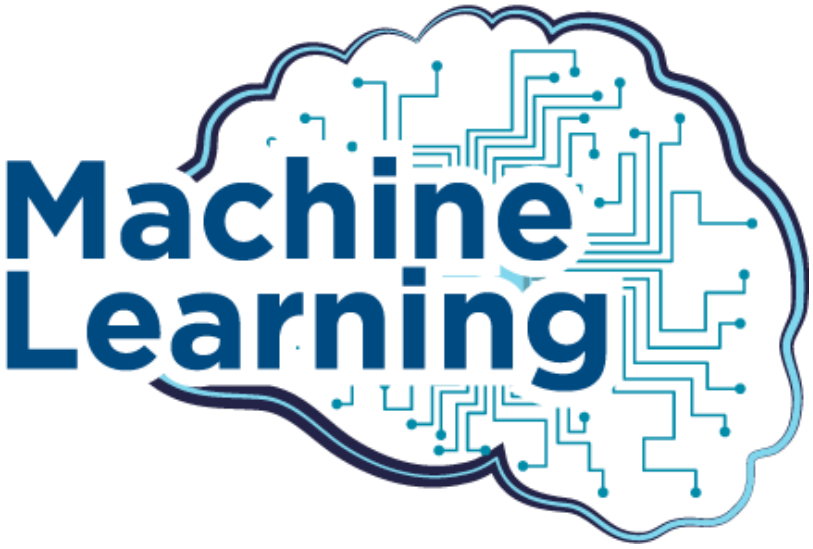


Blood Pressure Disease Prediction Using Machine Learning Algorithms.

Machine Learning Project Report
November 23, 2022

TABLE OF CONTENT

DATASET: -

Dataset we are using in this case study is from Kaggle named as “Blood pressure data for disease prediction” and contains medical information of patients. Dataset contains 2000 patients records and 15 features. We are using “Blood pressure abnormality” as our target variable. It is a Binary class classification problem, where value of class is 0 and 1 (0 means disease not detected and 1 means disease detected). Dataset is balanced and consist of 14 attributes other than target attribute.

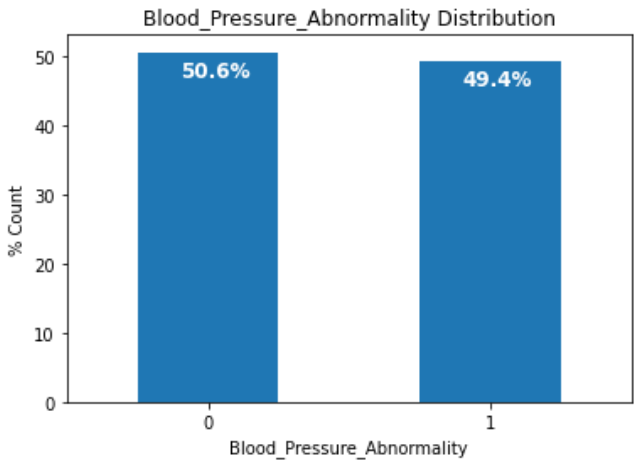
Problem Statement: -

In this project, we are predicting that patient has blood pressure disease or not using several factors such as age, gender, BMI etc. This project will help medical and health professionals to evaluate the patient medical condition more precisely and accurately.

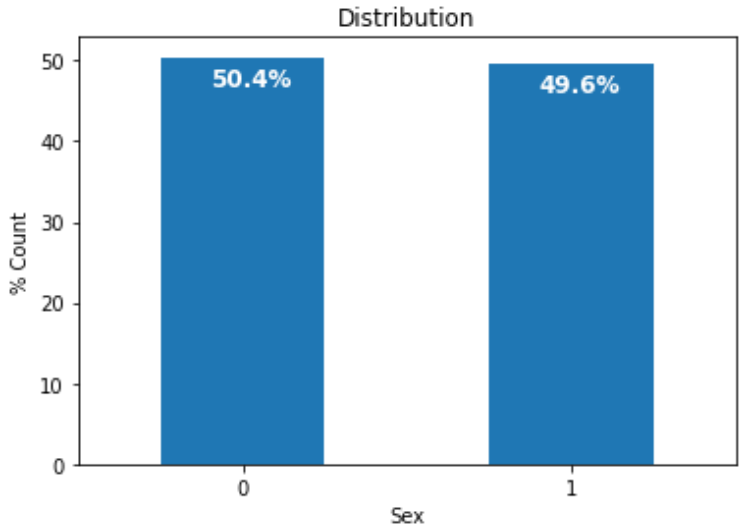
Exploratory Data Analysis: -

EDA is a technique which is used to explore more about the data and fetch some useful information from it using graphical representations. It is to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

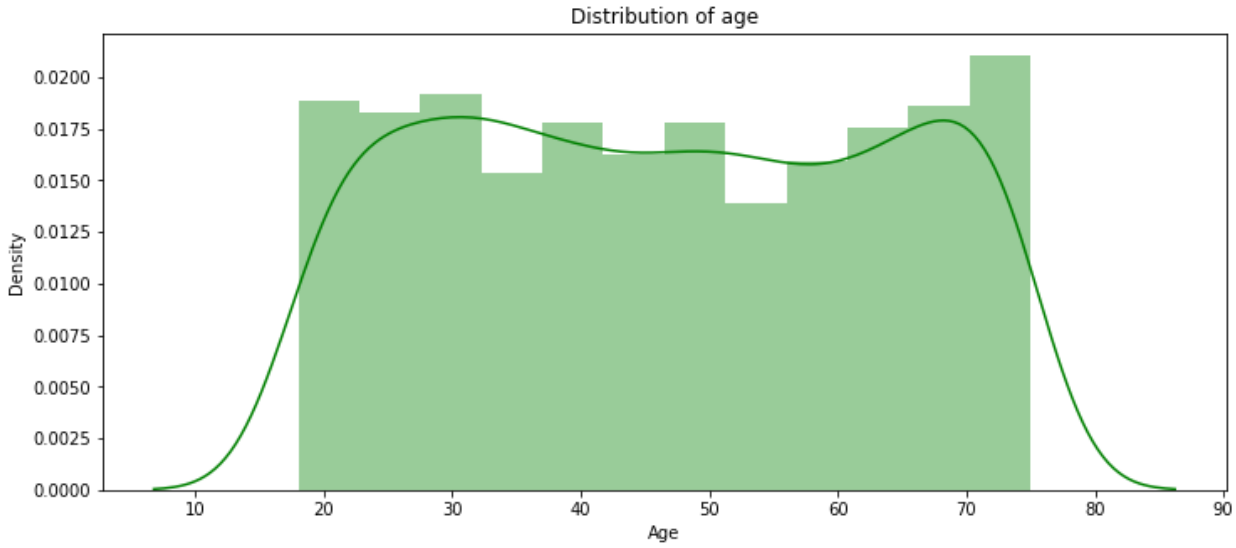
In our data we are also using EDA to get some knowledge about the data. In our dataset we have total 2000 patients’ data, out of which 987 are having blood pressure problem whereas 1013 people are not having this issue.



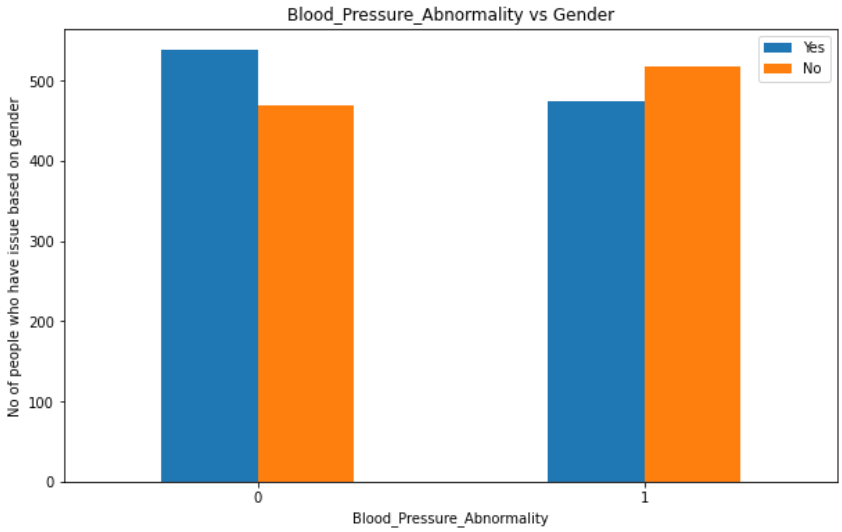
Out of 2000 employees, females are little higher than males. As we can see in this graph below



Most of the people in our dataset are of old age, as we can see in this age distribution graph.



Males are having more Blood pressure issues as compared to the females, as we can see in this bar chart.



Data Pre- Processing: -

Before using the data in the machine learning algorithms, preparing the data is a crucial step. Data preprocessing is a data mining technique used to turn the raw data into a format that is both practical and effective.

For our dataset, we also must perform some data pre-processing. First, we load our dataset using pandas and check shape of our dataset which is 2000 and 15 columns. Dataset is normalized and contain both integer and float values. Data has some null values in some columns which we removed by adding average of values and somewhere add 0 in null spaces. After that we dropped some unnecessary column such index id.

Train – Test Split: -

When machine learning algorithms are used to make predictions on data that was not used to train the model, their performance is estimated using the train-test split technique. Here we are splitting our dataset in train dataset and test dataset with the ratio of 70-30% where 70% is our train dataset and 30% is our test dataset.

METHODOLOGY: -

We are using different machine learning algorithms in this project for blood pressure disease prediction and comparing results of each algorithm to check out which model is fitting and performing well on training data set. We are also using some model enhancement techniques to improve performance of the model. These techniques are K-fold cross validation and ensemble-based model (voting classifier). Our sole purpose is to improve performance of model with fine accuracy.

First, we are applying our models on all 15 attributes and checking performance of our models. After that, we are using feature selection technique and selecting topmost impactful features using lasso regularization approach. In the end, we are using random forest feature selection technique to further shortlist best 9 features and applying different machine learning models on them.

Model Evaluation: -

We are evaluating performance of our model's using accuracy, confusion matrix, AUC graph, ROC_AUC score, sensitivity, specificity, miss rate, precision, and recall values. Before we move forward to model implementation, it is necessary to know following terms.

1) Precision: -

The proportion of True Positives to All Positives is known as precision. For our problem statement, that would be the measure of cases that we correctly identify having a disease out of all the cases having it.

2) Recall: -

The recall is the number of our model rightly identifying True Positives values. Thus, for all the cases who have disease, recall tells us how many we correctly identified as having a disease.

3) Sensitivity: -

The sensitivity of a machine learning model indicates how effectively it can recognize successful examples. Other names for it include the true positive rate (TPR) or recall.

Sensitivity is used to evaluate model performance since it allows us to ascertain how many examples the model was able to correctly identify.

4) Specificity: -

Specificity is the number of true negatives that are rightly predicted by the model.

5) Miss Rate: -

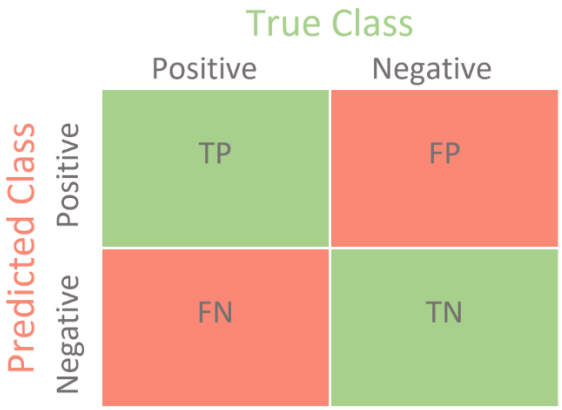
The miss rate is commonly known as false positive rate – is the likelihood that the test will fail to detect a true positive. FN/FN+TP, where FN is the number of false negatives and TP is the number of true positives (FN+TP being the total number of positives), is the formula used to compute it.

6) False Positive Rate: -

The ratio of false positives to true negatives, or false positives to true negatives plus true negatives (FP+TN), is used to compute the false positive rate. It's the likelihood that an error will occur, meaning that a positive result will be reported when a negative value exists.

7) Confusion Matrix: -

A classification problem's prediction outcomes are compiled in a confusion matrix. Count values are used to describe the number of accurate and inaccurate predictions for each class. This is the confusion matrix.



8) Classification Report: -

The accuracy of predictions made by a classification algorithm is evaluated using a classification report. How many of the forecasts came true, and how many were wrong? More specifically, the metrics of a categorization report are predicted using True Positives, False Positives, True Negatives, and False Negatives.

```
from sklearn.metrics import classification_report
print(classification_report(testy,y_pred))
```

[13] ✓ 0.1s

```
              precision    recall  f1-score   support

           0       0.91      0.94      0.92        63
           1       0.93      0.90      0.92        60

    accuracy                           0.92       123
   macro avg       0.92      0.92      0.92       123
weighted avg       0.92      0.92      0.92       123
```

9) F-1 Score: -

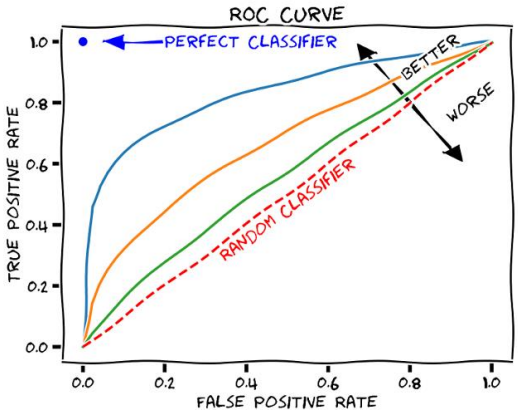
The accuracy of a model on a dataset is gauged by the F-score, also known as the F1-score. It is employed to assess binary categorization schemes that divide examples into "positive" and "negative" categories.

The precision and recall of the model are combined through the F-score.

10) ROC – AUC Curve: -

A measurement tool for binary classification issues is the Receiver Operator Characteristic (ROC) curve. In essence, it separates the "signal" from the "noise" by plotting the TPR against the FPR at different threshold values. The capacity of a classifier to differentiate between classes is measured by the Area Under the Curve (AUC), which is used as a summary of the ROC curve.

The model performs better at differentiating between the positive and negative classes the higher the AUC. The classifier can accurately discriminate between all Positive and Negative class points when AUC = 1. The classifier would be predicting all Negatives as Positives and all Positives as Negatives, however, if the AUC had been 0.



K-FOLD Cross Validation: -

A resampling technique called cross-validation is used to assess machine learning models on a small data sample.

The process contains a single parameter, k, that designates how many groups should be created from a given data sample. As a result, the process is frequently referred to as k-fold cross-validation. When k is set to a specific value, that value may be used in place of k when referring to the model, such as when k=5 is substituted for 5-fold cross-validation.

In applied machine learning, cross-validation is mostly used to gauge how well a machine learning model performs on untrained data. That is, to use a small sample to gauge how the model will generally perform when used to make predictions on data that was not included during the model's training.

In our project, we are using 10-fold cross validation, in which we are calculating cross validation score on our training data. It is returning us accuracies of 10 folds and mean accuracy. K-fold CV help in decreasing model complexity and in some cases, it also improves model results.

Feature Selection: -

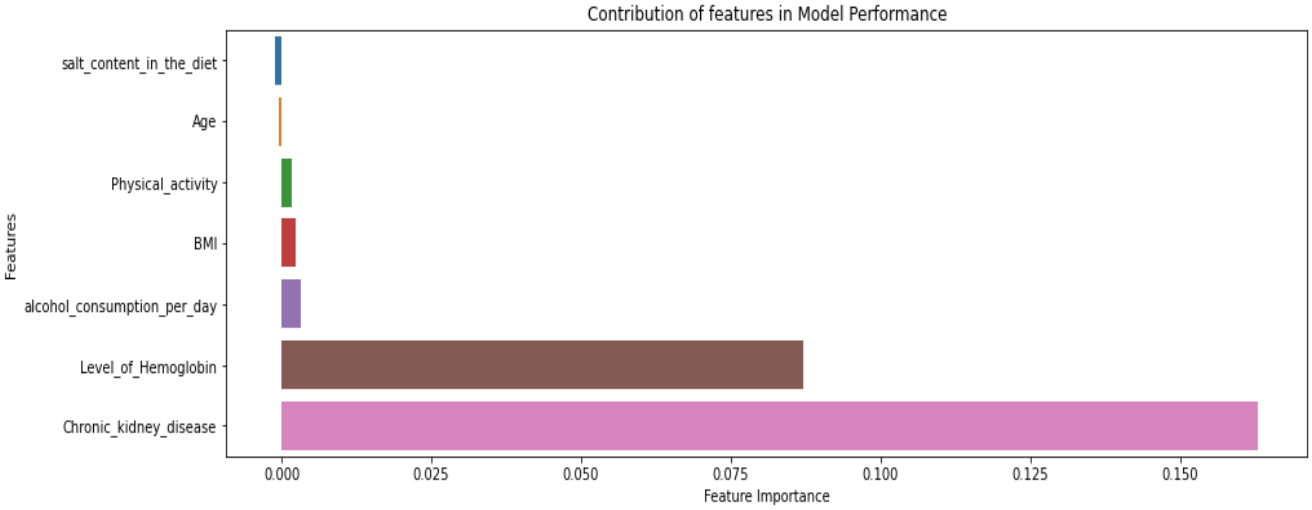
It is almost never the case that all the variables in a dataset can be used to develop a machine learning model in practice. Repetitive factors decrease a classifier's capacity to generalize and may also lower the classifier's overall accuracy. Additionally, a model becomes more complex overall as more variables are added to it.

In this project, we are using 2 feature selection techniques for selecting useful features.

1- Feature Selection using Lasso Regularization

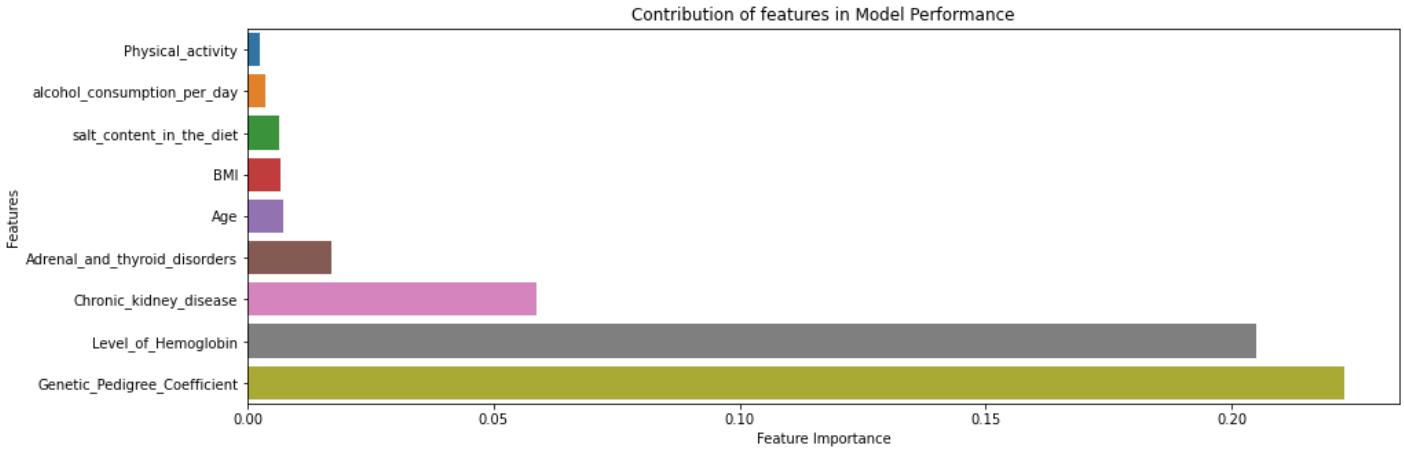
First, we will apply all machine learning models on all our features then after that we will select some features using lasso regularization approach and apply models on selected features.

Lasso Regularization is used for feature selection, by shrinking coefficient to 0. Features whose beta coefficient shrink to 0 will get removed. Mainly lasso is use for reducing overfitting, but it can also be used for feature selection.



2- Feature Selection Using Random Forest SFS: -

We are selecting 9 features in the end using Random Forest sequential feature selector approach. It is one of the most famous and widely use method of selecting useful feature for model. From this we are using selected features and applying models on these features.



Model Implementation: -

At first, we are implementing different machine learning classifier on our dataset first without any feature selection and then after doing section. We have 15 attributes, and our data is 70-30% divided into train and test sets.

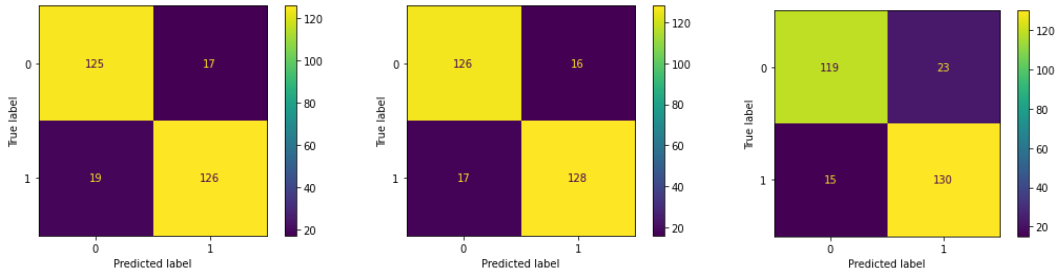
1) Logistic Regression: -

The first model we implement is a classification model called the logistic regression model, which is frequently used to foretell whether a given instance belongs to a particular class or not. It employs the sigmoid function, which returns a number between 0 and 1, to predict the probability of a specific class. In our situation, logistic regression provides accuracy on our test data set of more than 80%. With an AUC value of only 0.8, it provides accuracy of 83%, which is good. Additionally, its performance improved following feature selection.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	Logistic Regression	Logistic Regression	Logistic Regression
Accuracy	68.5%	68.6%	68.16%
AUC Score	0.68	0.688	0.681
Specificity	0.683	0.680	0.697
Sensitivity	0.686	0.69	0.666
Recall	0.686	0.696	0.66
Precision	0.693	0.693	0.69
Miss Rate (FNR)	0.313	0.303	0.33
Miss Rate (FPR)	0.316	0.313	0.302

K-Folds Cross – Validation Confusion Metrics for Logistic Regression: -

Following are the results of K-Fold Cross validation in term of confusion metrics, when applied on 15, 6 and 9 features respectively. First metrics from left is with 15 features, middle one is result with 6 features and left one is confusion metrics result with 9 features.



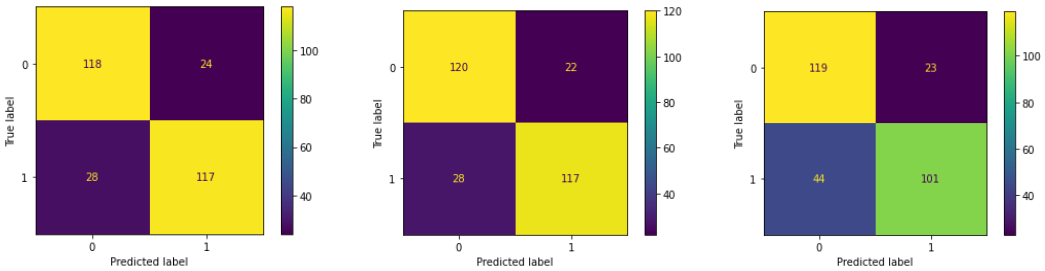
2) Decision Tree: -

The second model we're using on our train dataset is a decision tree that uses decision trees to classify results. It is a well-known machine learning algorithm that is used for both classification and regression issues. Although we used a decision tree on several criteria in our research, the outcome of ROC score prediction was satisfactory. Nevertheless, the accuracy of the decision tree was 84.0% at all features, and it performed better following Random Forest feature selection. These are some of the outcomes from the decision tree implementation.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	Decision Tree	Decision Tree	Decision Tree
Accuracy	84.0%	67.33%	84.33%
AUC Score	0.84	0.67	0.8433
Specificity	0.85	0.687	0.843
Sensitivity	0.83	0.6601	0.84
Recall	0.830	0.66	0.83
Precision	0.852	0.687	0.84
Miss Rate (FNR)	0.169	0.33	0.15
Miss Rate (FPR)	0.149	0.31	0.158

K-Folds Cross – Validation Confusion Metrics for Decision Tree: -

Following are the results of K-Fold Cross validation in term of confusion metrics, when applied on 15, 6 and 9 features respectively. First metrics from left is with 15 features, middle one is result with 6 features and left one is confusion metrics result with 9 features.



3) K- Nearest Neighbor: -

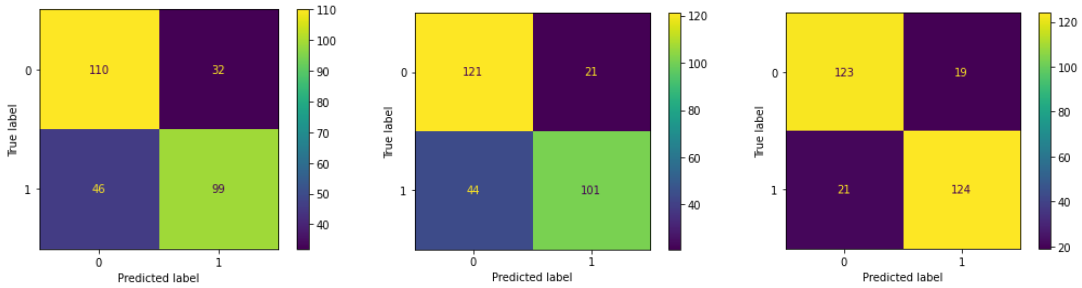
The supervised machine learning approach known as the k-nearest neighbors (KNN) model is straightforward and simple to apply. It can be used to tackle classification and regression issues. KNN searches for nearby nodes and returns results based on them. It determines its closest neighbors by calculating the distance to various neighbors. According to similarity in a particular group of nearby data points, KNN classifies data points.

We are also using KNN-model in this project; we implement model with different number of neighbors and its performance decrease after feature selection.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	KNN	KNN	KNN
Accuracy	49.16%	50.66%	48.36%
AUC Score	0.49	0.508	0.48
Specificity	0.547	0.591	0.55
Sensitivity	0.437	0.424	0.424
Recall	0.437	0.424	0.424
Precision	0.501	0.52	0.498
Miss Rate (FNR)	0.562	0.575	0.575
Miss Rate (FPR)	0.452	0.408	0.445

K-Folds Cross – Validation Confusion Metrics for KNN Classifier: -

Note: The left one is K-fold CV result represented in term of confusion metrics with all features, middle one is with 6 features (after feature selection with Lasso approach) and 3rd one (Right one) is confusion metrics with 9 features (after feature selection with random forest feature selector).



4) Random Forest Classifier: -

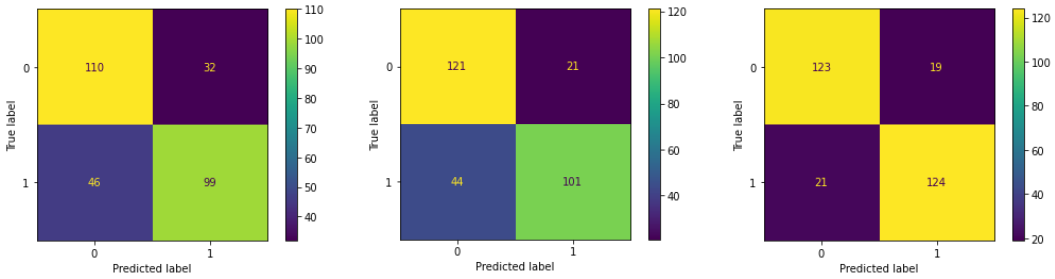
One of the well-known supervised learning methods is random forest. It's an ensemble model that integrates the results of various decision trees to make a learner that is more potent. RF is resistant to overfitting and has a strong noise resistance. The two basic ideas behind Random Forest are bagging and random selection.

In our project, performance of random forest is quite well. It is giving us ROC score better than previous models maybe because it is good with handling large number of features. It is good with high dimension data as we know that it works with making subsets by replacement. So, maybe this is the reason it is giving us good accuracy and roc score on train dataset.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	Random Forest	Random Forest	Random Forest
Accuracy	88.83%	74.33%	88.16%
AUC Score	0.88	0.74	0.88
Specificity	0.89	0.670	0.89
Sensitivity	0.88	0.813	0.869
Recall	0.882	0.813	0.869
Precision	0.897	0.719	0.895
Miss Rate (FNR)	0.117	0.186	0.130
Miss Rate (FPR)	0.105	0.329	0.105

K-Folds Cross – Validation Confusion Metrics for Random Forest: -

Note: The left one is K-fold CV result represented in term of confusion metrics with all features, middle one is with 6 features (after feature selection with Lasso approach) and 3rd one (Right one) is confusion metrics with 9 features (after feature selection with random forest feature selector).



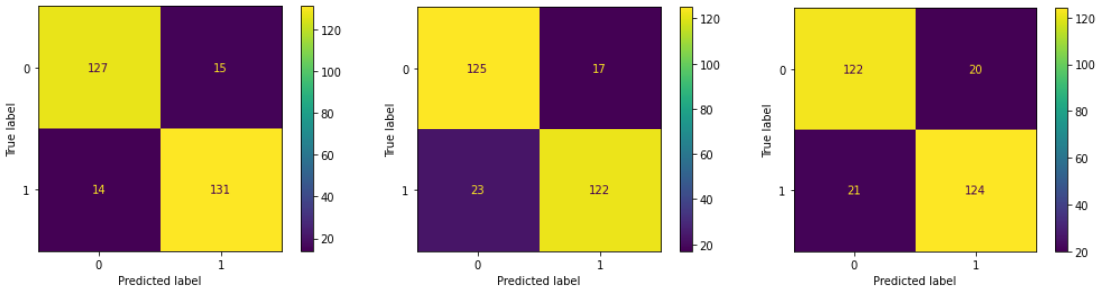
5) Gradient Boosting: -

The supervised machine learning algorithm gradient boosting is used to solve classification and regression problems. By transforming weak learners into strong learners, we can enhance the model prediction of any given algorithm using this ensemble technique. The weak student is successively corrected by predecessors, becoming a strong learner as a result. When compared to other algorithms, gradient boosting classifiers are frequently quick and require less storage.
In this project, we also use Gradient boosting classifier and it's giving us optimum results as compared to other algorithms. Gradient boosting as good as random forest because it converts and enhance weak learner by reducing its error, but if the data is noisy than accuracy of gradient may affect sometime. We implement gradient boosting and it's fitting quite well.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature
Model	Gradient Boosting	Gradient Boosting	Gradient Boosting
Accuracy	89.5%	75.5%	90.5%
AUC Score	0.895	0.75	0.905
Specificity	0.921	0.731	0.942
Sensitivity	0.869	0.777	0.869
Recall	0.869	0.777	0.869
Precision	0.920	0.750	0.939
Miss Rate (FNR)	0.130	0.222	0.130
Miss Rate (FPR)	0.07	0.268	0.057

K-Folds Cross – Validation Confusion Metrics for Gradient Boosting: -

Note: The left one is K-fold CV result represented in term of confusion metrics with all features, middle one is with 6 features (after feature selection with Lasso approach) and 3rd one (Right one) is confusion metrics with 9 features (after feature selection with random forest feature selector).



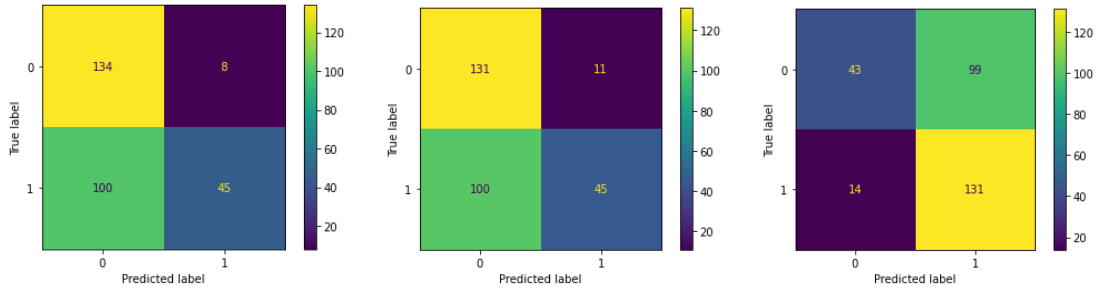
6) Naive Bayes Classifier: -

A group of supervised learning algorithms known as "naive" Bayes methods utilize Bayes' theorem with the "naive" assumption that every pair of features is conditionally independent given the value of the class variable. On our dataset, Naïve bayes is performing below average as compared to the other models.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	Naïve Bayes	Naïve Bayes	Naïve Bayes
Accuracy	75.83%	73.0%	76.0%
AUC Score	0.760	0.732	0.762
Specificity	0.8911	0.833	0.884
Sensitivity	0.6307	0.6307	0.640
Recall	0.6307	0.630	0.640
Precision	0.8577	0.797	0.852
Miss Rate (FNR)	0.369	0.369	0.359
Miss Rate (FPR)	0.108	0.166	0.115

K-Folds Cross – Validation Confusion Metrics for Naïve Bayes Classifier: -

Note: The left one is K-fold CV result represented in term of confusion metrics with all features, middle one is with 6 features (after feature selection with Lasso approach) and 3rd one (Right one) is confusion metrics with 9 features (after feature selection with random forest feature selector).



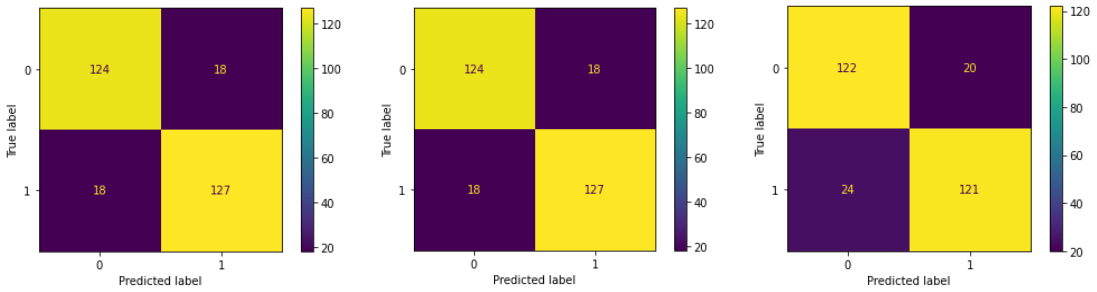
7) XGBOOST: -

XGBOOST is a decision-tree-based ensemble Machine Learning algorithm that uses a framework. In our case study, XGboost is performing very well and giving us accuracy of 91.05% with good roc, precision and recall values after feature selection.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	XGBoost	XGBoost	XGBoost
Accuracy	90.10%	74.83%	90.83%
AUC Score	0.902	0.746	0.908
Specificity	0.92	0.666	0.935
Sensitivity	0.8833	0.826	0.882
Recall	0.8833	0.826	0.882
Precision	0.933	0.720	0.934
Miss Rate (FNR)	0.130	0.173	0.117
Miss Rate (FPR)	0.06	0.333	0.064

K-Folds Cross – Validation Confusion Metrics for XGBoost: -

Note: The left one is K-fold CV result represented in term of confusion metrics with all features, middle one is with 6 features (after feature selection with Lasso approach) and 3rd one (Right one) is confusion metrics with 9 features (after feature selection with random forest feature selector).



Proposed Ensemble Method (Voting Classifier): -

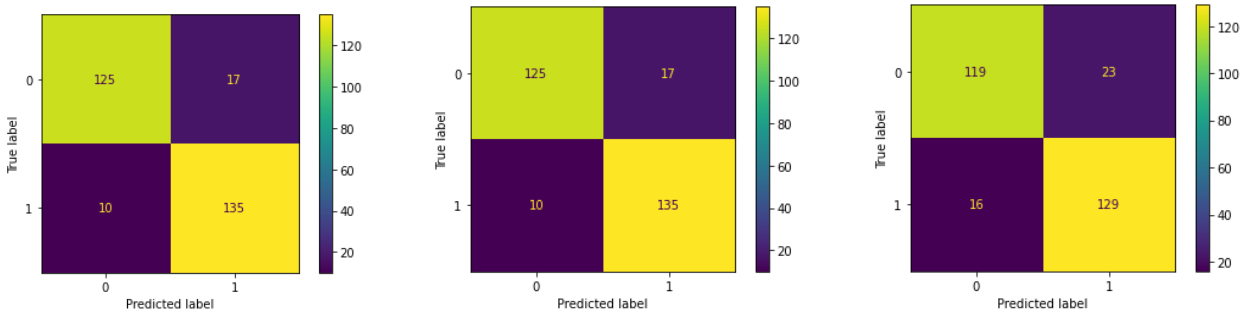
Voting classifier is basically an ensemble technique which unite different base classifiers and based on result of these classifiers it computes new result. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. We are using soft voting technique here which it computes probability based on probabilities and weights associated with different classifiers.

In voting classifier, we are using our best models. We are using 3 models which are random forest models, gradient boosting model and XGBOOST which were giving us best results. So, after combining all our best performing model we create a voting classifier model, which is giving us best result better than all other base model.

Features	Without feature selection	After Lasso Feature selection	After Random Forest Feature Selection
Model	Voting Classifier	Voting Classifier	Voting Classifier
Accuracy	91.0%	75.16%	90.66%
AUC Score	0.910	0.75	0.907
Specificity	0.93	0.744	0.931
Sensitivity	0.88	0.758	0.882
Recall	0.88	0.758	0.882
Precision	0.931	0.755	0.931
Miss Rate (FNR)	0.11	0.241	0.117
Miss Rate (FPR)	0.068	0.255	0.068

K-Folds Cross – Validation Confusion Metrics for Voting Classifier: -

Note: The left one is K-fold CV result represented in term of confusion metrics with 30 features, middle one is with 6 features (after feature selection with Lasso approach) and 3rd one (Right one) is confusion metrics with 9 features.



Results and Insights: -

After applying Lasso-based Feature selection, accuracy and ROC score of each model decrease significantly, more that than we also witnessed highest accuracy of 91% in voting classifier after applying Random Forest feature selection. We selected 9 best performing features using this technique which improves our model's accuracy a lot.

After that, we selected features using Random Forest-based feature selection approach. After that, accuracy of every model increases except Random Forest model where we see nominal drop in accuracy from 90% to 89%. Which is normal in machine learning.

CONCLUSION: -

So, here we Concluded that Voting classifier which is our proposed ensemble model and combination of three best performing models (RF, GB, and XGBoost) is giving us our best overall results in term of all evaluation metrics such as it is giving us best results in term of Accuracy, ROC, Specificity, Precision etc. both before and after feature selection. So, we consider this as our best model.

Among our base models, Gradient Boosting is performing very well best, but its accuracy decreases with 6 features.

Decision tree is good in term of accuracy, but its ROC score and other evaluation metrics are not up to the mark. Best result of LR were achieved on full features selected and its performance decrease after feature selection.

Whereas Gradient boosting, XGBoost and Random Forest are also giving good results but slightly less efficient than Voting classifier. Whereas KNN, and Naïve Bayes are giving average results.

After considering all the results, voting classifier shows the highest performance results to predict blood pressure disease.