Martina De Luca

Homework Assignment 4

MSML605: Computing Systems for Machine Learning

2022-05-19

Objective

The objective is to perform data analysis and apply machine learning algorithms over a data set that contains health information to predict if a person is more likely to have heart disease in the future.

The first part of this work is dedicated to reviewing and understanding the available data, then performing the necessary adjustments and transformations to apply multiple machine learning algorithms for classification. Finally, comparing the results to identify the best algorithm for this use case.

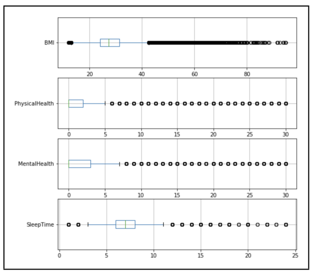
Data Description

The data set chosen is the 'Personal Key Indicators of Heart Disease' from Kaggle [1], which was extracted from a CDC survey from 2020 that contains information from 400,000 adults related to their health status. The author from this data set has already performed an initial data clean and removed irrelevant features to keep a smaller subset.
The specific data set used for this analysis contains 319795 rows and 18 features (including the label), which are listed below with relevant information about them [2].

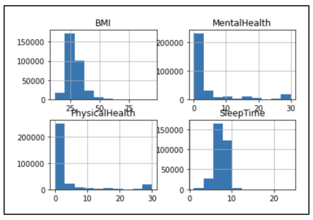
Feature	**Type**	**Values**
BMI	Numerical	Minimum: 12.02; Maximum: 94.85
Physical Health	Numerical	Minimum: 0; Maximum: 30
Mental Health	Numerical	Minimum: 0; Maximum: 30
Sleep Time	Numerical	Minimum: 1; Maximum: 24
Heart Disease	Categorical	Yes, No
Smoking	Categorical	Yes, No
Alcohol Drinking	Categorical	Yes, No
Stroke	Categorical	Yes, No
Diff Walking	Categorical	Yes, No
Sex	Categorical	Female, Male
Age Category	Categorical	50-54, 35-39, 60-64, 80 or older, 70-74, 40-44, 25-29, 45-49, 30-34, 55-59, 65-69, 18-24, 75-79
Race	Categorical	Black, Hispanic, American Indian/Alaskan Native, Other, White, Asian
Diabetic	Categorical	Yes; Yes (during pregnancy); No; No, borderline diabetes
Physical Activity	Categorical	Yes, No
Gen Health	Categorical	Poor, Very good, Excellent, Good, Fair
Asthma	Categorical	Yes, No
Kidney Disease	Categorical	Yes, No
Skin Cancer	Categorical	Yes, No

Data Exploration, Cleaning and Feature Engineering

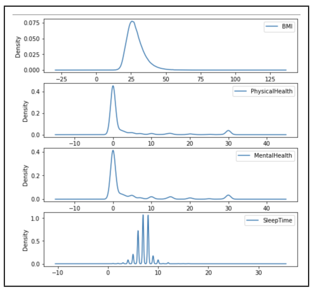
Data Exploration: For the numerical features, the following plots were generated to visualize their distributions. From these plots, we can observe that the data from the four numerical features is right-skewed and contains multiple outliers.



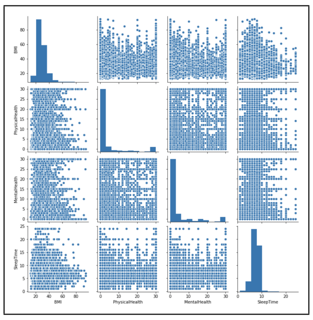
(a) Box Plot



(b) Histogram

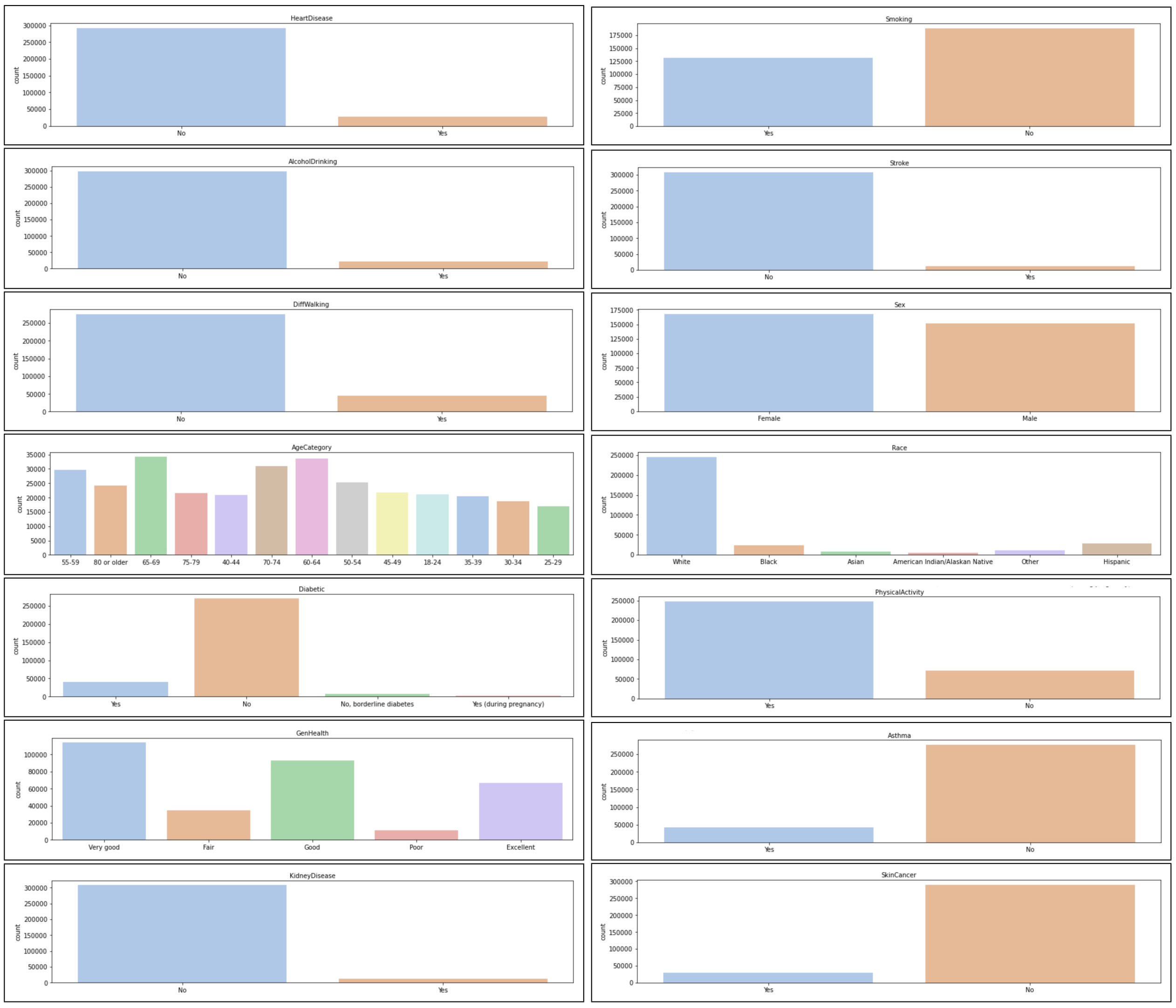


(c) Kernel Density Estimation Plot

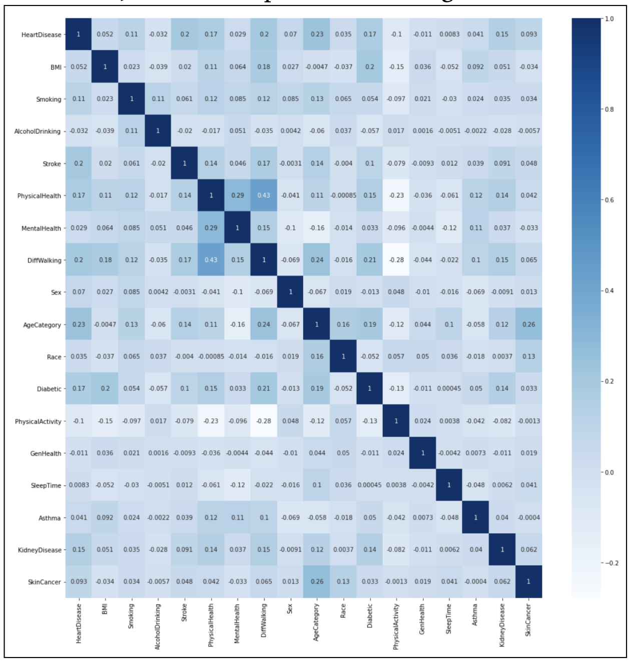


(d) Pair Plot

For the categorical features, the following plots were generated to visualize the data. From these plots, we can see that the labels are not evenly distributed between the two possible classes for Heart Disease: Yes, No. Therefore our labels are not balanced.

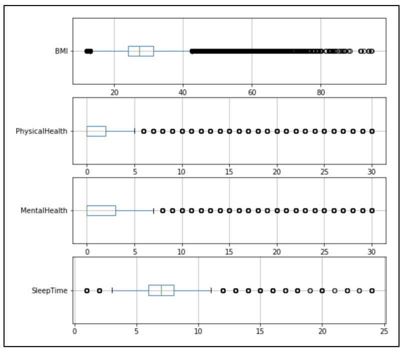


Then, to generate the Pearson Correlation Matrix, the categorical features need to be transformed into numerical, which was performed using the Label Encoder (this is also needed to use the algorithms later).

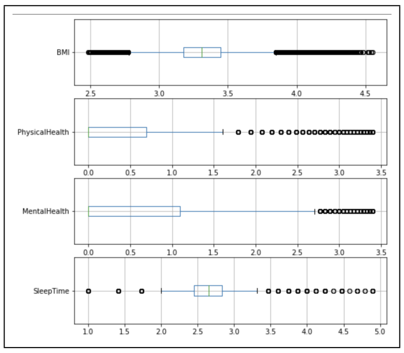


DATA CLEANING: Since there are no missing values, there is no need to perform imputation. For improving the skewed data for the numerical features, box-cox method [3] was applied to BMI and Sleep Time features, and logarithmic function to Physical and Mental Health features. The below tables and plots show how the skewness was reduced; but both the Physical and Mental Health features are still very skewed, this is because the majority of the values for these features are zero (which means that the person didn't have physical or mental health issues in the last 30 days) which is a valid input.

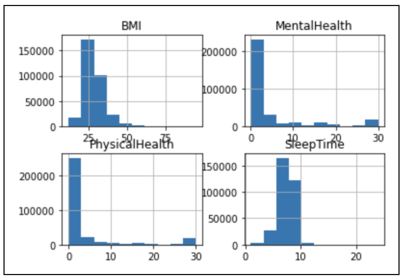
Feature	Initial Skew	Final Skew
BMI	1.33	-0.01
Physical Health	2.60	1.78
Mental Health	2.33	1.38
Sleep Time	0.68	0.30



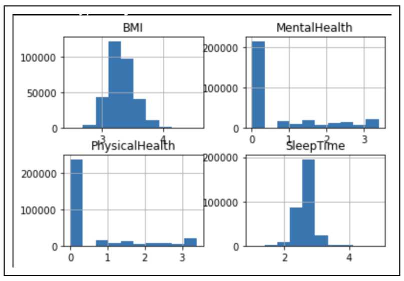
(a) Before



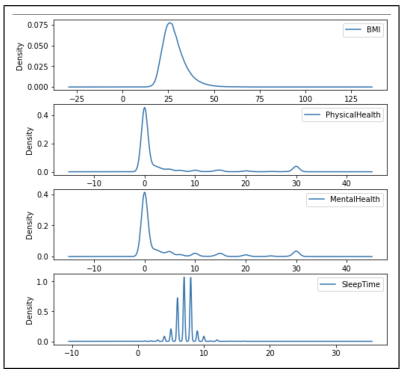
(b) After



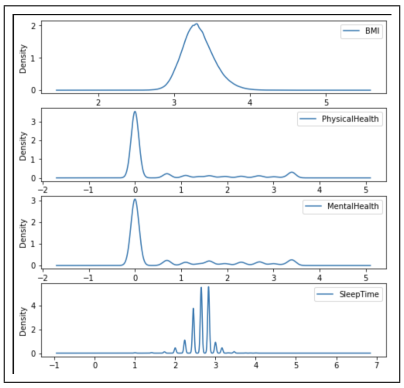
(c) Before



(d) After



(e) Before



(f) After

After data was clean and the skewness reduced, the data set was split between features and label. By checking the number of instances in each label, it is noticeable that the data is not balanced. To solve this issue, SMOTE [4] was applied for oversampling the minority class (Note: SMOTEENN and SMOTETomek methods were attempted, but execution would not conclude). After the data was balanced, the data was split between train and test sets with an 80-20 split.

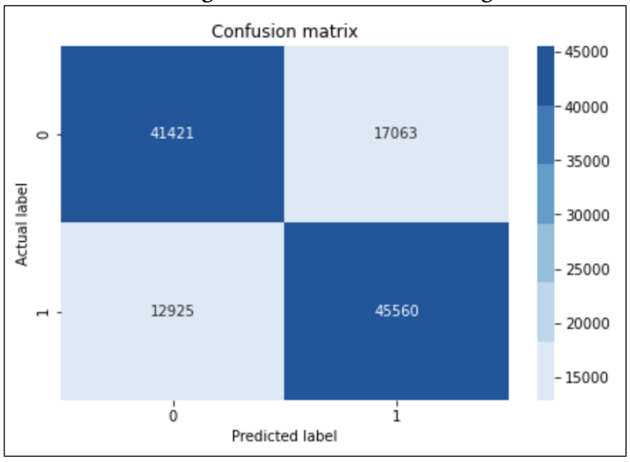
Before applying the algorithms, RobustScaler was used to normalize and scale the data and handle outliers. Note that the scaler is fit only on training data and the same transformation is used for both train and test sets.

FEATURE ENGINEERING: From the previous observations on the values of Physical and Mental Health, the algorithms were tested removing these features from the data set. This change didn't significantly improve the results, it slightly reduced the values of the metrics, so the features were kept in the data set. In addition, the algorithms were also tested removing features with low correlation to the Heart Disease label (based on the Pearson Correlation Matrix), such as Sex and Race, but again, this change didn't improve significantly the results, therefore all features were kept.

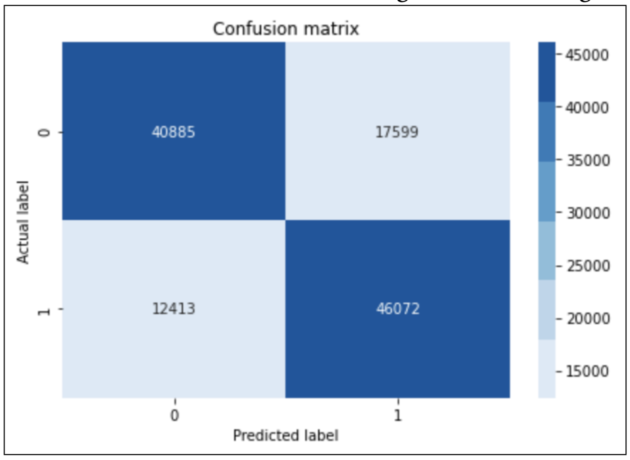
ML Algorithms

For this binary classification problem, five algorithms were chosen: Logistic Regression, Support Vector Machines, Decision Trees, Boosting and Stacking. For all of them, the following steps were followed: create the model with the chosen parameters, fit the model with the training data, predict the labels for the test data, compare the predicted values to the actual ones, calculate several metrics and plot the confusion matrix to show the relation between true positives, true negatives, false positives and false negatives in each scenario.

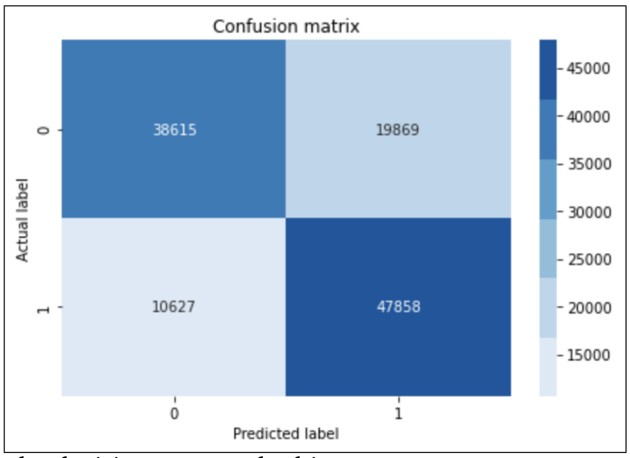
Logistic Regression: The first algorithm chosen was Logistic Regression, this model was implemented with the default settings. From this model we get the following confusion matrix:



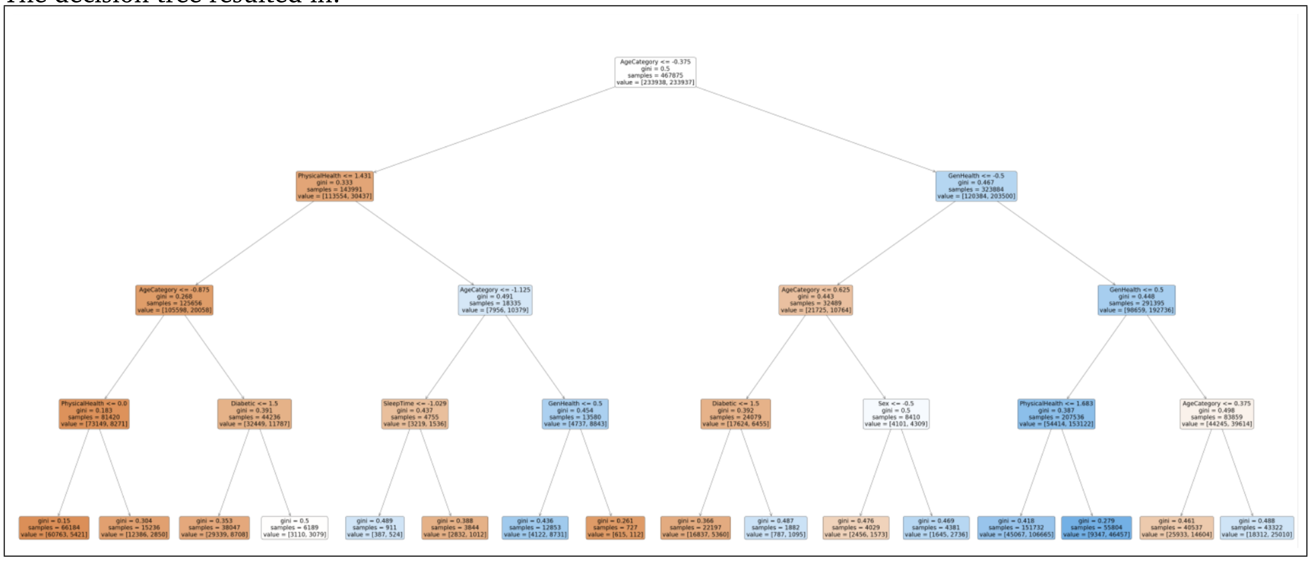
Support Vector Machine: The second algorithm tested was support vector machines, specifically using the function for linear support vector classification. The only parameter specified different than the default was for the function to use 'dual=False', since the number of samples is significantly larger than the number of features [5]. From this model we get the following confusion matrix:



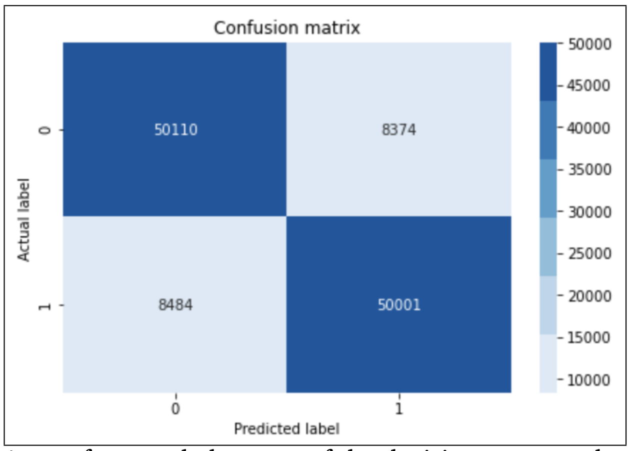
Decision Trees: The third algorithm was the decision tree with a maximum depth of 4, which was the best parameter option in order to achieve good accuracy, but still remaining a legible decision tree diagram. From this model we get the following confusion matrix:



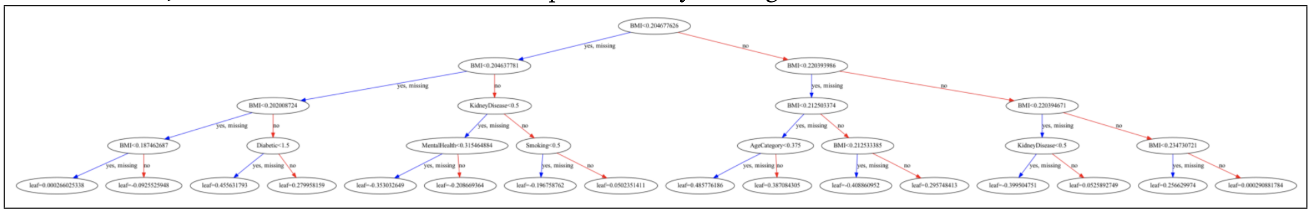
The decision tree resulted in:



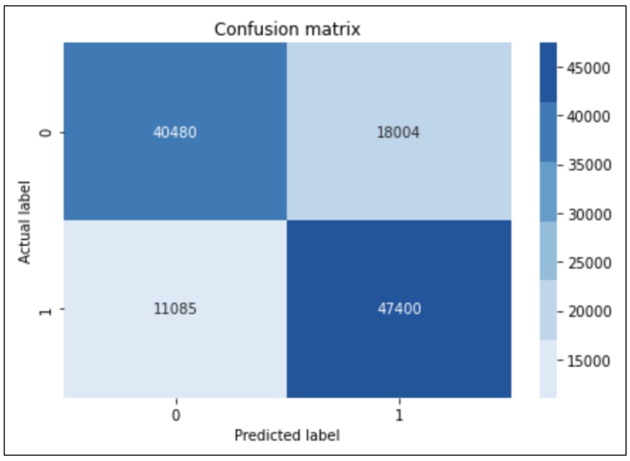
Boosting: In addition to the three previous algorithms I decided to also test the boosting method [6] [7] as an ensemble method of decision trees. From this model we get the following confusion matrix:



As a reference, below one of the decision trees produced by the algorithm:



Stacking: The final model implemented is a combination of the three initial models, using the stacking method [8]. From this model we get the following confusion matrix:



Comparison

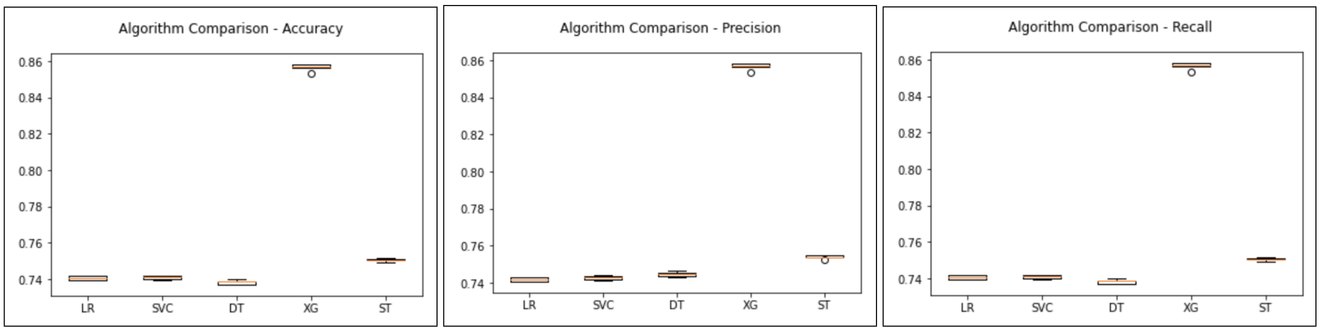
Firstly, we can compare the confusion matrices from each of the models above, whic are all quite similar with high values for true positives and true negatives, and lower values for false positives and false negatives.

Secondly, we can compare in the table below the metrics for each of the algorithms when run separately using train and test sets only. Here we can see that the results from the Boosting method are better than the other algorithms.

	Logistic Regression	SVM	Decision Trees	Boosting	Stacking
Accuracy	0.74	0.74	0.74	0.86	0.75
Precision	0.73	0.72	0.71	0.86	0.72
Recall	0.78	0.79	0.82	0.85	0.81
F1 Score	0.75	0.75	0.76	0.86	0.77

In addition, I performed cross validation with each of the algorithms, using stratified kfolds. Below we can see again that Boosting provides better results than the other algorithms [9] [10] [11].

	Logistic Regression	SVM	Decision Trees	Boosting	Stacking
Accuracy	0.74	0.74	0.74	0.86	0.75
Precision	0.74	0.74	0.74	0.86	0.75
Recall	0.74	0.74	0.74	0.86	0.75



Boosting would be the recommended approach, because not only does it deliver the best results in the metrics, the decision trees that the algorithm provides are easier to visualize and explain. In addition, this model presents the highest recall and the lowest false-negative rate, which are the main metrics we want to reduce for this scenario, since we want to reduce the number of times we miss diagnosing heart disease.

Summary Key Findings and Insights

As a summary, from the analysis we can observe that this data set contains many categorical features that we had to encode to use in our models (most of them were Yes/No questions which easily translate to 1 and 0). In addition, for the numerical features, we observed that Physical and Mental Health contained a large amount of zero values; as a recommendation, there could be another approach to gather this information in order to avoid this, perhaps formulating the question differently to get other type of value answers. Another important finding to note is that this data is highly imbalanced, so that required treatment during the analysis to yield better results.

Reviewing the performance of the algorithms tested, we can indicate that the performance metrics are high and similar. The algorithm that stands out, having the best score in all the metrics, is Boosting, which has been observed as a great model to easily and quickly get good results. This model also provides the advantage of using decision trees, which provide an easy visualization to understand the decisions made by the algorithm. This model executes so well because it is an ensemble method, where it improves itself by corrections based on previous results.

It is important to note that due to the nature of this use case, one of the key metrics we want to maximize is the recall, because we want to reduce the false negatives. Reviewing the metrics, we can see that most of the algorithms perform very well for recall, including Boosting.

Finally, as future work, on top of the possible improvements already mentioned, the predictions could improve by including additional features, performing more feature engineering and more hyper-parameter tuning to the models.

References

[1] Kamil Pytlak from Kaggle. Personal Key Indicators of Heart Disease. URL `https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease`.

[2] Khanejo from Github. IBM-Exploratory-Data-Analysis-for-Machine-Learning. URL `https://github.com/Khanejo/IBM-Exploratory-Data-Analysis-for-Machine-Learning/blob/main/final.ipynb`.

[3] Erik Marsja. How to use Square Root, log, & Box-Cox Transformation in Python. URL `https://www.marsja.se/transform-skewed-data-using-square-root-log-box-cox-methods-in-python/`.

[4] Jason Brownlee. SMOTE for Imbalanced Classification with Python, . URL `https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/`.

[5] Scikit-Learn Documentation: Linear SVC. URL `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html\#sklearn.svm.LinearSVC`.

[6] Jason Brownlee. Extreme Gradient Boosting (XGBoost) Ensemble in Python, . URL `https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/`.

[7] Jason Brownlee. How to Visualize Gradient Boosting Decision Trees With XGBoost in Python, . URL `https://machinelearningmastery.com/visualize-gradient-boosting-decision-trees-xgboost-python/`.

[8] Tavish. Basics of Ensemble Learning Explained in Simple English. URL `https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/`.

[9] Jason Brownlee. How To Compare Machine Learning Algorithms in Python with scikit-learn, . URL `https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/`.

[10] Dibyendu Deb. Comparing the performance of different machine learning algorithms. URL `https://dibyendudeb.com/comparing-machine-learning-algorithms/`.

[11] Evaluate multiple scores on sklearn cross val score. URL `https://microeducate.tech/evaluate-multiple-scores-on-sklearn-cross_val_score/`.