Supervised Learning Algorithm Exploration

Thao Duong
tduong44@gatech.edu

1 INTRODUCTION

Five supervised learning algorithms: decision trees, neural networks, boosting, Support Vector Machines, and knn, were explored with two separate datasets: heart disease and red wine quality. With the heart disease dataset, the goal is to correlate different 13 attributes such as age, sex, chest pain type, blood pressure, cholestorol to whether the patient has heart disease. With the red wine quality dataset, the goal is to correlate 11 physicochemical attributes such as acidity, sugar level, alcohol level, and pH to a sensory quality rating from 1 to 10.

In the process of finding the best model for each dataset, the following areas were explored:

- Analysis of 5 different supervised learning algorithms
- Hyperparameter tuning, algorithms specific
- Feature selection on the dataset
- Learning curve
- Training time and query time measurement
- Accuracy of prediction
    - Training score vs. cross validation score
    - Accuracy as a function of training examples

2 DATASET CHOICE AND THE CLASSIFICATION PROBLEMS

Datasets were taken off of Kaggle. There were 3 preferences/biases when datasets were chosen.

- Datasets smaller than 1MB were preferred over larger datasets. This is done to reduce runtime.
- Datasets with numerical data rather than text were preferred, to eliminate the extra step of mapping in algorithms such as kNN. Even with numerical representation, there were still interesting results that will be discussed further in each algorithm section.
- Datasets with a target/goal column to train for in supervised learning.

2.1 Hearts Disease

2.1.2 *Dataset composition*

This dataset contains 13 features/attributes such as age, sex, chest pain type, blood pressure, cholestorol, etc, that may correlate to the presence of heart disease in a patient. It has 1 column representing the goal or target, with values 0 representing no heart disease, and 1 means presence of heart disease. There are 303 data points in this set. All data are numerical values, integers or decimals. There is no missing data.

2.1.2 *Why this is an interesting dataset*

The sample size is small, combined with a relatively large number of dimensions, causing underfitting issues. If all 13 features were used to create the model, there was not enough data to produce consistent results. Randomly selecting different training sets, cross validation sets, and test sets can produce very different results. This is an example of the curse of dimensionality as discussed in lecture. For certain algorithms like decision tree, the feature with the highest correlation is automatically selected first, so reducing the number of dimensions does not help. However, for algorithms like kNN, low correlation features would add extra noise, removing low correlation features would result in better accuracy.

2.2 Red Wine Quality

2.2.2 *Dataset composition*

This dataset contains 11 different physicochemical attributes such as acidity, sugar level, pH, chlorides(salt) content, sulfate dioxide(gas) content, and alcohol level, etc, that may correlate with the quality rating of the wine. It has 1 column representing the rating of the wine, with values from 1 to 10. A rating of 1 is a bad wine. A rating of 10 is the best wine. There are 1599 data points in this set. All data are numerical values, integers or decimals. There is no missing data.

2.2.2 *Why this is an interesting dataset*

Unlike the heart disease dataset, the red wine quality dataset has sufficient data to build a model without reducing the dimensions. While the heart disease dataset is clearly a classification problem because the target column has values 1 or 0, the red wine quality dataset can be viewed as a classification or regression problem, with the target column having values from 1 to 10. The target column

data can also be arbitrarily divided (e.g. rating > 5 is good wine, rating <= 5 is bad wine) to simplify the classification problem even further. This results in better prediction because we only need to predict whether the wine is good or bad versus predicting a rating from 1 to 10.

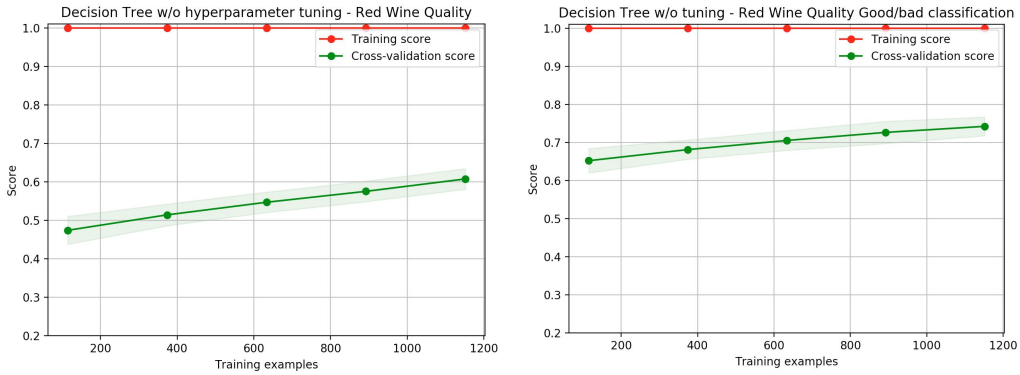


Figure 1—Red Wine Rating predicting ratings 1 to 10 vs predicting good or bad.

Adjusting the prediction goal (from 1-10 rating to good/bad rating), a 12% increased in accuracy was observed. This was done without other hyperparameter tuning (Figure 1).

3 ALGORITHM ANALYSIS

3.1 Decision Trees

Scikit-learn uses a variation of ID3 to generate the classifiers. More specifically, *DecisionTreeClassifier* class was used. Figure 2 compares the models generated with default parameters, and with tuned hyperparameters. The default parameters are: *criterion: gini, splitter: best, max_depth: None, min_samples_split: 2, ccp_alpha: 0.0*. Before tuning, the training score is at 1 while cross-validation scores between .67 to .73. This is an indication of overfitting.

3.1.1 *Hyperparameter tuning and pruning of decision nodes*

Gridsearch with the following values were used to decide the best parameter

- Criterion : {gini, entropy}. This parameter is used to determine which feature is used to split the tree.
- Splitter: {best, random}

- Max_depth: {int array from 2 to 10}. This parameter is used to determine the maximum height of the tree. Lower number means shorter tree. A lower number can help prevent overfitting, but too low can be underfitting.

- Min_samples_split: {2, 3, 4, 5}. This parameter is used to help prevent overfitting. Lower number potentially means more splitting, and overfitting. Higher number can potentially mean not enough splitting and underfitting

- Ccp_alpha: {.005 - .045 by .005 increments}. This parameter is used to prune the tree. Higher ccp_alpha values increased the number of nodes pruned.

Gridsearch was run multiple times, and the resulting parameters were not always consistent. The most common parameters picked were *{'ccp_alpha': 0.01, 'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 5, 'splitter': 'random'}* for the heart disease dataset. The hypertuned parameters produced about a 5% increase in accuracy.

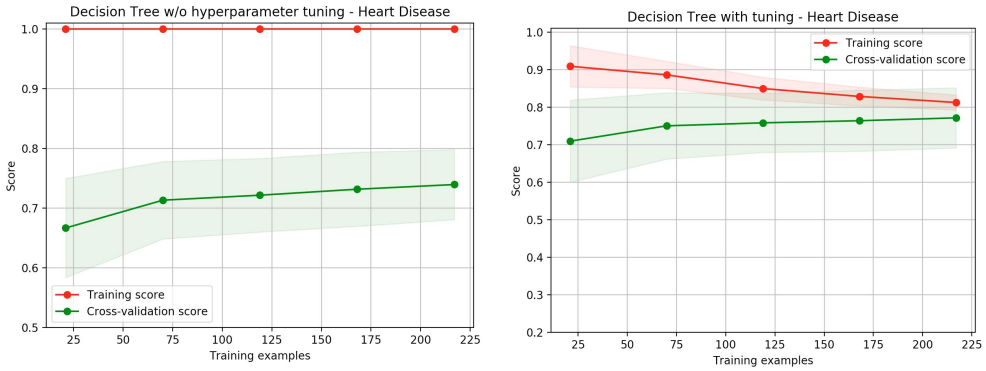


Figure **2**—Heart Disease prediction with and without hyperparameter tuning.

The red wine quality dataset has the same issue of overfitting before hyperparameter tuning (Figure 3.) The same grid search parameters were ran and the result was *{'ccp_alpha': 0.005, 'criterion': 'entropy', 'max_depth': 8, 'min_samples_split': 2, 'splitter': 'random'}*. However, hyperparameter tuning only decreased the accuracy of the training score, but did not improve the accuracy of the test score. To further improve the model, the range of the parameters can be increased or more parameters can be added to the grid. As the number of

parameters and range increase, the resource requirements will grow exponentially, and the probability of finding a more accurate model will decrease. The training score and cross-validation score is close together is an indication that this is close to an optimal model.

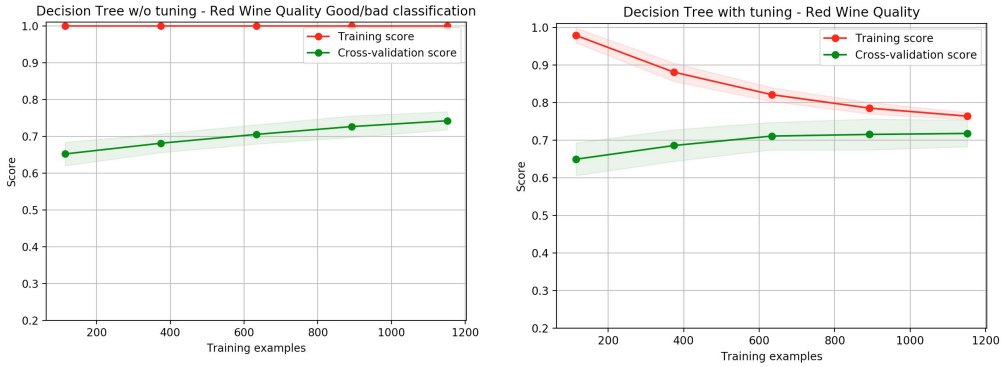


Figure 3 — Red wine quality prediction with and without hyperparameter tuning using Decision Tree.

3.1.2 *Learning Curve*

Without hyperparameter tuning score for both datasets were 1, meaning that it performs with 100% accuracy on the training set. However, the cross-validation score is 25%-35% less accurate. This suggests overfitting of the model. After hyperparameter tuning, accuracy decreased in the training set, but increased in the cross validation set in the heart disease model. This suggests an improvement of the model.

3.2 Neural Network

Scikit-learn uses a Multi-layer Perceptron(MPL) classifier with the following default parameters: *activation: relu, solver: adam, hidden_layer_sizes: 100.* See Scikit-learn docs for full details of other parameters and their default values.

Gridsearch with the following values were used to decide the best parameter for both datasets: heart disease and red wine ratings

3.2.1 *Hyperparameter tuning*

- Activation : { identity, logistic, tanh, relu }. This is the activation function for the hidden layer.
- Solver : { lbfgs, sgd, adam }

- Hidden_layer_sizes':[(1,),(2,),(3,),(4,),(5,),(6,),(7,),(8,),(9,),(10,),(11,),(12,),(13,),(14,),(15,),(16,),(17,),(18,),(19,),(20,),(21,)]
- max_iter: [50,100, 150, 200, 250, 300]

Figure 4 shows the models' performance for the heart disease dataset and the red wine ratings dataset after hyperparameter tuning. For heart disease, the parameters that produced the highest accuracy are: *{'activation': 'identity', 'hidden_layer_sizes': (18,), 'solver': 'lbfgs', 'max_iter': 200}*. For red wine quality, the resulting parameters are: *{'activation': 'relu', 'hidden_layer_sizes': (10,), 'max_iter': 250, 'solver': 'lbfgs'}*

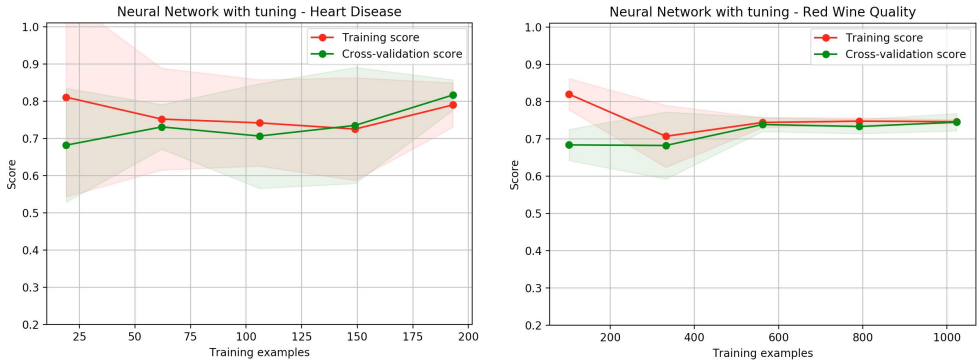


Figure 4 — Heart disease and red wine quality prediction after hyperparameter tuning using Neural Network.

3.2.2 *Accuracy before and after tuning*

Accuracy score before hyperparameter tuning was not charted to save space. However accuracy of the test set was calculated. There is an increase in accuracy of 11% for heart disease and 4% increase for red wine quality.

Table 1—Accuracy of test set before and after hyperparameter tuning for heart disease and red wine quality datasets using MPL

	Heart Disease	**Red Wine Quality**
Before hyperparameter tuning	*0.77*	0.73
After hyperparameter tuning	*0.88*	0.77

3.3 Boosting

Boosting combines an ensemble of weak estimators to make a stronger classifier. Scikit-learn uses Adaboost classifier, which is an implementation of AdaBoost-SAMME algorithm. Adaboost has the following default parameters: *base_estimator: DecisionTreeClassifier with max depth of 1, n_estimators: 50.* See Scikit-learn docs for full details of other parameters and their default values.

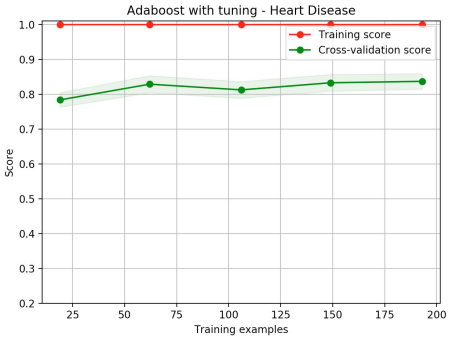
3.3.1 *Hyperparameter tuning*

Since parameters for decision trees were already tuned on the two datasets, were set when the decision tree estimator was used to further increase the accuracy of the boosting model. For heart disease, they were: *{'ccp_alpha': 0.01, 'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 5, 'splitter': 'random'}*. For red wine quality, they were: *{'ccp_alpha': 0.005, 'criterion': 'entropy', 'max_depth': 8, 'min_samples_split': 2, 'splitter': 'random'}.*

Gridsearch with the following values were used to decide the best parameter for both datasets: heart disease and red wine ratings

- Base_estimator: DecisionTreeClassifier, SVC(probability=True, kernel='linear')
- N_estimators: [50, 100, 150, 200, 250]

Figure 5 shows the models produced by Adaboost performance for the heart disease dataset and the red wine ratings dataset after hyperparameter tuning. For both datasets, the best parameters were *{Base_estimator: DecisionTreeClassifier, N_estimators: 250}.* It looks like there is still significant overfitting even after hyperparameter tuning, but the accuracy for both cross validation and test set is higher than decision tree, neural network alone. Explanations for overfitting can be attributed to the 250 complex trees. But it would take too long to grid search for both tree params and Adaboost params at the same time.



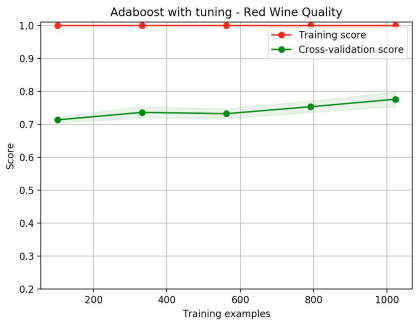


Figure 5 — Heart disease and red wine quality prediction after hyperparameter tuning using Adaboost.

3.3.2 *Accuracy before and after tuning*

Table 2 shows the accuracy of the test set before and after hyperparameter tuning. There is an increase in accuracy of 5% for heart disease and 2% increase for red wine quality.

Table 2—Accuracy of test set before and after hyperparameter tuning using Adaboost

	Heart Disease	**Red Wine Quality**
Before hyperparameter tuning	*0.84*	0.78
After hyperparameter tuning	*0.89*	0.80

3.4 Support Vector Machines

Support Vector Machines (SVM) is effective in high dimensional spaces, which is a better classification of our data, since the number of samples are relatively small. Scikit-learn uses C-Support Vector Classification (SVC) with the following default parameters: *C: 1.0, kernel: rbf, gamma:scale.* See Scikit-learn docs for full details of other parameters and their default values.

3.4.1 *Hyperparameter tuning*

Gridsearch with the following values were used to decide the best parameter for both datasets: heart disease and red wine ratings

- Kernel: rbf, gamma: [1e-3, 1e-4] C: [1, 10, 100, 1000]
- Kernel: linear, C: [1, 10, 100, 1000]

Figure 6 shows the models produced by SVC performance for the heart disease dataset and the red wine ratings dataset after hyperparameter tuning. For heart disease, the parameters that produced the highest accuracy are: {'C': 1, 'kernel': 'linear'}. For red wine quality, the resulting parameters are: {{'C': 1000, 'kernel': 'linear'}. Grid search took a long time and crashed, thus we adjusted C =10 to generate the graph.

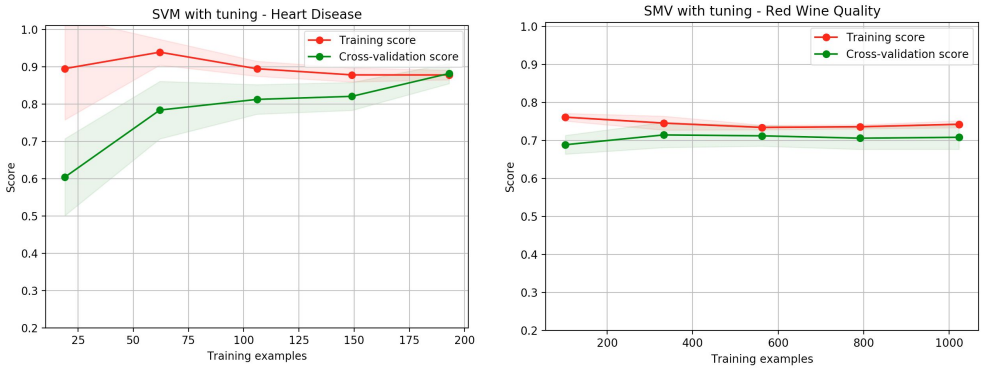


Figure 6 — Heart disease and red wine quality prediction after hyperparameter tuning using SVC.

3.4.2 *Accuracy before and after tuning*

Table 3 shows the accuracy of the test set before and after hyperparameter tuning. There is an increase in accuracy of 4% for heart disease and 16% increase for red wine quality. Even though the accuracy score for both the training and cross validation were high, the score for the test set was significantly lower. This suggests that more detailed cross validation is needed to train the model.

Table 3—Accuracy of test set before and after tuning for SVC

	Heart Disease	**Red Wine Quality**
Before hyperparameter tuning	*0.73*	0.62
After hyperparameter tuning	*0.77*	0.78

3.5 k-Nearest Neighbors (kNN)

kNN looks for the nearest k data points, and then tries to classify the query datapoint. Scikit-learn uses k-nearest neighbors vote with the following default parameters: *n_neighbors: 5, weights: ['uniform', 'distance'], algorithm: ['auto', 'ball_tree', 'kd_tree', 'brute'].* See Scikit-learn docs for full details of other parameters and their default values.

3.5.1 *Hyperparameter tuning*

Gridsearch with the following values were used to decide the best parameter for both datasets: heart disease and red wine ratings

- N_neighbors : [3, 4, 5,..., 20]

- Weights': ['uniform', 'distance'],
- Algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']

Figure 7 shows the models produced by kNN performance for the heart disease dataset and the red wine ratings dataset after hyperparameter tuning. For heart disease, the parameters that produced the highest accuracy are: *{'algorithm': 'auto', 'n_neighbors': 10, 'weights': 'uniform'}*. For red wine quality, the resulting parameters are: *{'algorithm': 'auto', 'n_neighbors': 19, 'weights': 'distance'}*. The curves training score and cv score do not converge, which suggest overfitting. This could mean poor correlation in the data, e.g. how does the feature "sex" contribute to the distance in kNN? It should have low correlation and should be removed for better results.

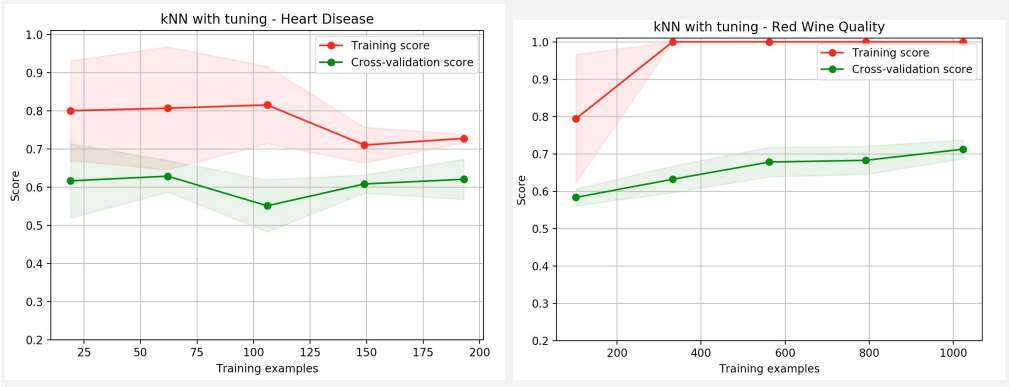


Figure 7 — Heart disease and red wine quality prediction after hyperparameter tuning using kNN.

3.5.2 *Accuracy before and after tuning*

Table 4 shows the accuracy of the test set before and after hyperparameter tuning. There is an increase in accuracy of 7% for heart disease and 12% increase for red wine quality.

Table 4—Accuracy of test set before and after tuning for kNN

	Heart Disease	**Red Wine Quality**
Before hyperparameter tuning	*0.60*	0.62
After hyperparameter tuning	*0.67*	0.74

4 TESTING

Two different datasets, heart disease and red wine quality were used to train and test each algorithm. Each dataset was split 80%/20% for training and testing. The 80% is further split into cross validation chunks. Scikit-Learn ShuffleSplit was used to create cv chunks. Then the estimator, param-grid, and the cross-validation chunks were passed into GridSearch, which finds the best parameter combination. The model is built with the highest accuracy model. The model is finally tested with the test data.

There were some variations produced by each run. The same program can produce different hyperparameters, learning curves, accuracy scores from run to run. Although there are obvious trends, they are not absolute. The part that took the longest is the GridSearch hyperparameter tuning. As parameters or range in each parameter increases, the tuning time grows exponentially.

5 ALGORITHM COMPARISON

5 different models were built and evaluated for speed and accuracy.

Table 5—Accuracy, Train Time, and Query Time of 5 different algorithms - Heart Disease

Algorithm	**Accuracy**	**Train Time**	**Query Time**
Decision Tree	*80%*	2.63ms	1.9ms
MPL Neural Network	*83%*	120ms	2.4ms
Adaboost	*77%*	426ms	45.1ms
SVM	83%	228ms	2.5ms
k-NN	68%	2.3ms	7.1ms

Table 6—Accuracy, Train Time, and Query Time of 5 different algorithms - Red Wine Quality

Algorithm	**Accuracy**	**Train Time**	**Query Time**
Decision Tree	*67%*	3.2ms	1.9ms
MPL Neural Network	*72%*	220ms	2.2ms
Adaboost	*80%*	690ms	64ms
SVM	73%	6720ms	264ms

k-NN	73%	2.7ms	5.1ms

MPL and SVM have the highest accuracy, followed by Decision Tree, Adaboost, and then last kNN. kNN and Decision Tree were the quickest to train, followed by MPL, SVM, then last Adaboost. Decision Tree was fastest to query, followed by MPL, SVM, kNN, then Adaboost.

6 CONCLUSION

Five different classification algorithms were analyzed and tested with two different dataset. Each algorithm was trained with each dataset to produce the best model. GridSearch was used to optimize hyperparameters of each algorithm for the highest accuracy prediction. Training Score and Cross-Validation score were plotted to look at fit (overfit, underfit.) Datasets were split into train/cv/test chunks to aid the process of getting the best models.

The following were observed.

- Hypertuning of parameters required much more time compared to training the model or testing the model.
- There isn't always one clear best algorithm. It depends on the data set. But sometimes one algorithm is better. There can be a bad one.
- There a many "knobs" to adjust, one can spend a lot of time adjusting these knobs.

7 REFERENCES

1. Heart Disease. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation:Robert Detrano, M.D., Ph.D. https://www.kaggle.com/ronitf/heart-disease-uci
2. Red Wine Quality. P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009