Project Description

Heart Disease is one of the leading causes of death in America. Thus, it is important to find methods that can be used to detect heart disease early in order to save lives and allow hospitals to manage their resources properly so they can focus on those groups who are at high risk of disease. This project aims to use machine learning methods to predict whether a patient will have high risk of heart disease or not. There are eleven independent variables in the dataset: age, sex, ChestPainType, RestingBP, Cholesterol, FastingBS, RestingECG, MaxHR, ExerciseAngina, Oldpeak and ST_Slope. The independent variables are a mix of numeric and categorical variables. The dependent variable is a binary numeric variable: 0 for not being high risk of heart disease, 1 for being high risk.

Part 1: Preprocessing, Data Mining and Visualization

The first step in this project was visualizing and exploring the dataset. First, I checked what percentage of people were deemed high risk for heart disease to see how balanced the dataset was. I found about 55% were positive while 45% were negative which indicates the data isn't super unbalanced which is important to know because it would affect how I interpret the effectiveness of my model results later. Next, I created histograms/bar charts for each independent variable in the dataset, with the independent variable as the x-axis and count as the y-axis for both cases (0/1 for heart disease). This allowed me to see general trends in the data. Some observations (among many) I noticed were heart disease became more common the older the age, the majority of heart disease victims were men, chest pain type ASY is much more likely to lead to heart disease and the majority of ST_Slope flat and ST_Slope Down were likely to have heart disease.

Furthermore, I created boxplots for the numeric independent variables to detect outliers. I noticed problematic outliers with regards to RestingBP and Cholesterol having some values equal to zero, which makes absolutely no sense for living human beings. Instead of removing all those data points (since there were a lot), I decided to fill values with 0 with the median value of their column (a more sensible value that would be less sensitive to outliers). Afterwards, I did label encoding so the categorical data could be passed in properly into the machine learning models later. Then, I created a correlation matrix for all the variables. From the matrix, I learned that none of the variables have super strong positive or negative correlations as the highest positive correlation was 0.49 (ExerciseAngina and HeartDisease) and highest negative correlation was -0.56 (ST_Slope and HeartDisease). I decided as input features to keep the features in the correlation matrix with correlation higher than the absolute value of 0.1, which meant keeping all features except Cholesterol and RestingECG (0.043 and 0.057 respectively). Besides filling in zero-values and label encoding, for data preprocessing, I also performed PCA (using 8 components because 8 significant signals exist in the independent variables, as 8 components explained greater than 90% of the variance) and used standard scaling (since data in different units with different ranges) to reduce the dimensionality of the data. For derived/alternative features, I decided to build my classifiers on the reduced-dimension pca projected data.

Part 2: Classification

Machine Learning Method #1: Support Vector Machine

The Support Vector Machine classification algorithm works by mapping data points to a high-dimensional space and then tries to find the optimal hyperplane that divides the data into two classes. Its advantages include a robustness to noise and ability to handle large data sets. I applied this classifier on the dimension-reduced pca projected data. For my classifier, I kept it simple and did not change most of the parameters from the default, except I made the probability parameter equal to true in order to make my ROC later.

Machine Learning Method #2: Random Forest Classifier

The Random Forest Classifier uses random forests, which are meta estimators that fit a number of decision tree classifiers on subsets of datasets and uses averaging to improve predictive accuracy by reducing over-fitting. I applied this classifier on the pca projected data. In terms of parameters, I implemented a max depth of 5 and a random_state of 0 and kept everything else default.

Machine Learning Method #3: K Neighbors Classifier

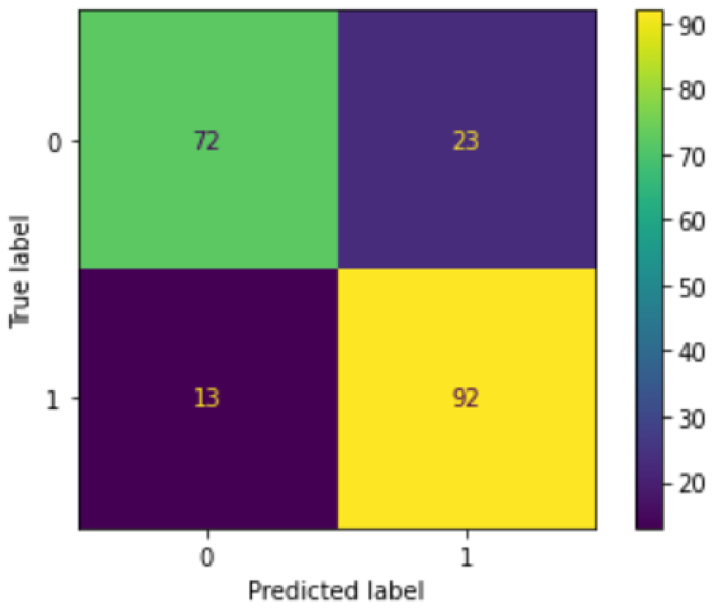
The way this classifier works is that it assumes that similar things exist in close proximity. It initializes a k number of neighbors and computes the distance between a data point and its k closest neighbors, noting the labels of its closest neighbors and choosing the mode of the k labels as its final classification. I applied this classifier on the pca projected data. For parameters, I just chose a number of neighbors to be 15 and kept everything else default.

Part 3: Evaluation

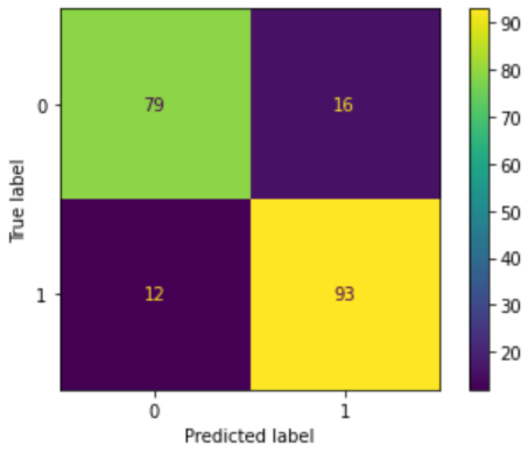
The following table shows the evaluation statistics for each classification method. We can see that the lowest F1 belongs to SVM while highest belongs to random forest. The lowest bias belongs to Random Forest while highest is SVM and lowest variance belongs to Random Forest while highest is K Neighbors.

	F1	**Bias**	**Variance**
SVM	**0.836**	**0.180**	**0.046**
Random Forest	**0.869**	**0.140**	**0.040**
K Neighbors	**0.860**	**0.155**	**0.053**

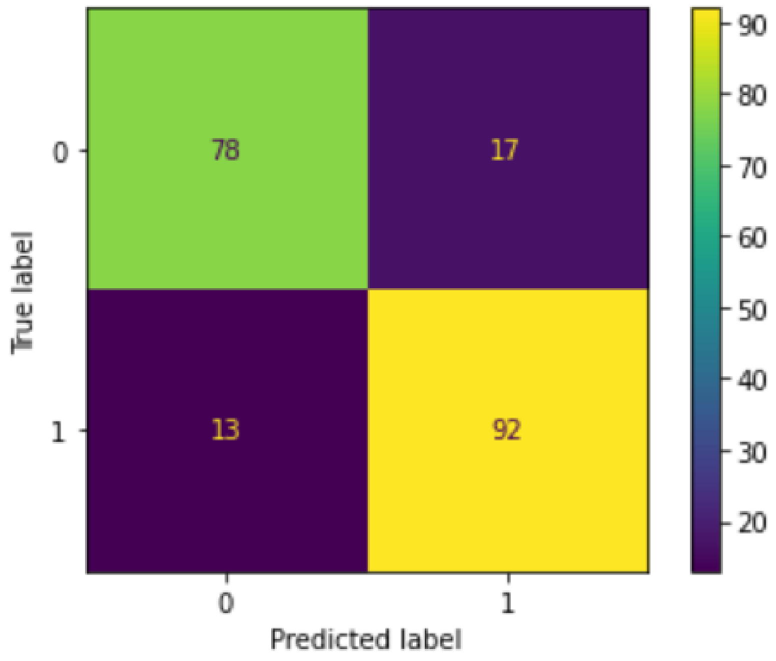
Confusion Matrix SVM



Confusion Matrix Random Forest

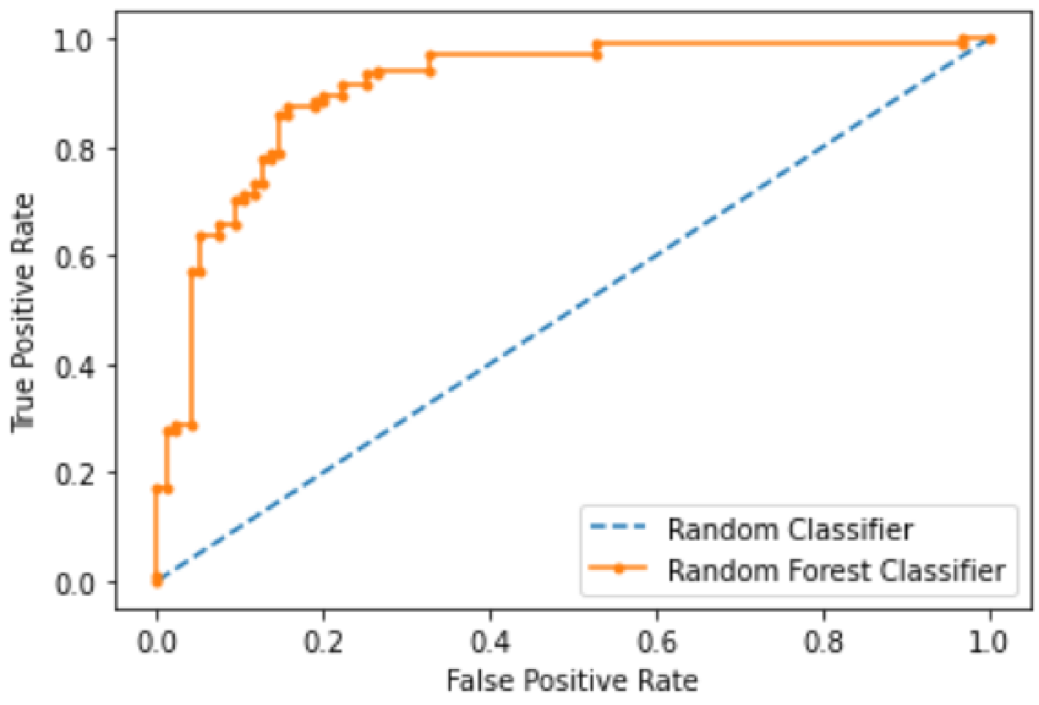


Confusion Matrix K Neighbors



The above confusion matrices simply give us information about the number of true positives, true negatives, false positives and false negatives there are after doing classification.

ROC Random Forest



The ROC curve above tells us how well the model is capable of distinguishing between classes.

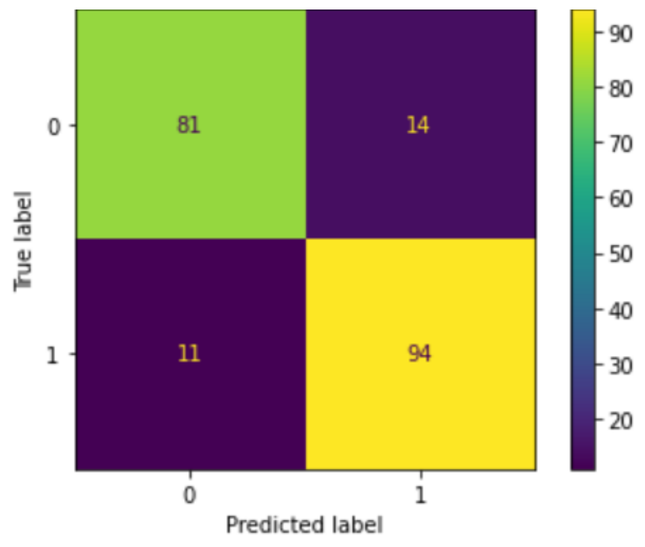
Overall, the classifier that did the best in my opinion is Random Forest and I based that on it having the highest f1 score which I think is an appropriate statistic to base it on because it represents the harmonic mean of precision and recall. In my data, the variance is pretty low while the bias is pretty high so the next step would be to increase the model power by doing things such as increasing the parameters like increasing the number of trees, increasing depth of decision tree or specifying a max number of features to be included at each node split among other things. A good operating point on the ROC above would be up and to the left, where it’s a high true positive rate and a low false positive rate, such as the point where it’s about 15% false positive and 90% true positive.

Part 4: Iteration

I choose to modify my Random Forest Classifier. I was trying to optimize the f1 score.

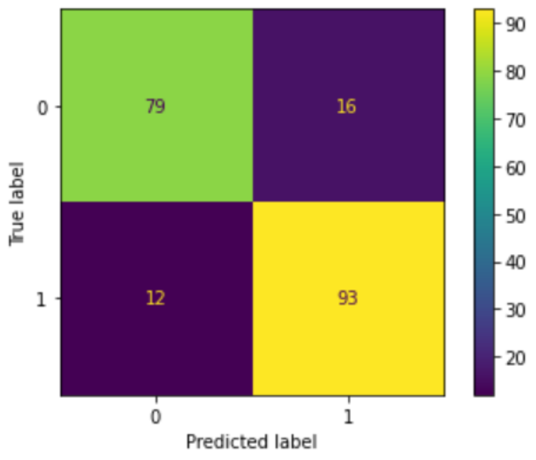
Iteration #1: Changed the number of estimators from 100 to 200 and the max depth from 5 to 10. As a result, my f1 score went from 0.869 to 0.883.

Confusion Matrix



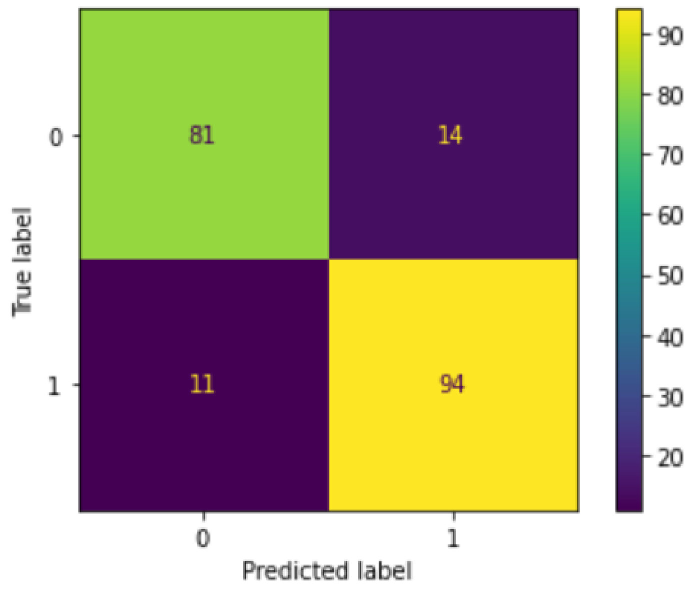
Iteration #2: Changed criterion from default gini to entropy and increased the max depth to 15. However, compared to my first iteration, my f1 score went down from 0.883 to 0.869.

Confusion Matrix



Iteration #3: Changed the number of estimators to 500, max depth to 25, and max features to log2. My f1 score did not improve from my peak f1 score in the first iteration (remained the same at 0.883).

Confusion Matrix

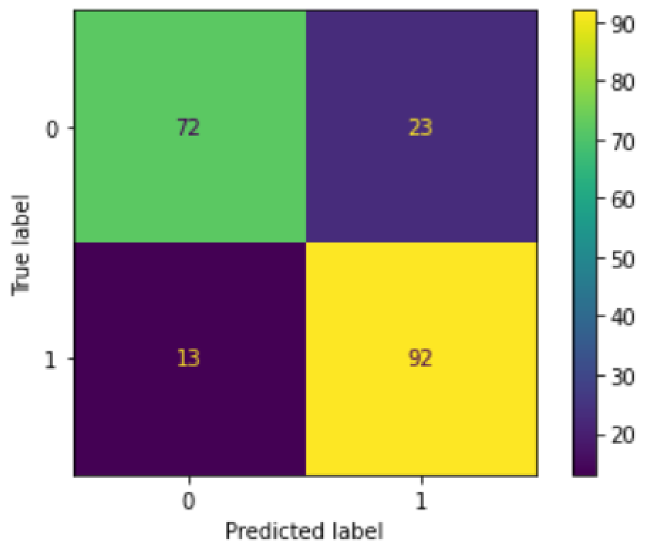


Extensions:

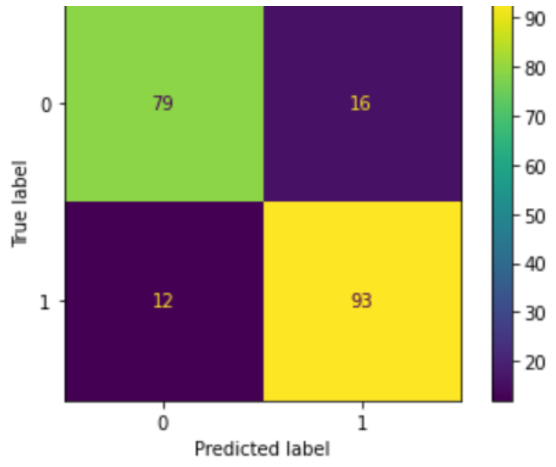
Evaluation Statistics for Additional Machine Learning Methods:

	F1	Bias	Variance
Logistic Regression	0.836	0.180	0.026
Gradient Boosting Classifier	0.869	0.130	0.084

Logistic Regression Confusion Matrix



Gradient Boosting Classifier Confusion Matrix



From looking at the evaluation statistics above and the confusion matrices, we can see that neither of these additional classifiers brought an improvement in performance. Compared to Random Forest, the f1 score decreased and bias increased for logistic regression while for gradient boosting classifier, f1 remained the same, bias slightly decreased but variance increased.

Table Showing Area Under the Curve for Multiple Classifier Results:

Classifier	AUC
SVM	0.903

Random Forest	0.910
K Neighbors	0.890
Logistic Regression	0.911
Gradient Boosting Classifier	0.883

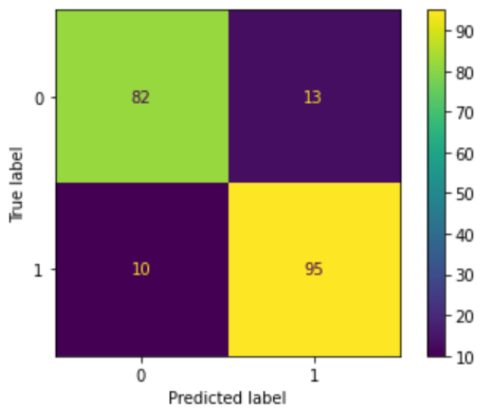
From generating additional ROC curves, we can see that for all the classifiers the areas under the curve are pretty close to each other with the difference between the smallest and highest being about 0.03 which suggests there is not that much difference between the classifiers in their ability to distinguish between classes. Logistic regression seems to have the best ability to distinguish, with AUC at 0.911 while gradient boosting classifier has the worst at 0.883.

Attempt additional iterations to make your system better: I used GridSearchCV as a method of hyperparameter tuning for the Random Forest Classifier. I used this grid parameter:

grid_param = { "n_estimators": [100, 200, 500], "max_depth": [5, 10, 25], "max_features": ["sqrt", "log2"], "min_samples_split": range(2, 10, 1), "min_samples_leaf": range(1, 5, 1) }. This grid parameter led to about 2880 iterations being performed to find the best parameters, which were found to be {'max_depth': 25, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}. After implementing these parameters into the random forest classifier, my f1 score (the statistic that I was trying to optimize) increased from 0.883 to 0.892. Accuracy also increased from 0.875 to 0.885.

Metrics after hyperparameter tuning:

```
Accuracy: 0.885
```



```
f1_score: 0.892018779342723
Bias: 0.135
Variance: 0.047
ROC AUC=0.907
```

Reflection:
Doing this project taught me how to apply all the methods I learned in this class to an unknown real-life dataset. I learned about the importance of visualizing and exploring datasets (to see general trends/possible outliers) before moving into preprocessing and model building. Additionally, I learned how to properly preprocess data so it can enter into machine learning algorithms in a readable format and I also learned how to interpret ROC curves and how to tune parameters to try to optimize performance. Overall, the most interesting part of this project was being given the dataset and not much else guidance and just making decisions on my own and seeing where it took me.

Acknowledgements:
Towardsdatascience.com, sci-kit learn documentation, machinelearningmastery.com, geeks4geeks, kaggle