Assignment 4 - Data Science
Disease Detection

Mohammad Hossein Basouli

June 6, 2025

Abstract

In this analysis we try to apply multiple machine learning algorithms & techniques to a dataset that contains symtoms related to eleven different diseases. Some key findings of this analysis are: ***Naive Bayes*** *performs really well when the features are binary -* ***Ensembleing*** *of Different Methods usually increases the accuracy, by a few percentages - Reduction of correlated features benefits accuracy of all of the algorithms (in this analysis).*

1 Introduction

1.1 Background:

Machine learning algorithms are commonly used at the intersection of Computer Science and Medicine to identify diseases based on specific symptoms. In this analysis, we will leverage these algorithms to classify 11 distinct diseases based on a given set of symptoms. The dataset, sourced from a Kaggle competition, includes a column labeled ID that identifies the case number, a label column specifying the disease associated with each case, and 64 features, numbered from 0 to 63. The dataset contains 564 rows and, as mentioned earlier, a total of 66 columns.

1.2 Objectives:

1. Perform a basic *Exploratory Data Analysis* step in order to understand the data.

2. Start with selecting a handful of basic classfying algorithms such as *Linear Discriminant Analysis*, *SVM*, *Naive Bayes*, etc. and find the best parameters for training those algorithms.

3. Evaluate your algorithms and see how well they are performing.

4. Start exploring other methods such as *Ensembling of The Previous Methods*, *Bagging* and *Boosting* methods.

5. Try *Feature Engineering* techniques such as *Dimension Reduction*, etc.

6. Examine the models to see what are the most important features.

2 Exploratory Data Analysis

2.1 Correlation Matrix of the Features:

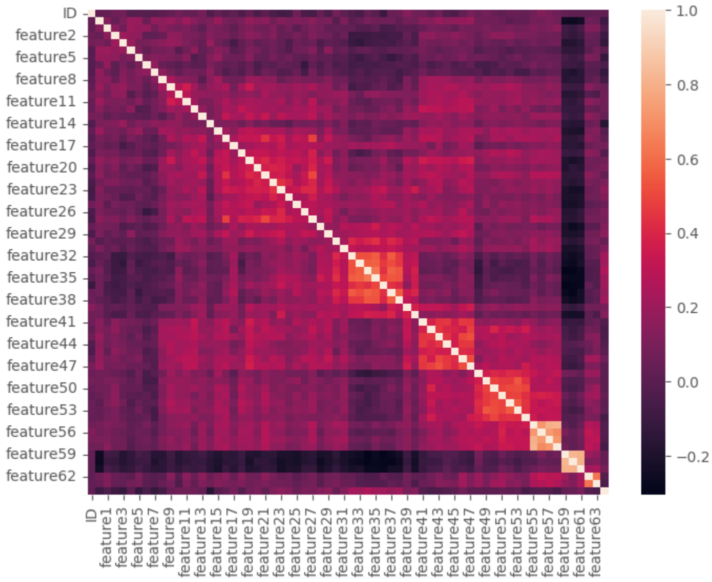

Figure 1: Correlation Matrix of the Features 0 through 63. The heatmap shows that the features *feature56*, *feature57* and *feature58* are highly correlated. This is also true for the features *feature59*, *feature60* and *feature61*.

2.2 Proportion of the Data in Each Class:

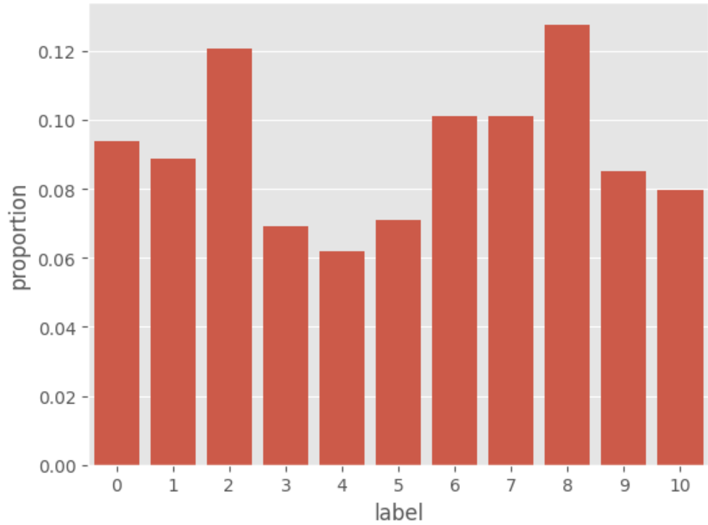

Figure 2: Proportion of The Classes. This barplot shows that the data in the differet classes of the disease is approximately balance.

3 Baseline Models

We will summarize the results that we got with and without feature reduction (we have reduced the set of features to a smaller set, by replacing highly correlated features (Figure 1) by a single one of them.) for each of the models separately. We also can see that our algorithms perform slightly better after the feature reduction.

3.1 Support Vector Machine

3.1.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for SVM...
Fitting 3 folds for each of 24 candidates, totalling 72 fits
✅ Best Params for SVM: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
🎯 Best Score: 0.3127
```

Figure 3: Grid Search Results for **SVM** Before Feature Reduction

```
🔍 Running GridSearchCV for SVM...
Fitting 3 folds for each of 24 candidates, totalling 72 fits
✅ Best Params for SVM: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
🎯 Best Score: 0.3127
```

Figure 4: Grid Search Results for **SVM** After Feature Reduction

3.1.2 Evaluation:

```
SVM
Train Accuracy: 0.7960
Test Accuracy:  0.3717
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.77      0.91      0.83        11
           1       0.30      0.30      0.30        10
           2       0.17      0.29      0.21        14
           3       0.57      0.50      0.53         8
           4       0.60      0.43      0.50         7
           5       0.00      0.00      0.00         8
           6       0.17      0.09      0.12        11
           7       0.78      0.64      0.70        11
           8       0.19      0.21      0.20        14
           9       0.28      0.50      0.36        10
          10       0.40      0.22      0.29         9

    accuracy                           0.37       113
   macro avg       0.38      0.37      0.37       113
weighted avg       0.37      0.37      0.36       113
```

Figure 5: Accuracy Results for **SVM** Before Feature Reduction

```
SVM
Train Accuracy: 0.7982
Test Accuracy:  0.3717
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.71      0.91      0.80        11
           1       0.20      0.20      0.20        10
           2       0.22      0.36      0.27        14
           3       0.57      0.50      0.53         8
           4       0.60      0.43      0.50         7
           5       0.00      0.00      0.00         8
           6       0.25      0.09      0.13        11
           7       0.64      0.64      0.64        11
           8       0.18      0.21      0.19        14
           9       0.29      0.50      0.37        10
          10       0.40      0.22      0.29         9

    accuracy                           0.37       113
   macro avg       0.37      0.37      0.36       113
weighted avg       0.36      0.37      0.35       113
```

Figure 6: Accuracy Results for **SVM** After Feature Reduction

3.2 Logistic Regression

3.2.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for Logistic Regression...
Fitting 3 folds for each of 39 candidates, totalling 117 fits
✅ Best Params for Logistic Regression: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
🎯 Best Score: 0.2994
```

Figure 7: Grid Search Results for **Logistic Regression** Before Feature Reduction

```
🔍 Running GridSearchCV for Logistic Regression...
Fitting 3 folds for each of 39 candidates, totalling 117 fits
✅ Best Params for Logistic Regression: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
🎯 Best Score: 0.2994
```

Figure 8: Grid Search Results for **Logistic Regression** After Feature Reduction

3.2.2 Evaluation:

```
Logistic Regression
Train Accuracy: 0.4479
Test Accuracy:  0.3805
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.56      0.91      0.69        11
           1       0.00      0.00      0.00        10
           2       0.18      0.29      0.22        14
           3       0.57      0.50      0.53         8
           4       0.75      0.43      0.55         7
           5       0.00      0.00      0.00         8
           6       0.33      0.27      0.30        11
           7       0.62      0.73      0.67        11
           8       0.25      0.29      0.27        14
           9       0.36      0.50      0.42        10
          10       0.33      0.22      0.27         9

    accuracy                           0.38       113
   macro avg       0.36      0.38      0.36       113
weighted avg       0.35      0.38      0.35       113
```

Figure 9: Accuracy Results for **Logistic Regression** Before Feature Reduction

```
Logistic Regression
Train Accuracy: 0.4612
Test Accuracy:  0.3540
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.56      0.91      0.69        11
           1       0.00      0.00      0.00        10
           2       0.18      0.29      0.22        14
           3       0.57      0.50      0.53         8
           4       0.60      0.43      0.50         7
           5       0.00      0.00      0.00         8
           6       0.33      0.18      0.24        11
           7       0.53      0.73      0.62        11
           8       0.19      0.21      0.20        14
           9       0.31      0.40      0.35        10
          10       0.40      0.22      0.29         9

    accuracy                           0.35       113
   macro avg       0.33      0.35      0.33       113
weighted avg       0.32      0.35      0.32       113
```

Figure 10: Accuracy Results for **Logistic Regression** After Feature Reduction

3.3 Naive Bayes

3.3.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for Naive Bayes...
Fitting 3 folds for each of 10 candidates, totalling 30 fits
✅ Best Params for Naive Bayes: {'alpha': 0.1, 'fit_prior': True}
🎯 Best Score: 0.3060
```

Figure 11: Grid Search Results for **Naive Bayes** Before Feature Reduction

```
🔍 Running GridSearchCV for Naive Bayes...
Fitting 3 folds for each of 10 candidates, totalling 30 fits
✅ Best Params for Naive Bayes: {'alpha': 0.1, 'fit_prior': True}
🎯 Best Score: 0.3104
```

Figure 12: Grid Search Results for **Naive Bayes** After Feature Reduction

3.3.2 Evaluation:

```
Naive Bayes
Train Accuracy: 0.3792
Test Accuracy:  0.3894
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.77      0.91      0.83        11
           1       0.20      0.20      0.20        10
           2       0.20      0.07      0.11        14
           3       0.56      0.62      0.59         8
           4       0.30      0.43      0.35         7
           5       0.67      0.25      0.36         8
           6       0.33      0.27      0.30        11
           7       0.54      0.64      0.58        11
           8       0.00      0.00      0.00        14
           9       0.27      0.70      0.39        10
          10       0.33      0.44      0.38         9

    accuracy                           0.39       113
   macro avg       0.38      0.41      0.37       113
weighted avg       0.36      0.39      0.35       113
```

Figure 13: Accuracy Results for **Naive Bayes** Before Feature Reduction

```
Naive Bayes
Train Accuracy: 0.3814
Test Accuracy:  0.3982
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.67      0.91      0.77        11
           1       0.20      0.20      0.20        10
           2       0.33      0.14      0.20        14
           3       0.50      0.62      0.56         8
           4       0.33      0.43      0.38         7
           5       1.00      0.25      0.40         8
           6       0.33      0.27      0.30        11
           7       0.64      0.64      0.64        11
           8       0.00      0.00      0.00        14
           9       0.28      0.70      0.40        10
          10       0.33      0.44      0.38         9

    accuracy                           0.40       113
   macro avg       0.42      0.42      0.38       113
weighted avg       0.40      0.40      0.37       113
```

Figure 14: Accuracy Results for **Naive Bayes** After Feature Reduction

3.4 Linear Discriminant Analysis

3.4.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for LDA...
Fitting 3 folds for each of 11 candidates, totalling 33 fits
✅ Best Params for LDA: {'shrinkage': 1.0, 'solver': 'lsqr'}
🎯 Best Score: 0.3016
```

Figure 15: Grid Search Results for **Linear Discriminant Analysis** Before Feature Reduction

```
🔍 Running GridSearchCV for LDA...
Fitting 3 folds for each of 11 candidates, totalling 33 fits
✅ Best Params for LDA: {'shrinkage': 1.0, 'solver': 'lsqr'}
🎯 Best Score: 0.3015
```

Figure 16: Grid Search Results for **Linear Discriminant Analysis** After Feature Reduction

3.4.2 Evaluation:

```
LDA
Train Accuracy: 0.3636
Test Accuracy:  0.3717
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.67      0.91      0.77        11
           1       0.00      0.00      0.00        10
           2       0.33      0.14      0.20        14
           3       0.50      0.50      0.50         8
           4       0.30      0.43      0.35         7
           5       0.33      0.12      0.18         8
           6       0.33      0.27      0.30        11
           7       0.57      0.73      0.64        11
           8       0.00      0.00      0.00        14
           9       0.28      0.70      0.40        10
          10       0.33      0.44      0.38         9

    accuracy                           0.37       113
   macro avg       0.33      0.39      0.34       113
weighted avg       0.32      0.37      0.33       113
```

Figure 17: Accuracy Results for **Linear Discriminant Analysis** Before Feature Reduction

```
LDA
Train Accuracy: 0.3725
Test Accuracy:  0.3717
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.62      0.91      0.74        11
           1       0.00      0.00      0.00        10
           2       0.40      0.14      0.21        14
           3       0.50      0.62      0.56         8
           4       0.33      0.43      0.38         7
           5       1.00      0.12      0.22         8
           6       0.33      0.27      0.30        11
           7       0.58      0.64      0.61        11
           8       0.00      0.00      0.00        14
           9       0.28      0.70      0.40        10
          10       0.33      0.44      0.38         9

    accuracy                           0.37       113
   macro avg       0.40      0.39      0.34       113
weighted avg       0.38      0.37      0.33       113
```

Figure 18: Accuracy Results for **Linear Discriminant Analysis** After Feature Reduction

4 Ensemble Models

We will summarize the results that we got with and without feature reduction (we have reduced the set of features to a smaller set, by replacing highly correlated features (Figure 1) by a single one of them.) for each of the ensemble models separately. We can see that our algorithms perform slightly better after the feature reduction. We also observe that the performance of the ensembling methods is so much higher than the baseline models in general.

4.1 Random Forest

4.1.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for Random Forest...
Fitting 3 folds for each of 24 candidates, totalling 72 fits
✅ Best Params for Random Forest: {'bootstrap': True, 'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200}
🎯 Best Score: 0.3061
```

Figure 19: Grid Search Results for **Random Forest** Before Feature Reduction

```
🔍 Running GridSearchCV for Random Forest...
Fitting 3 folds for each of 24 candidates, totalling 72 fits
✅ Best Params for Random Forest: {'bootstrap': True, 'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}
🎯 Best Score: 0.3127
```

Figure 20: Grid Search Results for **Random Forest** After Feature Reduction

4.1.2 Evaluation:

```
Random Forest
Train Accuracy: 0.8980
Test Accuracy:  0.3274
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.62      0.91      0.74        11
           1       0.00      0.00      0.00        10
           2       0.18      0.21      0.19        14
           3       0.57      0.50      0.53         8
           4       0.50      0.43      0.46         7
           5       0.33      0.12      0.18         8
           6       0.11      0.09      0.10        11
           7       0.64      0.64      0.64        11
           8       0.07      0.07      0.07        14
           9       0.29      0.50      0.37        10
          10       0.29      0.22      0.25         9

    accuracy                           0.33       113
   macro avg       0.33      0.34      0.32       113
weighted avg       0.31      0.33      0.31       113
```

Figure 21: Accuracy Results for **Random Forest** Before Feature Reduction

```
Random Forest
Train Accuracy: 0.9446
Test Accuracy:  0.3628
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.62      0.91      0.74        11
           1       0.00      0.00      0.00        10
           2       0.31      0.36      0.33        14
           3       0.50      0.50      0.50         8
           4       0.60      0.43      0.50         7
           5       0.00      0.00      0.00         8
           6       0.14      0.09      0.11        11
           7       0.54      0.64      0.58        11
           8       0.17      0.21      0.19        14
           9       0.40      0.60      0.48        10
          10       0.29      0.22      0.25         9

    accuracy                           0.36       113
   macro avg       0.32      0.36      0.34       113
weighted avg       0.32      0.36      0.33       113
```

Figure 22: Accuracy Results for **Random Forest** After Feature Reduction

4.2 Ada Boost

4.2.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for AdaBoost...
Fitting 3 folds for each of 108 candidates, totalling 324 fits
✅ Best Params for AdaBoost: {'estimator__max_depth': 5, 'estimator__min_samples_leaf': 2, 'estimator__min_samples_split': 2, 'learning_rate': 1.0, 'n_estimators': 100}
🎯 Best Score: 0.2771
```

Figure 23: Grid Search Results for **Ada Boost** Before Feature Reduction

```
🔍 Running GridSearchCV for AdaBoost...
Fitting 3 folds for each of 108 candidates, totalling 324 fits
✅ Best Params for AdaBoost: {'estimator__max_depth': 5, 'estimator__min_samples_leaf': 1, 'estimator__min_samples_split': 2, 'learning_rate': 1.0, 'n_estimators': 200}
🎯 Best Score: 0.2772
```

Figure 24: Grid Search Results for **Ada Boost** After Feature Reduction

4.2.2 Evaluation:

```
AdaBoost
Train Accuracy: 0.9911
Test Accuracy:  0.3009
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.75      0.82      0.78        11
           1       0.13      0.20      0.16        10
           2       0.12      0.14      0.13        14
           3       0.40      0.25      0.31         8
           4       0.50      0.14      0.22         7
           5       0.33      0.25      0.29         8
           6       0.14      0.09      0.11        11
           7       0.57      0.36      0.44        11
           8       0.12      0.14      0.13        14
           9       0.45      0.50      0.48        10
          10       0.29      0.44      0.35         9

    accuracy                           0.30       113
   macro avg       0.35      0.30      0.31       113
weighted avg       0.33      0.30      0.30       113
```

Figure 25: Accuracy Results for **Ada Boost** Before Feature Reduction

```
AdaBoost
Train Accuracy: 0.9933
Test Accuracy:  0.3451
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.82      0.82      0.82        11
           1       0.21      0.30      0.25        10
           2       0.23      0.21      0.22        14
           3       0.71      0.62      0.67         8
           4       0.50      0.14      0.22         7
           5       0.14      0.12      0.13         8
           6       0.29      0.18      0.22        11
           7       0.50      0.55      0.52        11
           8       0.12      0.14      0.13        14
           9       0.38      0.50      0.43        10
          10       0.18      0.22      0.20         9

    accuracy                           0.35       113
   macro avg       0.37      0.35      0.35       113
weighted avg       0.36      0.35      0.34       113
```

Figure 26: Accuracy Results for **Ada Boost** After Feature Reduction

4.3 XG Boost

4.3.1 Hyperparameter Tuning:

```
🔍 Running GridSearchCV for XGBoost...
Fitting 3 folds for each of 256 candidates, totalling 768 fits
✅ Best Params for XGBoost: {'colsample_bytree': 0.7, 'gamma': 0, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 1, 'n_estimators': 100, 'reg_alpha': 1.0, 'reg_lambda': 1.0, 'subsample': 0.7}
🎯 Best Score: 0.2972
```

Figure 27: Grid Search Results for **XG Boost** Before Feature Reduction

```
🔍 Running GridSearchCV for XGBoost...
Fitting 3 folds for each of 256 candidates, totalling 768 fits
✅ Best Params for XGBoost: {'colsample_bytree': 0.7, 'gamma': 0, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 3, 'n_estimators': 100, 'reg_alpha': 1.0, 'reg_lambda': 1.0, 'subsample': 0.7}
🎯 Best Score: 0.2861
```

Figure 28: Grid Search Results for **XG Boost** After Feature Reduction

4.3.2 Evaluation:

```
XGBoost
Train Accuracy: 0.8537
Test Accuracy:  0.3540
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.77      0.91      0.83        11
           1       0.11      0.10      0.11        10
           2       0.16      0.21      0.18        14
           3       0.56      0.62      0.59         8
           4       0.50      0.29      0.36         7
           5       0.00      0.00      0.00         8
           6       0.22      0.18      0.20        11
           7       0.70      0.64      0.67        11
           8       0.25      0.29      0.27        14
           9       0.31      0.40      0.35        10
          10       0.33      0.22      0.27         9

    accuracy                           0.35       113
   macro avg       0.36      0.35      0.35       113
weighted avg       0.35      0.35      0.35       113
```

Figure 29: Accuracy Results for **XG Boost** Before Feature Reduction

```
XGBoost
Train Accuracy: 0.7095
Test Accuracy:  0.3274
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.71      0.91      0.80        11
           1       0.12      0.10      0.11        10
           2       0.11      0.14      0.12        14
           3       0.45      0.62      0.53         8
           4       0.50      0.29      0.36         7
           5       0.00      0.00      0.00         8
           6       0.17      0.09      0.12        11
           7       0.58      0.64      0.61        11
           8       0.31      0.29      0.30        14
           9       0.25      0.30      0.27        10
          10       0.20      0.22      0.21         9

    accuracy                           0.33       113
   macro avg       0.31      0.33      0.31       113
weighted avg       0.31      0.33      0.31       113
```

Figure 30: Accuracy Results for **XG Boost** After Feature Reduction

4.4 Voting Between All of The Models

4.4.1 Evaluation:

```
🧠 VotingClassifier:
Train Accuracy: 0.4124
Test Accuracy:  0.3628
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.67      0.91      0.77        11
           1       0.00      0.00      0.00        10
           2       0.33      0.21      0.26        14
           3       0.50      0.50      0.50         8
           4       0.33      0.43      0.38         7
           5       0.25      0.12      0.17         8
           6       0.33      0.27      0.30        11
           7       0.54      0.64      0.58        11
           8       0.00      0.00      0.00        14
           9       0.29      0.70      0.41        10
          10       0.27      0.33      0.30         9

    accuracy                           0.36       113
   macro avg       0.32      0.37      0.33       113
weighted avg       0.31      0.36      0.32       113
```

Figure 31: Accuracy Results for **Voting Classifier** (Mix of all of the models) Before Feature Reduction

```
🧠 VotingClassifier:
Train Accuracy: 0.4169
Test Accuracy:  0.3717
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.62      0.91      0.74        11
           1       0.11      0.10      0.11        10
           2       0.25      0.14      0.18        14
           3       0.50      0.62      0.56         8
           4       0.38      0.43      0.40         7
           5       0.50      0.12      0.20         8
           6       0.33      0.27      0.30        11
           7       0.64      0.64      0.64        11
           8       0.00      0.00      0.00        14
           9       0.29      0.70      0.41        10
          10       0.27      0.33      0.30         9

    accuracy                           0.37       113
   macro avg       0.35      0.39      0.35       113
weighted avg       0.34      0.37      0.33       113
```

Figure 32: Accuracy Results for **XG Boost** (Mix of all of the models) After Feature Reduction

5 Feature Importance:

5.1 Feature Importance via MDI in Random Forest:

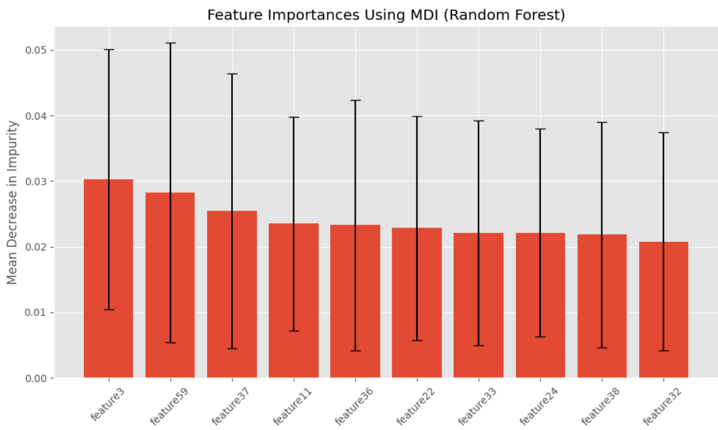

Figure 33: Feature Importance via MDI in **Random Forest**

5.2 Feature Importance via Permutation Test in Random Forest:

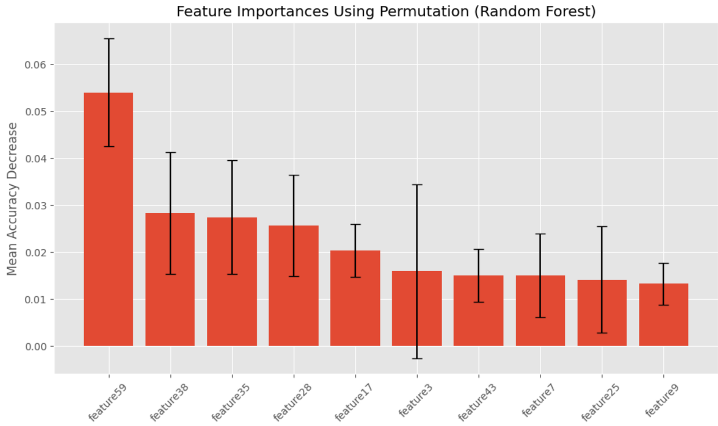

Figure 34: Feature Importance via Permutation Test in **Random Forest**

6 Conclusion:

Our conclusion is that the **Ensembling Methods** such as **Random Forest**, **Voting**, etc outperform single, baseline methods like **Logistic Regression**, **LDA**, etc. in general. Also we saw that the feature reduction worked pretty well on this dataset and led to better results.