Supervised Learning Assignment

Honya Elfayoumy
CS 7641

Abstract—I used two datasets for supervised learning and evaluation. The first dataset was for heart disease prediction, and the second was for predicting hotel reservations. The datasets were split into training and testing sets, and various supervised learning algorithms were trained and evaluated on them. Overall, the results of the study highlight the importance of careful evaluation and selection of algorithms for supervised learning problems, as the performance of different algorithms can vary greatly depending on the specific problem and dataset.

I. Introduction to Datasets

A. Heart Disease Dataset

The Heart Disease Data Set [1] from the UCI Machine Learning Repository is a dataset that contains information about patients diagnosed with heart disease. The dataset contains information on a group of patients who have been diagnosed with heart disease and includes a range of demographic and medical information about each patient. The demographic information included in the dataset is fairly basic, and includes the patient's age and sex. The medical information, on the other hand, is more comprehensive and includes various measurements taken from each patient, such as cholesterol levels, blood pressure readings, and electrocardiogram results.

The features include a combination of continuous and binary attributes, as some of the attributes have been transformed into binary format through One-Hot encoding. The target variable for classification is binary in nature. The target variable in this dataset is the presence of heart disease, which is binary and can be either 0 (no heart disease) or 1 (heart disease present). This dataset is commonly used as a benchmark for machine learning algorithms in the field of cardiovascular disease prediction and is a great resource for anyone looking to gain experience working with medical data or building predictive models for healthcare applications. The correlation between different attributes is shown in Figure 1. I can see outliers in some of the columns as shown in 2.

B. Hotel Reservation Dataset

The Hotel Reservations dataset for the classification task by Kaggle is a subset of the larger Hotel Reservations dataset that is focused on solving a binary classification problem. The problem is to predict whether a hotel room reservation will be canceled or not based on the features of the reservation such as the arrival date, the number of adults, the number of children, the number of babies, the meal type, the country of origin, the deposit type, and others. The correlation between different attributes is shown in Figure 4.

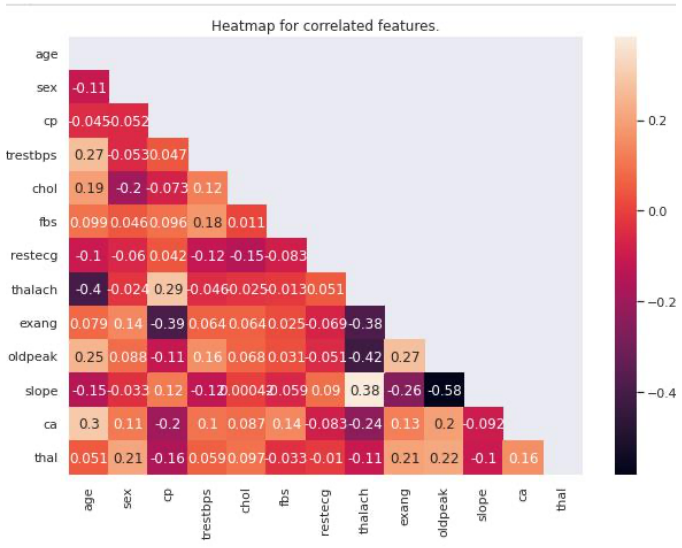


Fig. 1: Heatmap for correlated features for Heart Disease Dataset

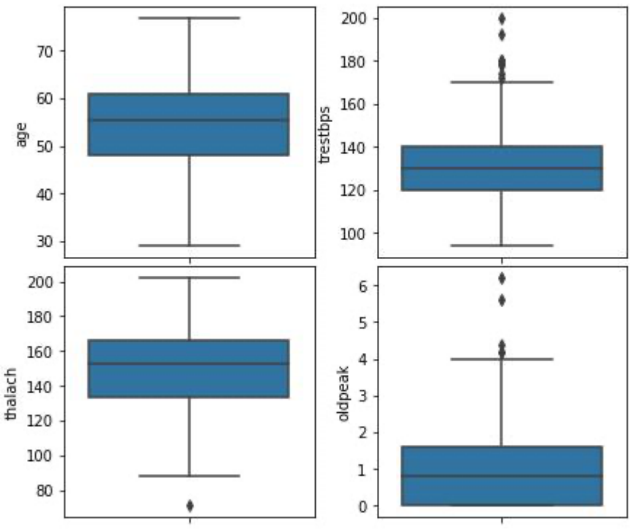


Fig. 2: Boxplot for checking outliers for Heart Disease Dataset

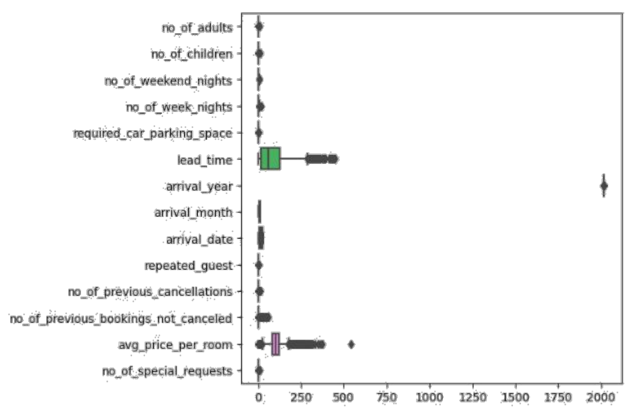


Fig. 3: Boxplot for checking outliers for Hotel Reservation Dataset

The data consists of instances, each representing a single hotel booking, and the target variable is the "is-canceled" column, which indicates whether a booking was canceled (1) or not (0). The data includes various features such as customer information, booking information, and arrival and departure dates.

This dataset can be used for supervised learning tasks, specifically classification problems, where the goal is to build a model that can predict whether a booking will be canceled or not based on the available features. The data provides a valuable resource for data scientists to experiment with various machine learning algorithms and feature engineering techniques and evaluate their performance on this classification task. I also checked the outliers in data as shown in Figure 3. I can see outliers in some of the columns, like lead time and average rice per room.

II. Pre-Processing of Datasets

Both datasets underwent a MinMaxScaler normalization process, which scales the features between 0 and 1. Duplicate rows were also removed. The reason for using MinMaxScaler is to ensure that all features are on a similar scale, which can help improve the performance of certain machine-learning algorithms. Some algorithms, such as k-nearest neighbors, are sensitive to the scale of the features and may perform poorly when features are not on a similar scale.

To eliminate multicollinearity among the independent attributes, a threshold of 0.8 was employed. Additionally, to improve the performance of the machine learning classifiers, the Synthetic Minority Oversampling Technique (SMOTE) was applied in cases where a particular class had less representation. This helps balance the class distribution and improve the performance of the machine learning algorithm by ensuring it has enough information to learn the patterns in both classes. For instance, in the heart disease dataset, there were fewer samples for individuals without heart disease. The datasets were divided into two stratified splits, with 80% of the data

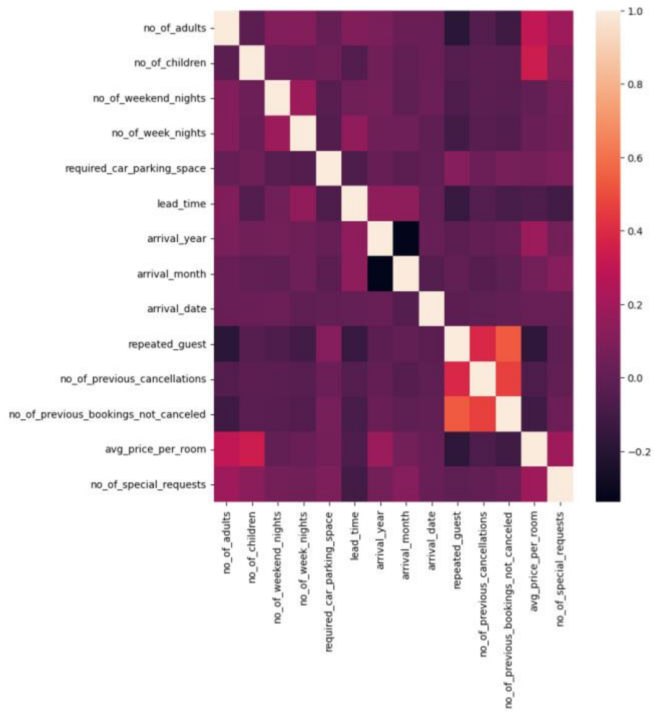


Fig. 4: Heatmap for correlated features for Hotel Reservation Dataset

being used for training and validation and 20% of the data being kept for testing.

III. Decision Trees

A. Heart Disease Dataset

In this experiment, the aim was to study the performance of decision trees on the heart disease dataset using Figure 5, 6, 7, 8 and 9. In the beginning, a decision tree was employed with a maximum depth of the tree. But this resulted in overfitting, as the model performed well on the training data but poorly on the testing data, as shown in Figure 2. Overfitting happens when the tree becomes too complex, and instead of finding patterns, it memorizes noise in the training data, leading to poor performance on unseen data. Using a decision tree, I find the importance of various features 6 as Chest Pain is the most informative feature to predict heart disease, and Fasting blood sugar levels is the least important in predicting heart disease.

To prevent overfitting, a grid search was conducted on the max-depth value to find the optimal depth that resulted in the best accuracy or other performance metrics. The use of SMOTE technique was also implemented to balance the target variable. Additionally, cross-validation was employed to provide a more robust estimate of the model's performance and reduce the risk of overfitting.

As shown in Figure 8, using these techniques improved the performance of the decision tree on the heart disease dataset. The optimal max-depth was found to be 2 for the decision

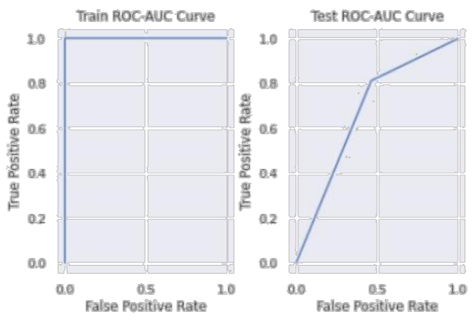
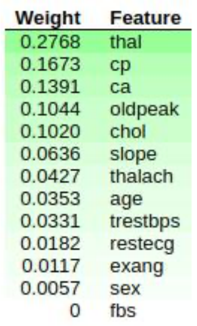


Fig. 5: ROC-AUC with Decision Trees for training and testing partitions on Heart Disease Dataset



Weight	Feature
0.2768	thal
0.1673	cp
0.1391	ca
0.1044	oldpeak
0.1020	chol
0.0636	slope
0.0427	thalach
0.0353	age
0.0331	trestbps
0.0182	restecg
0.0117	exang
0.0057	sex
0	fbs

Fig. 6: Various Features Importance using Decision Trees on Heart Disease Dataset

tree. For example, I achieved a recall of 70.4% on training data and 66.5% on testing data as shown in Figure 8.

B. Hotel Reservation Dataset

In this study, I evaluated the effectiveness of decision trees, specifically random forests, on the hotel reservation dataset. The results are presented in Figures 10, 12, 13 and 14. Initially,

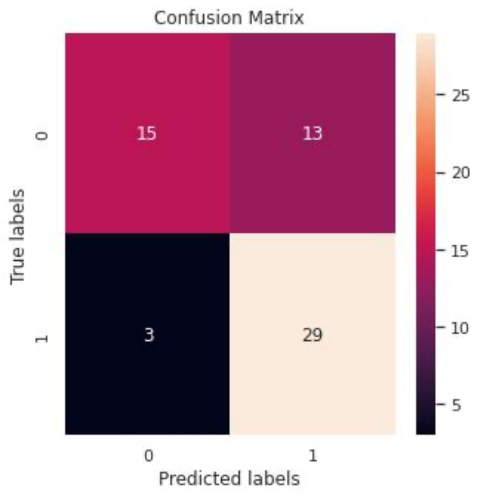


Fig. 7: Confusion matrix with Tuned Decision Trees on Heart Disease Dataset

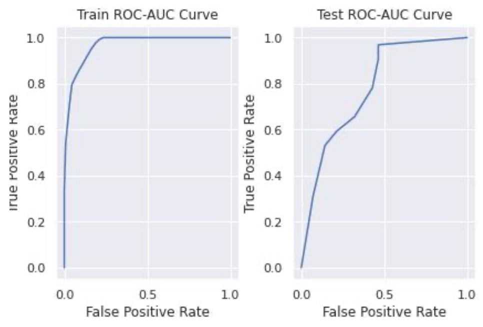


Fig. 8: ROC-AUC with Optimal Decision Trees for training and testing partitions on Heart Disease Dataset

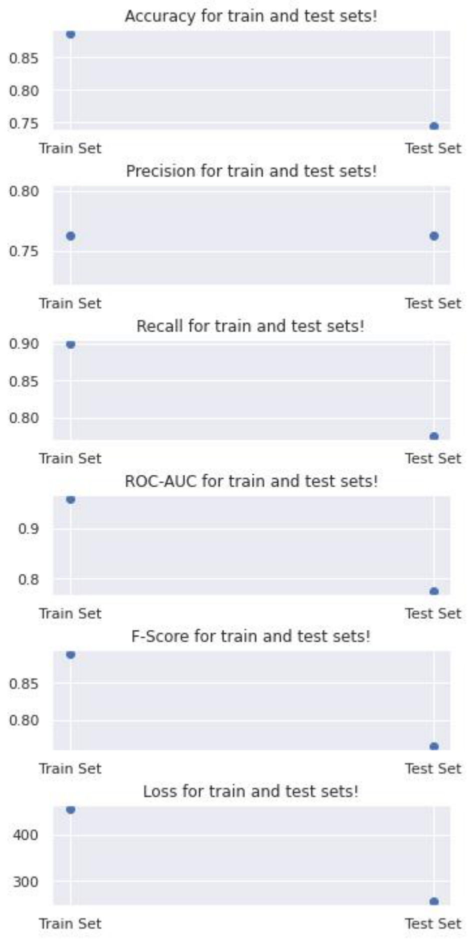


Fig. 9: Various Evaluation Metrics with Tuned Decision Trees on Heart Disease Dataset

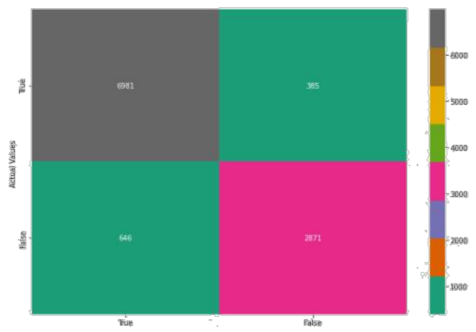
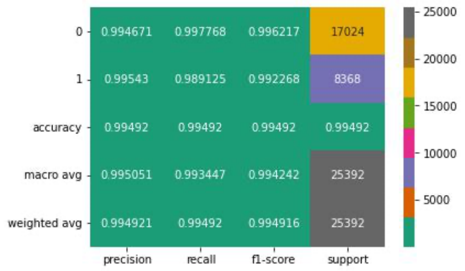


Fig. 10: Confusion matrix with RandomForest with maximum depth on Hotel Reservation Test Dataset



	precision	recall	f1-score	support
0	0.994671	0.997768	0.996217	17024
1	0.99543	0.989125	0.992268	8368
accuracy	0.99492	0.99492	0.99492	0.99492
macro avg	0.995051	0.993447	0.994242	25392
weighted avg	0.994921	0.99492	0.994916	25392

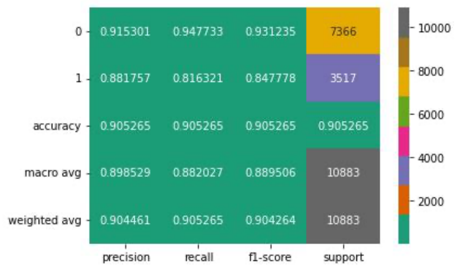
25000
20000
15000
10000
5000

Fig. 11: Various Evaluation Metrics Report with Random-Forest Classifier with maximum depth on Hotel Reservation Training Dataset

I used a random forest classifier with a maximum tree depth of 100 estimators and achieved a macro-averaged recall of around 99% on the training data and 87% on the testing data, as shown in Figures 10 and 11. To enhance the performance, I applied similar techniques used for the heart disease dataset.

The grid search was conducted to identify the optimal values of the hyperparameters, max-depth, minimum sample leaf, and the number of estimators. The aim was to improve the performance of the random forest classifier on the hotel reservation dataset. The max depth determines the depth of each tree in the forest, the minimum sample leaf determines the minimum number of samples required at a leaf node, and the number of estimators sets the number of trees in the forest. These hyperparameters are important in controlling overfitting and enhancing the model's performance.

The experiment results show that tuning the hyperparameters of the decision tree on the hotel reservation dataset did not lead to improved performance, as depicted in Figure 13 and 14. The optimal hyperparameters were determined to be a max-depth of 13, a minimum sample leaf of 20, a minimum sample split of 50, and 60 estimators for the random tree. With these hyperparameters, a macro-averaged recall of 83% was achieved on testing data, as shown in Figure 14. However, the performance could have been better if the grid search had been more extensive, with a greater range of values for the maximum depth and other hyperparameters.



	precision	recall	f1-score	support
0	0.915301	0.947733	0.931235	7366
1	0.881757	0.816321	0.847778	3517
accuracy	0.905265	0.905265	0.905265	0.905265
macro avg	0.898529	0.882027	0.889506	10883
weighted avg	0.904461	0.905265	0.904264	10883

10000
8000
6000
4000
2000

Fig. 12: Various Evaluation Metrics Report with RandomFor-est Classifier with maximum depth on Hotel Reservation Test Dataset

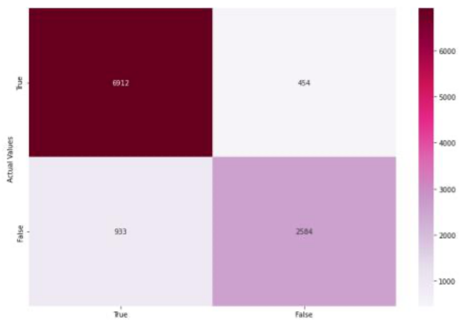
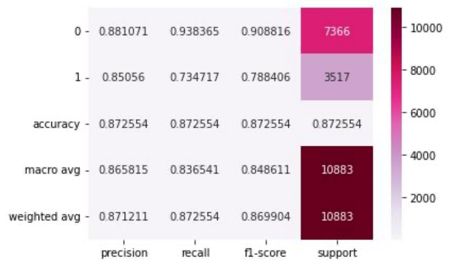


Fig. 13: Hyper-Parameter Tuning with Random Forest: Con-fusion matrix with RandomForest on Hotel Reservation Test Dataset (max-depth=13, min-samples-leaf=20, min-samples-split=50 and n-estimators=60)

IV. Neural Networks

A. Heart Disease Dataset

In this experiment, a neural network consisting of 3 dense layers was utilized. The training process involved 10 epochs with a learning rate of 0.01. It was observed that after 5 epochs, the test accuracy began to decline while the training accuracy continued to rise, as shown in Figure 15. This demonstrates that the model was converging initially. However, it later started to overfit as the performance on training data continued to increase while the performance on testing data



	precision	recall	f1-score	support
0	0.881071	0.938365	0.908816	7366
1	0.85056	0.734717	0.788406	3517
accuracy	0.872554	0.872554	0.872554	0.872554
macro avg	0.865815	0.836541	0.848611	10883
weighted avg	0.871211	0.872554	0.869904	10883

10000
8000
6000
4000
2000

Fig. 14: Hyper-Parameter Tuning with Random Forest: Various Evaluation Metrics Report with RandomForest Classifier on Hotel Reservation Test Dataset (max-depth=13, min-samples-leaf=20, min-samples-split=50 and n-estimators=60)

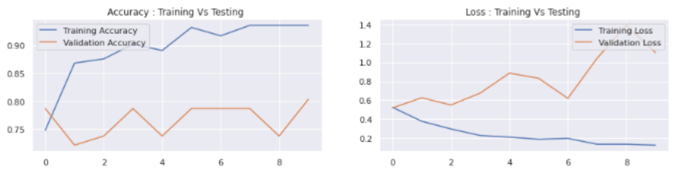


Fig. 15: Accuracy and Loss with Neural Networks on Heart Disease Dataset

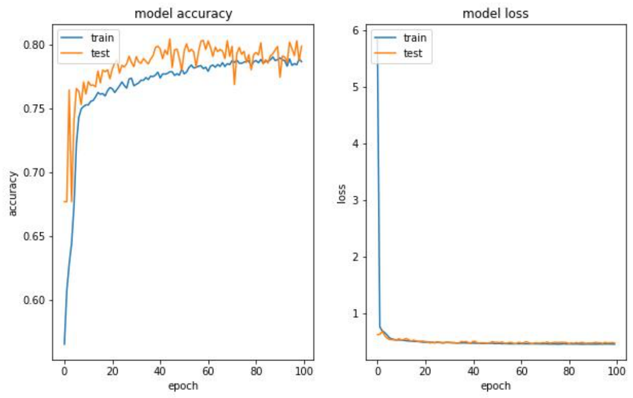


Fig. 16: Accuracy and Loss with Neural Networks on Hotel Reservation Dataset

decreased. The conclusion from these observations is that the training process should be stopped after 5 epochs, or the learning rate should be decreased to obtain a more stable trend.

B. Hotel Reservation Dataset

In this experiment, a neural network consisting of 6 dense layers was used. The accuracy and loss results on dataset are shown in 16. The training process was completed over 100 epochs with a learning rate of 0.001. After around 10 epochs, it was noticed that the test model accuracy began to fluctuate while the training accuracy remained relatively stable. Despite this, the model was able to achieve a small loss after a few epochs.

An interesting observation was made, where test accuracy was found to be better than training accuracy in some cases. This could be due to the fact that the test distribution is easier for the neural network to classify than the training distribution. For example, at around 40 epochs, the model's training accuracy was 78%, and its testing accuracy was 80% as displayed in 16. Further examination using other evaluation metrics, such as the f1-score and precision, would provide a deeper insight.

After 40 epochs, the training should be stopped, as the loss was also small. By using a smaller learning rate, the fluctuations in the training and testing accuracy could be reduced, allowing the model to converge in one direction.

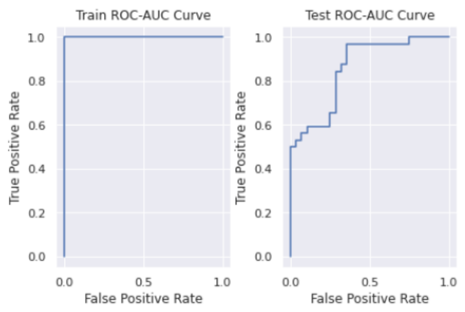


Fig. 17: ROC-AUC with Xgboost for training and testing partitions on Heart Disease Dataset

V. Boosting

A. Heart Disease Dataset

In this section, I used XGBoost (eXtreme Gradient Boosting), which is a powerful open-source machine-learning library for gradient boosting trees. It combines multiple weak decision trees to create a strong predictive model. It is widely used for its robust performance and accuracy and has advanced features such as cross-validation and automatic handling of missing values. The results are shown for XGBoost and Optimized XGBoost in Figure 17, 20, and 21. I also find the feature importance using XGBoost as shown in Figure 19. According to XGBoost as well, Thalassemia and Chest pain is the most important features to predict heart disease, while Resting electrocardiographic is the least informative for the XGBoost classifier.

Initially, I employed the XGBoost algorithm with a maxi-mum depth. My model performed exceptionally well on the training data, achieving a 98.7% recall, as depicted in Figure
18. However, the performance on the testing data was only average, with a recall of 82.9%.

To enhance the performance, I carried out a grid search to determine the optimal values for max-depth, learning rate, and iteration. After conducting a grid search, the optimal values for the XGBoost model on the heart disease dataset were found to be a learning rate of 0.03, a maximum depth of 6, and 100 iterations. These values resulted in a recall of 98.3% on the training data and 83.5% on the testing data, as depicted in Figure 21

Small-depth decision trees are sometimes better than maximum-depth trees because they simplify the model and reduce overfitting, improving interpretability, efficiency, and generalization. However, the optimal depth depends on the data and desired performance, and a maximum depth tree may be necessary in some cases.

B. Hotel Reservation Dataset

In this section, I used different boosting classifiers to improve the performance of the problem on the hotel reservation dataset. I started with the Adaboosting Boosting classifier with maximum depth, but it was not performing well. I varied different parameters like max depth, but I couldn't able to

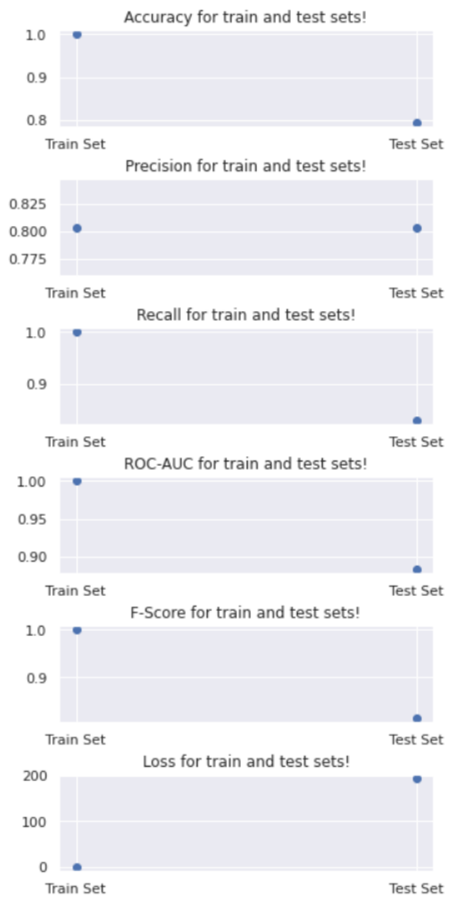
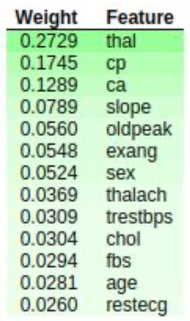


Fig. 18: Various Evaluation Metrics with Xgboost on Heart Disease Dataset



Weight	Feature
0.2729	thal
0.1745	cp
0.1289	ca
0.0789	slope
0.0560	oldpeak
0.0548	exang
0.0524	sex
0.0369	thalach
0.0309	trestbps
0.0304	chol
0.0294	fbs
0.0281	age
0.0260	restecg

Fig. 19: Various Features Importance using Xgboost on Heart Disease Dataset

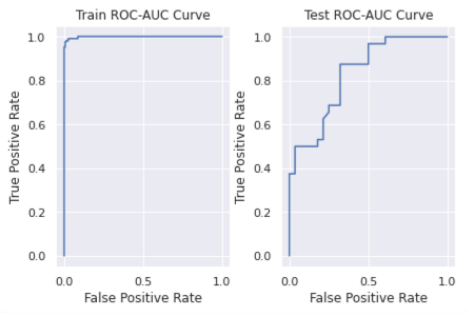


Fig. 20: ROC-AUC with Optimal Xgboost for training and testing partitions on Heart Disease Dataset

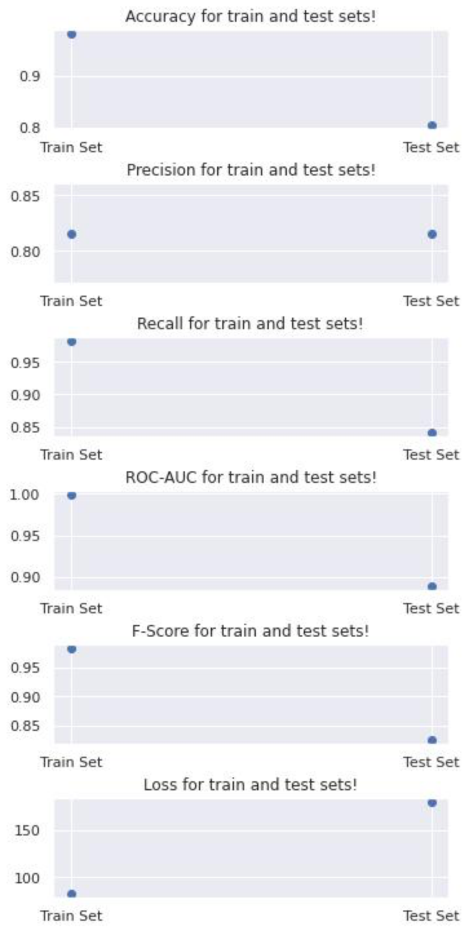


Fig. 21: Various Evaluation Metrics with Tuned Xgboost on Heart Disease Dataset

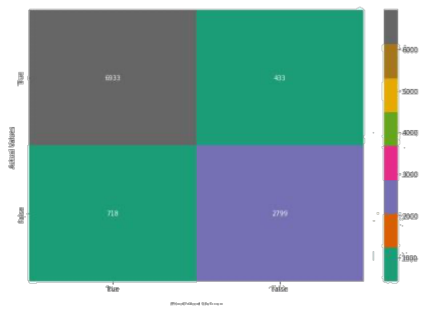
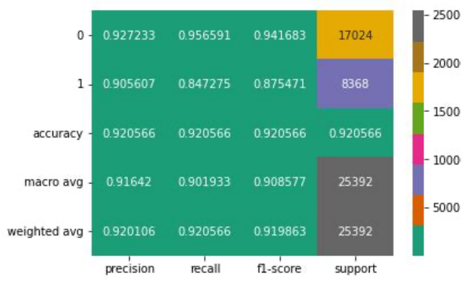


Fig. 22: Confusion matrix with XGBoost with maximum depth on Hotel Reservation Test Dataset



	precision	recall	f1-score	support
0	0.927233	0.956591	0.941683	17024
1	0.905607	0.847275	0.875471	8368
accuracy	0.920566	0.920566	0.920566	0.920566
macro avg	0.91642	0.901933	0.908577	25392
weighted avg	0.920106	0.920566	0.919863	25392

Fig. 23: Various Evaluation Metrics Report with XGBoost Classifier with maximum depth on Hotel Reservation Training Dataset

get good results. It was underfitting; I was able to achieve 77% shown in Figure 26.

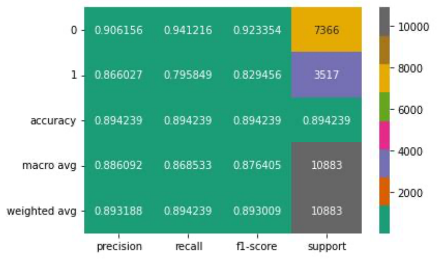
After Ada Boosting, I start doing experiments wth Gradient Boosting. I achieved around 81% but that was not satisfied answer as I can achieve better accuracy using ensemble methods as shown in Figure 25. Gradient Boosting is considered better than AdaBoosting because it optimizes the loss function directly and uses more flexible trees, leading to higher accuracy. It can handle non-linear data better than AdaBoost, which is limited to simple trees and linear decision-making.

After adjusting the hyperparameters, I moved on to using XGBoosting with maximum depth to build the model. XGBoosting outperformed the other ensemble methods and increased accuracy, resulting in 91% accuracy on the training data and 87% on the test data as shown in Figure 23 and 24. XGBoosting is superior to AdaBoosting and Gradient Boost-ing because it is faster in training and has better performance due to its advanced tree-boosting algorithm and parallel pro-cessing capabilities. XGBoost also has the capability to deal with missing values and can handle large datasets efficiently, as well as having built-in regularization parameters to avoid overfitting.

VI. Support Vector Machines

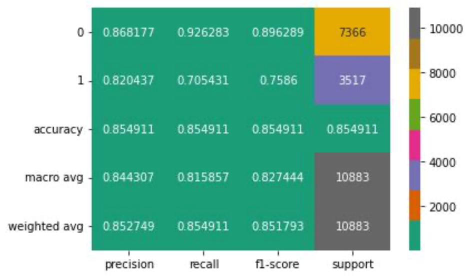
A. Heart Disease Dataset

In this section, I employed Support Vector Machines (SVM) with a radial basis function (RBF) kernel on the heart disease dataset, as depicted in figure 27. It's good because it maps



	precision	recall	f1-score	support
0	0.906156	0.941216	0.923354	7366
1	0.866027	0.795849	0.829456	3517
accuracy	0.894239	0.894239	0.894239	0.894239
macro avg	0.886092	0.868533	0.876405	10883
weighted avg	0.893188	0.894239	0.893009	10883

Fig. 24: Various Evaluation Metrics Report with XGBoost Classifier with maximum depth on Hotel Reservation Test Dataset



	precision	recall	f1-score	support
0	0.868177	0.926283	0.896289	7366
1	0.820437	0.705431	0.7586	3517
accuracy	0.854911	0.854911	0.854911	0.854911
macro avg	0.844307	0.815857	0.827444	10883
weighted avg	0.852749	0.854911	0.851793	10883

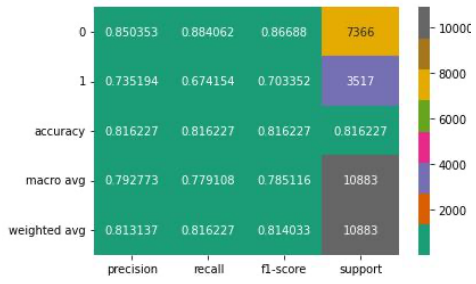
Fig. 25: Confusion matrix with GradientBoosting with maxi-mum depth on Hotel Reservation Test Dataset

the data into a high-dimensional feature space, which makes it possible to find the best boundary even when the data is not linearly separable. Additionally, the RBF kernel is flexible and can adapt to different shapes of data distributions, resulting in improved classification accuracy. The results showed that the model had a recall of 92.3% on the training data and 87.2% on the testing data.

.

B. Hotel Reservation Dataset

In this section, I utilized Support Vector Machines (SVM) with a radial basis function (RBF) kernel on the hotel reservation dataset, as depicted in the figure 28, 29, 31, and 32. The RBF kernel has the advantage of being flexible and



	precision	recall	f1-score	support
0	0.850353	0.884062	0.86688	7366
1	0.735194	0.674154	0.703352	3517
accuracy	0.816227	0.816227	0.816227	0.816227
macro avg	0.792773	0.779108	0.785116	10883
weighted avg	0.813137	0.816227	0.814033	10883

Fig. 26: Various Evaluation Metrics Report with AdaBoosting Classifier with maximum depth on Hotel Reservation Training Dataset

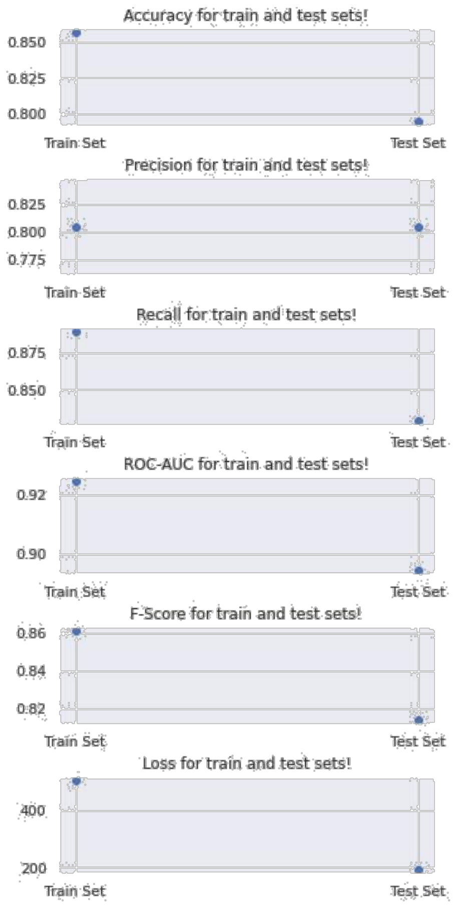


Fig. 27: Various Evaluation Metrics with SVM on Heart Disease Dataset

adapting to various shapes of data distributions, leading to enhanced classification accuracy. The results showed that the model had a recall of 75.2% on the test data as displayed in reffig:svm3hotel.

I also explored the effect of varying the regularization parameter C on model performance, as shown in Figure as shown in 31. The regularization parameter C determines the balance between a low training error and a low testing error. I discovered that a smaller C value creates a broader margin, allowing more observations to be misclassified, but produces a model with lower variance. Conversely, a larger C value creates a narrower margin and results in a model with higher variance and fewer misclassified observations. It is crucial to vary the C value to find the optimal balance and avoid overfitting or underfitting in SVM models. With grid search, I was able to find the optimal parameters where the evaluation metrics produced good results. Although the gain in performance was not substantial, I was able to achieve approximately 76% accuracy on the test data with (C=1, gamma=0.01,kernel='rbf') as shown in 32.

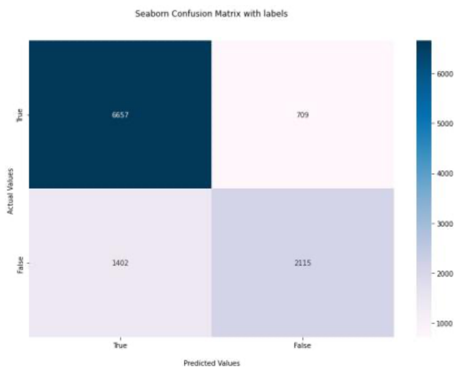
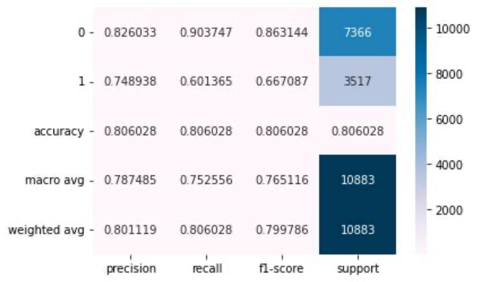


Fig. 28: Confusion matrix with SVC on Hotel Reservation Test Dataset



	precision	recall	f1-score	support
0	0.826033	0.903747	0.863144	7366
1	0.748938	0.601365	0.667087	3517
accuracy	0.806028	0.806028	0.806028	0.806028
macro avg	0.787485	0.752556	0.765116	10883
weighted avg	0.801119	0.806028	0.799786	10883

10000
8000
6000
4000
2000

Fig. 29: Various Evaluation Metrics Report with SVC on Hotel Reservation Training Dataset

VII. K-Nearest Neighbors

A. Heart Disease Dataset

In this study, I utilized the K-nearest neighbors algorithm to make predictions about heart disease. The K-NN algorithm is effective in situations where the data has high dimensions, small sample size, non-linear boundaries, unknown distributions, and label noise. The value of k in K-NN can be determined using cross-validation, the elbow method, resampling techniques, or by relying on expert knowledge. I used the elbow method, which involves plotting the accuracy of the classifier against the number of neighbors. The optimal value for k is determined as the point where the accuracy levels off.

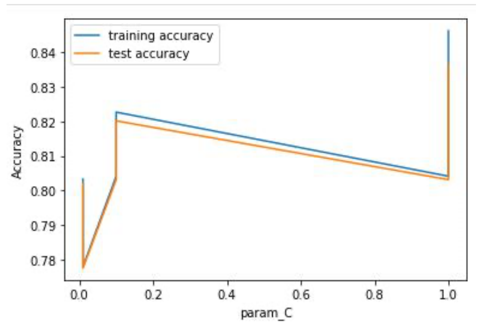


Fig. 30: Accuracy vs. Regularization parameteR - C in SVC classifier on Hotel Reservation Test Dataset

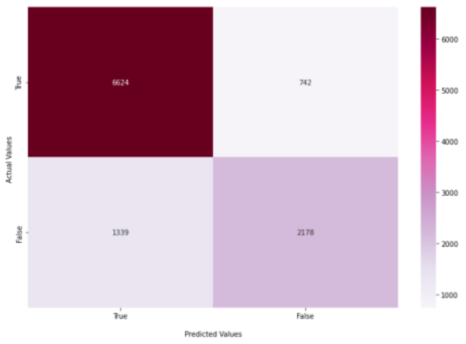
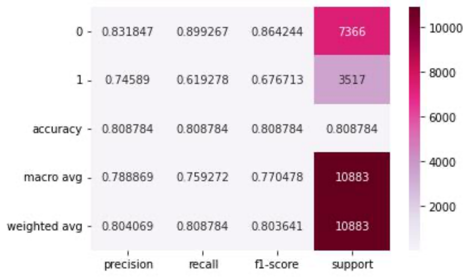


Fig. 31: Confusion matrix with SVC SVC (C=1, gamma=0.01,kernel=’rbf’) on Hotel Reservation Test Dataset



	precision	recall	f1-score	support
0	0.831847	0.899267	0.864244	7366
1	0.74589	0.619278	0.676713	3517
accuracy	0.808784	0.808784	0.808784	0.808784
macro avg	0.788869	0.759272	0.770478	10883
weighted avg	0.804069	0.808784	0.803641	10883

Fig. 32: Various Evaluation Metrics Report with SVC (C=1, gamma=0.01,kernel=’rbf’) on Hotel Reservation Training Dataset

When I used K=2, the performance was the worst, with a 65% recall rate on the testing data. When I increased the K value to K=3, I observed an improvement with a recall rate of 81% on the testing partition. However, as I continued to increase the K value to K=4, the performance started to deteriorate, as seen in Figure 33, 35, and 36.

B. Hotel Reservation Dataset

In this experiment, I made predictions regarding hotel reservation data using the K-nearest neighbors technique as shown in Figure 37, 38, and 39. Figure 37 illustrates how I

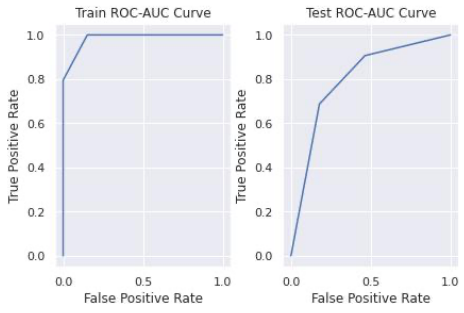


Fig. 33: ROC-AUC with KNN (K=2) for training and testing partitions on Heart Disease Dataset

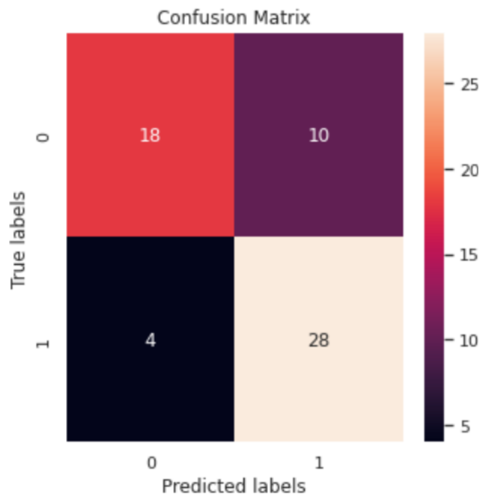


Fig. 34: Confusion matrix with KNN (K=3) on Heart Disease Dataset

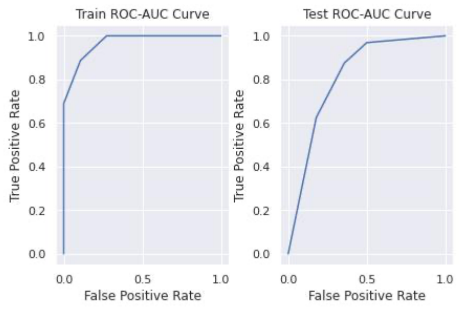


Fig. 35: ROC-AUC with KNN (K=3) for training and testing partitions on Heart Disease Dataset

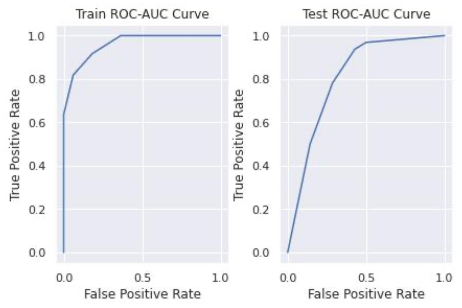


Fig. 36: ROC-AUC with KNN (K=4) for training and testing partitions on Heart Disease Dataset

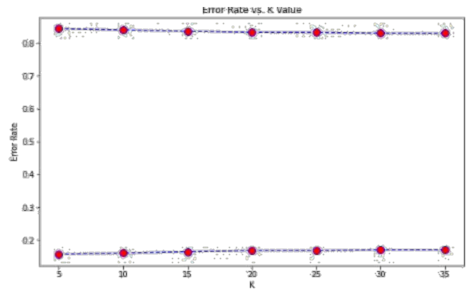


Fig. 37: K-Elbow Method on Hotel Reservation Training Dataset

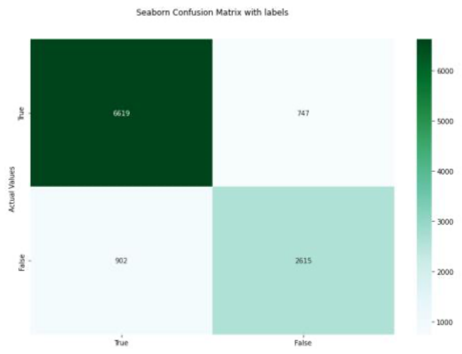


Fig. 38: Confusion matrix with KNN with K=5 on Hotel Reservation Test Dataset

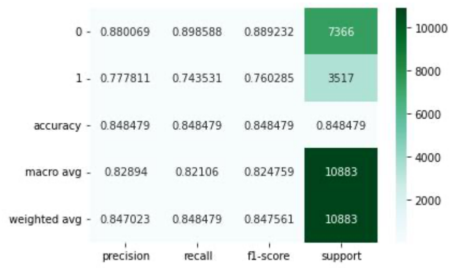
determine the optimal K value using the K-elbow approach. I applied the elbow approach, which involves graphing the classifier's accuracy vs. the number of neighbors. The point at which the accuracy plateaus is used to estimate the ideal value for k. The ideal K value in My situation is 5, where accuracy is high, and errors are few. I observe that a large value of K in the K-Nearest Neighbors (KNN) algorithm can result in a high number of noisy or irrelevant data points affecting the classification decision. When K is large, it increases the risk of overfitting and makes the model less flexible. Additionally, with a large K value, the KNN algorithm may take longer to compute the nearest neighbors and make predictions.

Similarly, as shown in Figure 37, a smaller K value is generally preferred because it allows the model to better capture the local structure of the data and make predictions based on the most similar neighbors. However, a smaller K value also increases the risk of overfitting and making decisions based on noisy or outlier data points. Therefore, finding the optimal K value that strikes a balance between overfitting and underfitting is important for the KNN algorithm.

The accuracy rapidly declines as I raise the K value, as seen by the K-elbow plot. Figure 39 illustrates my 82% accuracy with K=5 in this situation.

VIII. Comparison of Classifiers

For the heart disease dataset, some algorithms may perform better than others, depending on the specifics of the dataset and the problem being solved. A decision tree may be a good



	precision	recall	f1-score	support
0	0.880069	0.898588	0.889232	7366
1	0.777811	0.743531	0.760285	3517
accuracy	0.848479	0.848479	0.848479	0.848479
macro avg	0.82894	0.82106	0.824759	10883
weighted avg	0.847023	0.848479	0.847561	10883

Fig. 39: Various Evaluation Metrics Report with KNN with K=5 on Hotel Reservation Training Dataset

starting point due to its ease of interpretation and visualization, but it may also overfit. An XGBoost algorithm may be more accurate but is more complex. KNN may be fast and suitable for high-dimensional data, but its accuracy can be affected by choice of the number of nearest neighbors (k). SVM may be a good option for linear boundary problems but may not perform well for more complex relationships. Neural networks may be good for complex non-linear relationships but may be computationally expensive. According to My results, SVM performs better with kernel basis function as compared to all classifiers.

For the heart disease and hotel reservation dataset, SVM and XGBoost performed well as compared to other classifiers. One of the key advantages of XGBoost is its ability to handle large and complex datasets, as well as its robust performance and accuracy. It also has a number of advanced features, such as built-in cross-validation, early stopping, and automatic handling of missing values, which make it easy to use and customize. XGBoost builds trees one by one, where each new tree tries to correct the mistakes of the previous tree. It uses a gradient descent algorithm to minimize the loss function, which measures the difference between the actual and predicted values.

IX. Clock Time

In terms of processing time, the XGBoost and SVM algorithms consumed the most time during the training process. The K-nearest neighbors algorithm took the longest time when making predictions. An interesting observation was that SVM takes more time when the dataset has more features, while Neural Networks take longer time in both cases due to the high number of parameters. On the other hand, decision trees took the least time as they can produce results in log(m) time at most.

X. Conclusion

On heart disease and hotel reservation data sets, I investigated various classifiers. I use the hyperparameters of several categorization models to fine-tune them. My observations show that while XGBoost took longer, it outperformed other classifiers. I evaluated all classifiers from different angles and provided in-depth analysis on experimented datasets.

References

[1] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J.-J. Schmid, S. Sandhu, K. H. Guppy, S. Lee, and V. Froelicher, “International application of a new probability algorithm for the diagnosis of coronary artery disease,” The American Journal of Cardiology, vol. 64, no. 5, pp. 304–310, 1989.