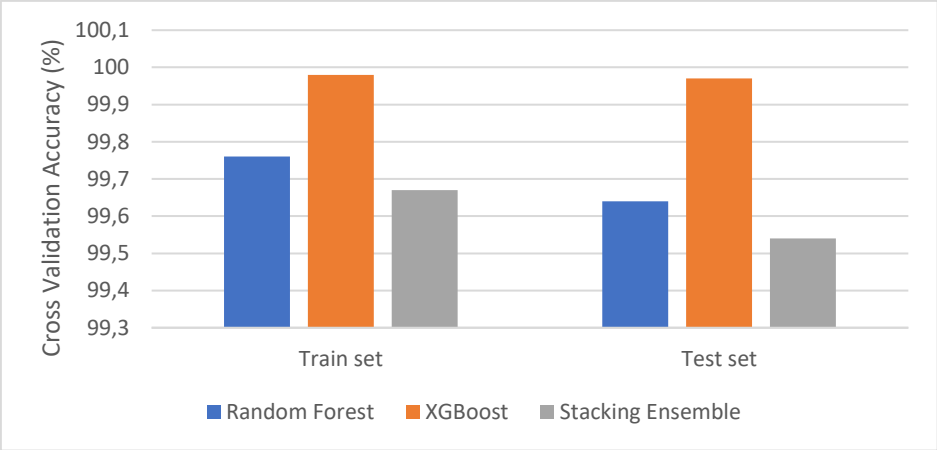


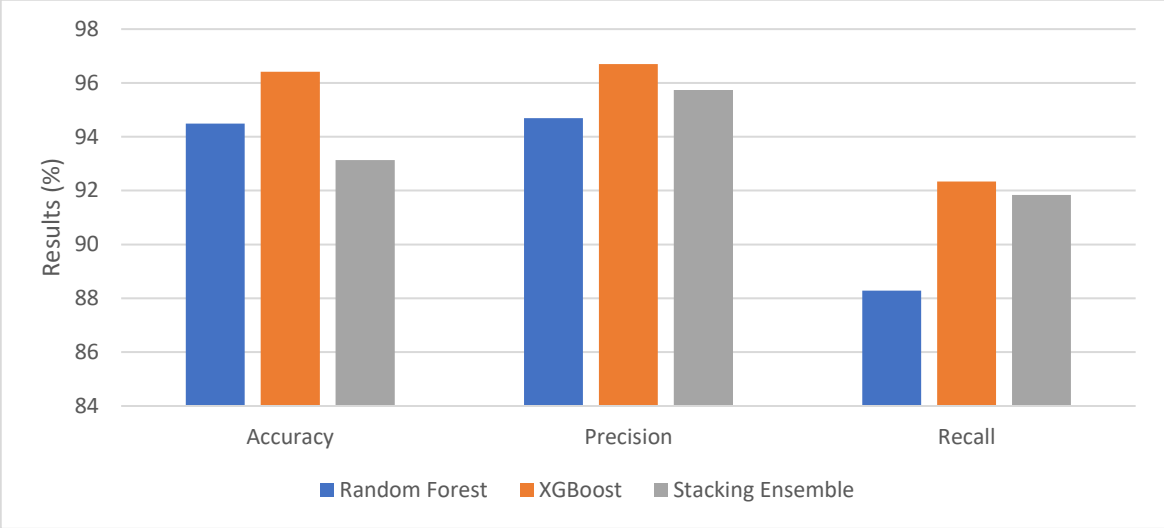
Figure 7. Cross Validation Accuracy Comparison Graph

In table 3, the author compares the cross-validation accuracy performance of the three ensemble method models. This is done using both the train and test datasets split. In figure 7, it is clear that the XGBoost is outperforming the other ensemble method models with an almost perfect score. Then, the Random Forest model is also a little bit ahead of the Stacking Ensemble method, with a margin of 0.02% for the train set and 0.03% for the test set.

Accuracy, Precision, and Recall Values On Test Set (PCA)

Table 4. Result Comparison On Test Set (PCA)

Evaluation Metrics	Random Forest	XGBoost	Stacking Ensemble
Accuracy	94.49%	96.42%	93.14%
Precision	94.69%	96.7%	95.74%
Recall	88.29%	92.34%	91.84%



In table 4, the results of the models are taken from the test set using PCA as the dimensional reduction method. Here, the best-performing model is still XGBoost, but the results decreased from the test set using feature selection as the dimensional reduction method, as shown in table 1. Other than that, the second-best performing model is the Stacking Ensemble model, but this model has a lot of decrease in performance results compared to the results in Table 1, with the precision value being the most decreased value from 99.24% to 91.84%. Lastly, the Random Forest model comes in third, with only a better-performing accuracy value than the Stacking Ensemble method.

Accuracy, Precision, and Recall Values On Validation Set (PCA)

Table 5. Result Comparison On Validation Set (PCA)

Evaluation Metrics	Random Forest	XGBoost	Stacking Ensemble
Accuracy	94.67%	94.67%	96.00%
Precision	91.84%	95.56%	95.74%
Recall	91.84%	87.76%	91.84%
Speed	9 minutes 45 seconds	1 minutes 10 seconds	25 minutes 16 seconds

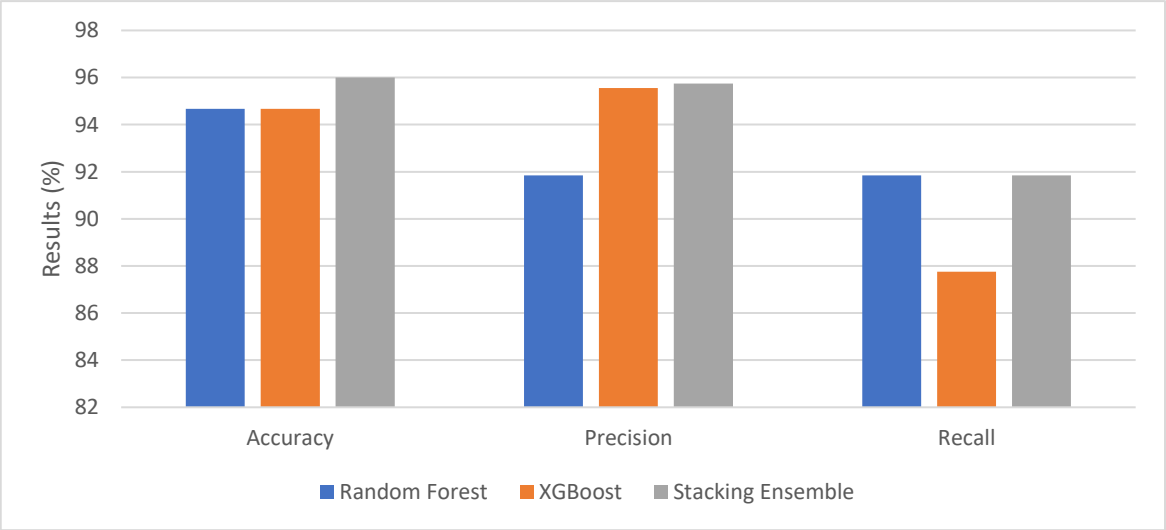


Figure 9. Validation Set Results Graph (PCA)

Based on figure 9, the best performing model with PCA dimension reduction in the validation set is the Stacking Ensemble method, with mean results of more than 94.5% compared to other models. Then, the XGBoost and Random Forest models have similar accuracy value results, but here we can see that the XGBoost model has a higher precision score than the Random Forest model. In contrast, the Random Forest model has a better recall value than the XGBoost model. Other than that, the XGBoost model actually has the lowest recall values than all the models. But out of all the models, the XGBoost model is the fastest model to compute the outputs.

Cross Validation Values On Train and Test Set (PCA)

Table 6. Cross Validation Accuracy Comparison (PCA)

Split Type	Random Forest	XGBoost	Stacking Ensemble
Train Set	96.76%	97.62%	96.44%
Test Set	92.85%	94.63%	92.85%

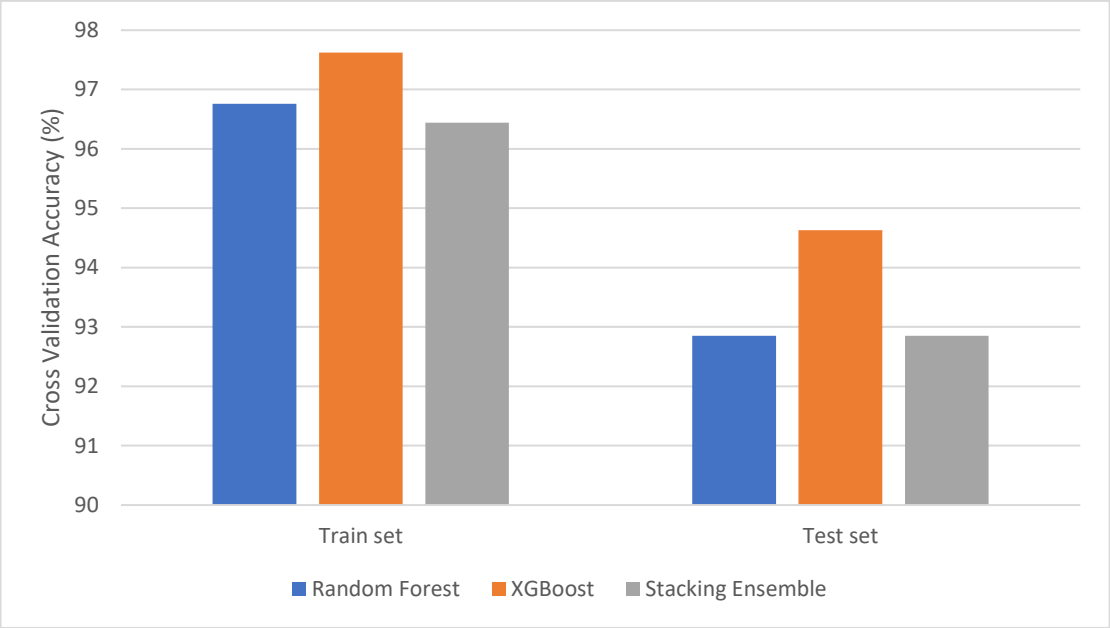


Figure 10. Cross Validation Accuracy Comparison Graph (PCA)

In table 6, the results of cross-validation accuracy for both the train set and test set using PCA as the dimensional reduction are compared. Here, the XGBoost model outperforms every other model used with a really high accuracy on both the test and train set with a value of 97.62%. While the stacking ensemble method and the Random Forest model have relatively the same results on both the training and test sets. Compared to the cross-validation accuracy results using feature selection as in table 3, every model gets a decrease in accuracy, proving that these models work better on feature selection method as the dimensional reduction method.

Accuracy Results Number of Features Used Experiments (Feature Selection)

Table 7. Feature Selection Experiment Accuracy Values Validation Set

Number Of Features	Random Forest	XGBoost	Stacking Ensemble
10	99.99%	99.99%	99.97%
20	99.97%	99.99%	99.95%
30	99.93%	99.99%	99.6%
40	99.74%	99.99%	99.61%
50	99.81%	99.99%	99.4%
60	99.54%	99.96%	99.13%
70	99.5%	99.95%	99.06%

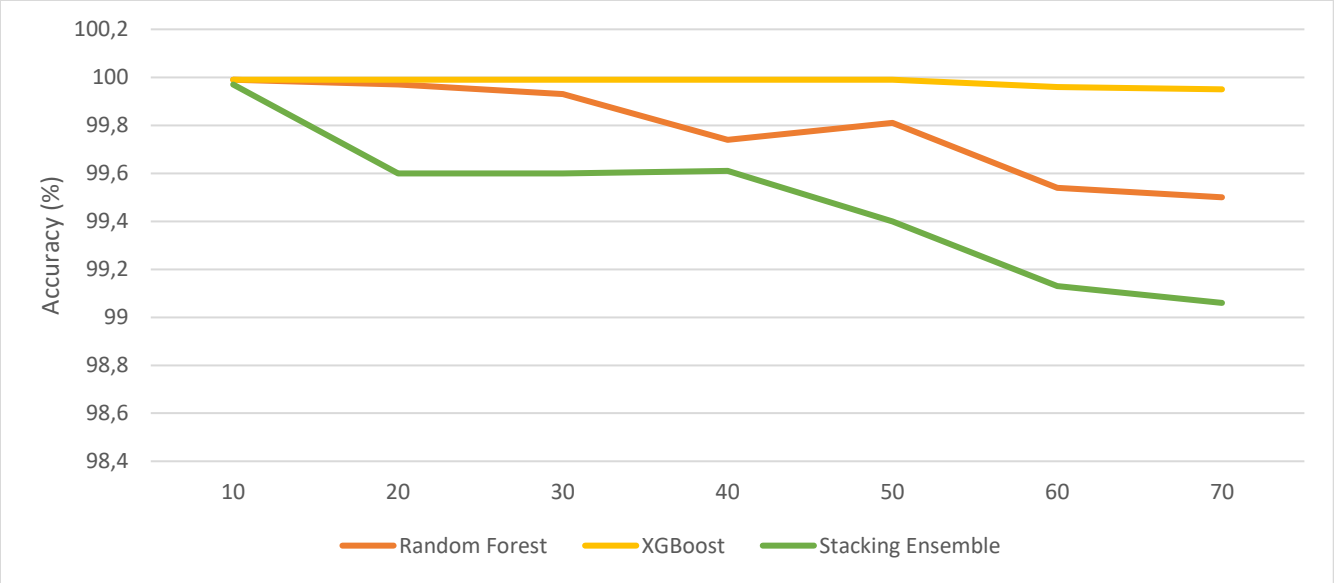


Figure 11. Number Of Features Experiment On Validation Set Accuracy Comparison Graph (Feature Selection)

Based on Figure 11, it shows that the accuracy decreases as the number of features used increases. This means that even with more features or information, it does not always mean that it can make the accuracy better. does not always mean that it can make the accuracy higher. Meanwhile, the XGBoost model's accuracy seems to remain constant at around 99.95%, this happens likely due to the model's robust handling of important features. Therefore, the optimal number of features is around 30 - 40 features.

Accuracy Results Number of Features Used Experiments (PCA)

Table 8. PCA Experiment Accuracy Values Validation Set

Cumulative Variance	Number of Features	Random Forest	XGBoost	Stacking Ensemble
0.8	22	94.18%	97.04%	93.7%
0.85	25	93.48%	96.9%	93.36%
0.9	29	93.84%	97.04%	93.3%
0.95	35	92.12%	97.01%	92.29%
1	58	97.06%	99.32%	97.05%

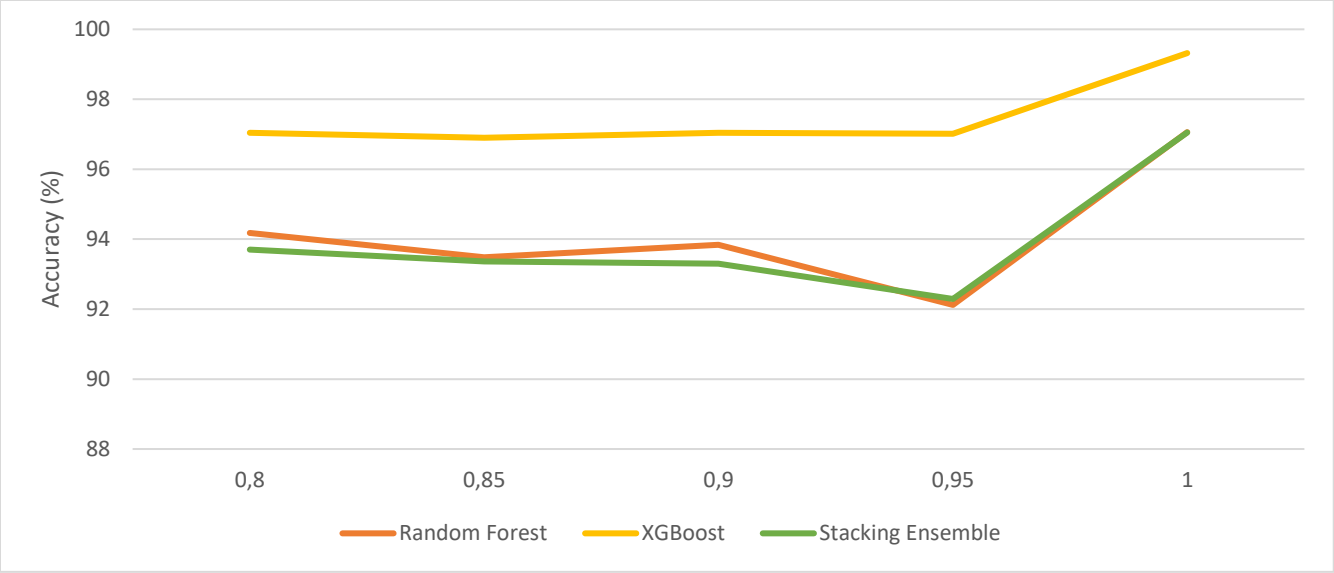


Figure 12. Number Of Features Experiment On Validation Set Accuracy Comparison Graph (PCA)

Based on figure 12, it shows that the accuracy decreases on all models when the number of features used increases, except for when the cumulative variance is kept at 100%. This is likely because the number of features decreased immensely when using PCA, while at 100% cumulative variance it gives more features and information but slows down the computational speed. The most consistent model is the XGBoost model, with a mean accuracy of 97.42% on all trials. Therefore, the optimal cumulative variance for this is the 100% cumulative variance.

Discussion

In this project, the author used 50% splitting and 50% testing because, after the split and test ratio, the 50% splitting and 50% testing ratio generates the best cross-validation accuracy results for both the Random Forest and XGBoost models. Other than that, the author also uses SMOTE because, after observing how the dataset is distributed, it is imbalanced which could result in overfitting or underperformance.

Feature selection reduces the amount of features used from 85 features to 40 features, which is more than half of it. In the feature selection results, it is clear that the XGBoost model is the best performing model out of all the models tested, with results more than 99.9% and a very fast computational speed of only 1 minute and 12 seconds. The second best performing model is the Random Forest model. While the stacking ensemble method has a very minimal decrease in results with more than 99.5%, the amount of time needed for this model to compute is 25 minutes and 16 seconds. These results can lead to overfitting, but the author has tried to prevent and prove that it is not overfitting by doing cross-validation, using multiple preprocessing methods, balancing the dataset, and comparing the results with the validation set, which has been split firstly before the model is fitted.

PCA in this project reduces the amount of features used from 85 to 58, which is 18 features more than the feature selection method. In the PCA results, it is not as good as the feature selection results. The XGBoost model is still the best performing model, and it only needs 1 minute and 10 seconds to compute, 2 seconds faster than using the feature selection method. Then, the Stacking Ensemble method model actually has a much better performance than all the models in the validation set results when using PCA. With PCA, the results of the models tend to not overfit with a slightly faster computational speed; this proves that by using PCA, it can help reduce the model's complexity.

Every model gets a slight decrease of around 2-4% of the mean accuracy results when using PCA as the dimensional reduction method compared to feature selection, which means that the model works better on more features. Other than that, the computational speed on both dimensional reduction methods doesn't have a big difference despite having a smaller number of features.

CONCLUSION

Based on the results of this project, the best-performing algorithm with the highest accuracy, precision, and recall values is the XGBoost model. That model works well when using both the feature selection and PCA as the dimensionality reduction methods. But there is some reduction in performance when using PCA as the dimensionality reduction rather than the feature selection method; this is because the PCA has a lower number of features compared to the feature selection method. Furthermore, the XGBoost model is more suitable to be used in real-time since the computational speed of the model is only above 1 minute, when the other models' computational speed is longer than 9 minutes. On the other hand, the stacking ensemble method was the worst performing model based on the accuracy, recall, and precision values, and the computational speed was the slowest. This happens because the stacking ensemble method is more complex and needs to fit 2 models, making it slower. Other than that, the best model based on the cross-validation accuracy is the XGBoost model on both the PCA and feature selection methods.

Furthermore, based on the cross-validation results, the best-performing model is also the XGBoost model, achieving more than 95% accuracy in both dimensional reduction methods and both the test and train sets. In contrast, the stacking ensemble method performs the worst in terms of accuracy and computational time. The best dimensionality reduction method for this particular project is the feature selection method. This is because the feature selection contains values of the features that are purely from the preprocessing steps, which helps the models to capture the correlations of each feature easily. Meanwhile, the PCA method is better at reducing the dimensionality of the dataset, making the model less complex, which can be seen from the results where it improves the computational speed of each model. However, by using the PCA method, every model experienced a decrease in performance accuracy, recall, and precision.

For the next research, the author suggests that other individual models can be tested and compared to the stacking ensemble method with other types of dimensionality reduction methods.

Also, the author suggests that other preprocessing methods can be tested so that the model can be optimized further.

DAFTAR PUSTAKA

[1] IATA Sustainability & Economics. Air Passenger Market Analysis January 2024 Resilient industry-wide growth brings global traffic to near recovery. *IATA*, https://www.iata.org/en/iata-repository/publications/economic-reports/air-passenger-market-analysis-january-2024/ (2024).

[2] Lu M, Wei P, He M, et al. Flight Delay Prediction Using Gradient Boosting Machine Learning Classifiers. https://doi.org:/10.32604/jqc.2021.016315

[3] Airline Flight Delay Prediction Using Machine Learning Models, https://dl.acm.org/doi/fullHtml/10.1145/3497701.3497725 (accessed 4 April 2024).

[4] Seongeun Kim EP. Prediction of flight departure delays caused by weather conditions adopting data-driven approaches. *Journal of Big Data*, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00867-5 (2024, accessed 4 April 2024).

[5] Mienye ID, Sun Y. A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access* 2022; 10: 99129–99149. https://doi.org/10.1109/access.2022.3207287

[6] Li Y, Chen W. A Comparative Performance Assessment of Ensemble Learning for Credit Scoring. Mathematics 2020; 8: 1756. https://doi.org/10.3390/math8171756.

[7] Wang X, Wang Z, Wan L, et al. Prediction of Flight Delays at Beijing Capital International Airport Based on Ensemble Methods. *Applied Sciences* 2022; 12: 10621. https://doi.org/10.3390/app122110621.

[8] Sahoo R, Pasayat AK, Bhowmick B, et al. A hybrid ensemble learning-based prediction model to minimise delay in air cargo transport using bagging and stacking. *International Journal of Production Research* 2022; 60: 644–660. https://doi.org/10.1080/00207543.2021.1915196.

[9] Horiguchi Y, Baba Y, Kashima H, et al. Predicting Fuel Consumption and Flight Delays for Low-Cost Airlines. *Proceedings of the AAAI Conference on Artificial Intelligence* 2017; 31: 4686–4693. https://doi.org/10.1609/aaai.v31i1.11332

[10] Zhang Y, Liu J, Shen W. A Review of Ensemble Learning Algorithms Used in Remote Sensing Applications. *Applied Sciences* 2022; 12: 8654. https://doi.org/10.3390/app12218654.

[11] Lambelho M, Mitici M, Pickup S, et al. Assessing strategic flight schedules at an airport using machine learning-based flight delay and cancellation predictions. *Journal of Air*

Transport Management 2020; 82: 101737. https://doi.org/10.1016/j.jairtraman.2019.101737.

[12] Cosmas Haryawan, Yosef Muria Kusuma Ardhana. ANALISA PERBANDINGAN TEKNIK OVERSAMPLING SMOTE PADA IMBALANCED DATA. *JIRE* 2023; 6: 73–78. https://doi.org/10.1016/j.jire.2023.01.009.

[13] Nasution MZ. PENERAPAN PRINCIPAL COMPONENT ANALYSIS (PCA) DALAM PENENTUAN FAKTOR DOMINAN YANG MEMPENGARUHI PRESTASI BELAJAR SISWA (Studi Kasus : SMK Raksana 2 Medan). *JurTI (Jurnal Teknologi Informasi)* 2019; 3: 41–48. https://doi.org/10.29303/jurti.v3i1.18.

[14] H. A. KRISTANTO, "AIRPORT WEATHER INFORMATION SYSTEM IN INDONESIAN TO PREDICT FLIGHT DELAY," other, Prodi Ilmu Komputer Unika Soegijapranata, 2013. Accessed: Jan. 03, 2025. [Online]. Available: https://repository.unika.ac.id/3511/

[15] W. SOESANTIO, "Comparing Random Forest Algorithm and Support Vector Machine for Predicting the Level of Satisfaction with Flights," other, Universitas Katholik Soegijapranata Semarang, 2022. Accessed: Jan. 03, 2025. [Online]. Available: https://repository.unika.ac.id/30029/