Table of Contents

SUPPORT2 Data Mining and Modeling

Project Overview

This project provides a comprehensive pipeline for data preprocessing, visualization, outlier detection, clustering, and classification, specifically designed for the **SUPPORT2 dataset** from the UCI Machine Learning Repository. The goal is to streamline exploratory data analysis (EDA), detect patterns in data, handle outliers, and build robust machine-learning models for classification tasks.

The **UCI Machine Learning Repository** describes this dataset as follows:

> "[The dataset] comprises 9,105 individual critically ill patients across 5 United States medical centers, accessioned throughout 1989–1991 and 1992–1994. Each row concerns hospitalized patient records who met the inclusion and exclusion criteria for nine disease categories: acute respiratory failure, chronic obstructive pulmonary disease, congestive heart failure, liver disease, coma, colon cancer, lung cancer, multiple organ system failure with malignancy, and multiple organ system failure with sepsis. The goal is to determine these patients' 2- and 6-month survival rates based on several physiologic, demographic, and disease severity information.[1]"

We hope that our trained machine learning models using this dataset can assist doctors in making difficult decisions regarding whether to proceed with the assisted suicide procedure or continue intensive care in countries where assisted suicide is legal. This dataset can also be used as a tool to justify patients' Do Not Resuscitate (DNR) decisions.

Model Prediction

For this project, the classification task aims to predict patient survival outcomes based on their medical records. The target labels are defined as three categories: **death in hospital**, indicating the patient passed away during hospitalization; **death out of hospital**, where the patient passed away after being discharged; and **survival**, where the patient lived beyond the observation period which is 180 days [1].

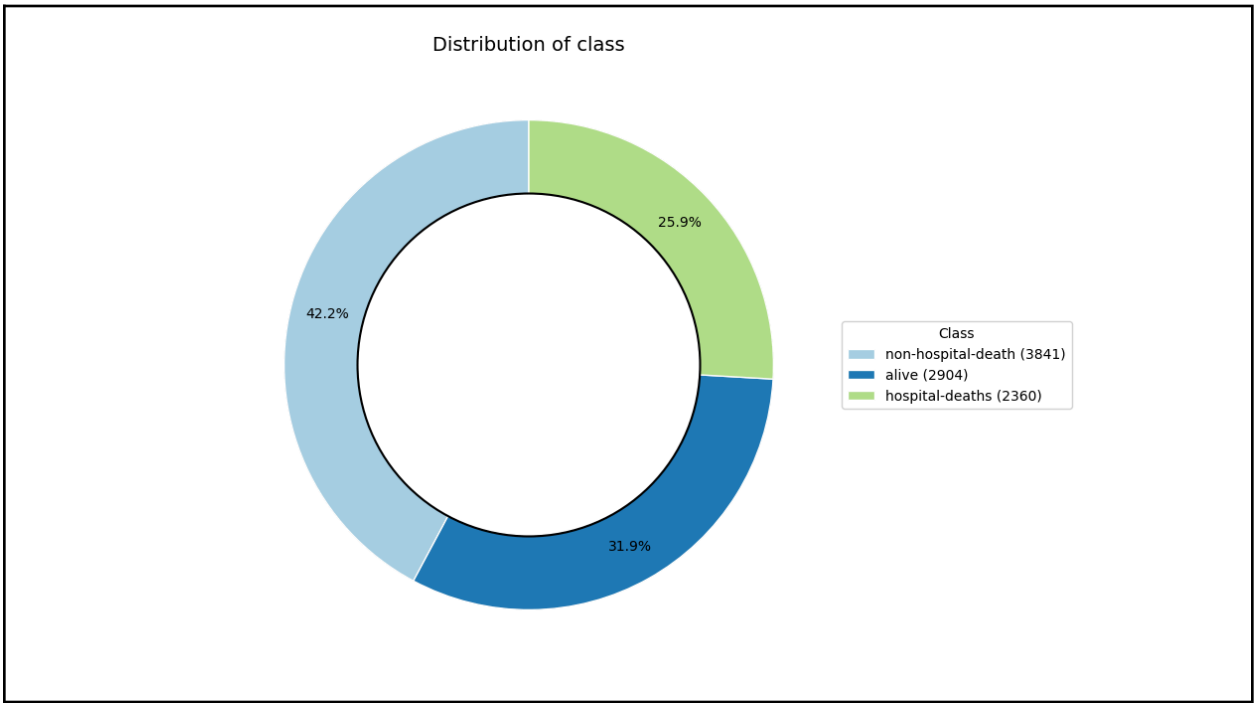


Fig1. The distribution of the target labels in the dataset. Note a slight imbalance in the class distribution.

Key Components of the Project

---

1. Data Preprocessing

Overview

- Cleans and prepares the SUPPORT2 dataset for analysis and modeling.
- Handles missing data via imputation or column removal.
- Encodes categorical features using one-hot encoding.
- Normalizes numerical features for consistency.

- Logs preprocessing steps and saves processed datasets for traceability.

Observations

The table below shows initial observations encountered before imputation, hot-one encoding, and Z-score normalization.

Column Names	Number of Unique Values	Number of Missing values	Type of Feature	Was Dropped Due to High Number of Missing values?
age	7323	0	Numerical	NO
num.co	10	0	Numerical	NO
edu	34	1634	Numerical	NO
charges	8504	172	Numerical	NO
totcst	8200	888	Numerical	NO
avtisst	355	82	Numerical	NO
scoma	12	1	Numerical	No
sps	605	1	Numerical	NO
aps	126	1	Numerical	NO
surv2m	950	1	Numerical	NO
surv6m	937	1	Numerical	NO
hday	85	0	Numerical	NO
diabetes	2	0	Numerical	NO
dementia	2	0	Numerical	NO
prg2m	54	1649	Numerical	NO
prg6m	90	1633	Numerical	NO
dnrday	180	30	Numerical	NO
meanbp	165	1	Numerical	NO
wblc	502	212	Numerical	NO

hrt	187	1	Numerical	NO
resp	67	1	Numerical	NO
temp	99	1	Numerical	NO
pafi	1490	2325	Numerical	NO
bili	298	2601	Numerical	NO
crea	133	67	Numerical	NO
sod	80	1	Numerical	NO
adlsc	1735	0	Numerical	NO
sex	2	0	Categorical	NO
dzgroup	8	0	Categorical	NO
dzclass	4	0	Categorical	NO
race	5	42	Categorical	NO
ca	3	0	Categorical	NO
dnr	3	30	Categorical	NO
income	NaN	2982	NaN	YES
totmcst	NaN	3475	NaN	YES
alb	NaN	3372	NaN	YES
glucose	NaN	4500	NaN	YES
bun	NaN	4352	NaN	YES
urine	NaN	4862	NaN	YES
adlp	NaN	5641	NaN	YES
adls	NaN	2867	NaN	YES

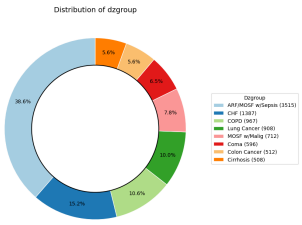
Notes

- The threshold used to decide whether to drop a column was chosen to be 30%, i.e, if a column has more than 30% missing values it would be dropped before imputation. If we proceed with imputation without dropping such columns, a significant portion of these features would have the same value introducing bias in the dataset.
- To impute the missing values we averaged the existing values in the columns within each combination of the labels. Each combination of the labels ‘death’ and ‘hospdeath’ will result in a prediction class as explained in the model prediction section above.
- Hot-One encoding was performed on the categorical features to transform them into numerical values as many ML models require only numerical input.
- The normalization method used on the dataset is Z-score.
- Label encoding was applied to transform the labels from categories such as **['non-hospital death,' 'hospital death,' and 'alive']** into numerical values. [0 , 1 ,2]

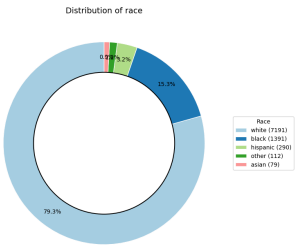
2. Data Visualization (EDA)

- **Pie Charts: Displays distributions of categorical features**

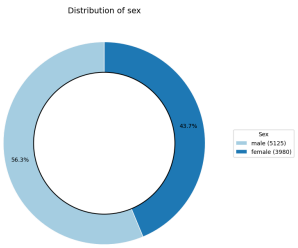
  Below we have included a small portion of the pie charts produced from categorical features before the data preprocessing stage



the percentage of each type of disease



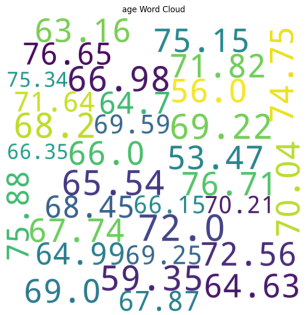
percentage of patients’ race:

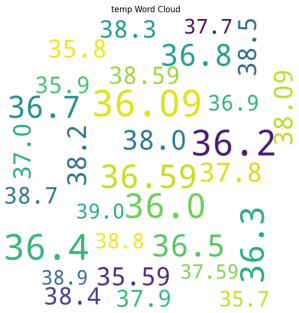


percentage of patients’ sex:

- **Word Clouds: Highlights frequencies of numerical features**

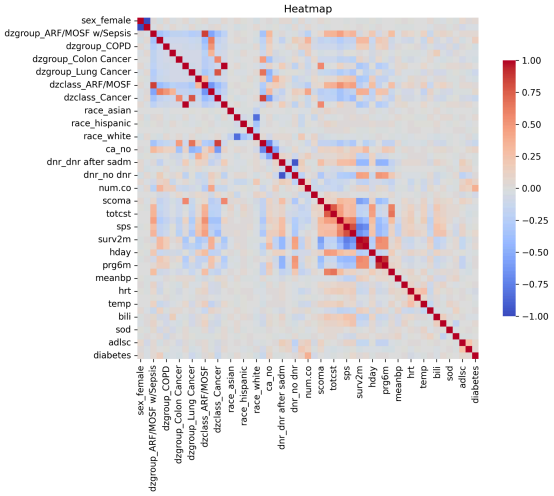
  "Below are word clouds for patients’ age and temperature, showing that the most frequent age range is 60 to 75 years and the most common temperature is around 36 degrees."





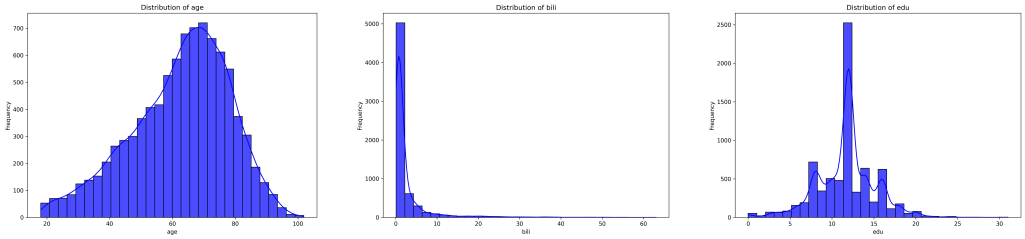
- **Heatmaps: Showcased the correlation among features**

  The heatmap shows variable correlations, with red for positive, blue for negative, and light colors for weak relationships. Overall, correlations are mostly weak **(diagonal dominant matrix)**, with a few strong clusters. Notably, disease class and disease group strongly correlate, suggesting redundancy and disease class could be dropped. Smaller clusters appear among clinical measures like **scoma, meanbp, and temp.**



- **Histograms: Visualizes distributions of numerical features**

  Some features have a near-normal distribution, while others show skewed or non-normal patterns with outliers or heavy tails. Below, we include histograms for selected features based on their domain relevance to visualize these distributions.



Age Distribution | Tbil (Total Bilirubin) Distribution | Edu (Education Level)

- To reduce the impact of skewness on our model predictions, we applied Z-score normalization to the dataset. This technique standardizes the features by centering them around a mean of 0 and scaling them to have a standard deviation of 1. Doing so minimizes the influence of extreme values and ensures that features are on a comparable scale, improving the performance and stability of machine learning models.

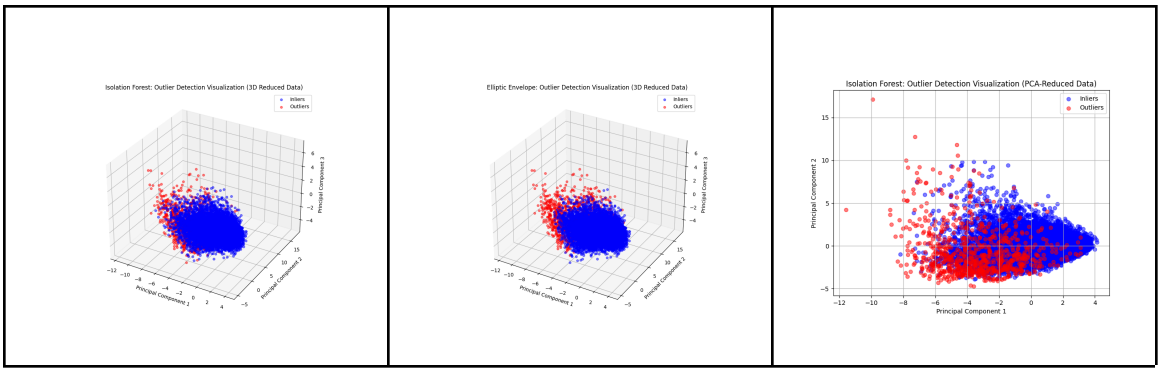
3. Outlier Detection

Overview

- Detects outliers in the dataset using the Isolation Forest and Elliptic Envelope algorithm.
- Reduces dimensionality using PCA for effective visualization.
- Generates 3D scatter plots to highlight inliers and outliers.

Observations

When closely examining the 3D plot generated by the Elliptic Envelope algorithm, particularly the boundary between the blue and red dots, and comparing it to the 3D plot produced by the Isolation Forest algorithm, a clear difference is clear. The Elliptic Envelope identifies more points as outliers. Specifically, some points that the Isolation Forest classified as inliers are instead classified as outliers by the Elliptic Envelope, reflecting its stricter criteria for outlier detection.



The reason why the Elliptic Envelope algorithm is stricter as compared to the Isolation Forest algorithm when it comes to determining outliers and inliers is that it assumes the data follows a multivariate Gaussian distribution and fits an ellipse around the data points. Any points outside this defined boundary are considered outliers, making it stricter in identifying deviations from the assumed distribution. In contrast, the Isolation Forest algorithm is more flexible, as it isolates points based on random splits in the feature space, which may result in fewer points being classified as outliers.

We selected the Elliptic Envelope as one of our outlier detection algorithms because, during the EDA phase of the project, we observed that several features showcased a near-normal distribution.

4. Clustering :

Overview:

- **K-Means:** Detects clusters by partitioning data into kkk groups based on proximity to the nearest cluster mean.
- **DBSCAN**: Detects clusters and outliers based on density.
- **Evaluate** clustering performance using Silhouette scores.
- **Reduce** dimensionality via PCA for 3D visualization of clusters.

K-Means *:* when the number of clusters are chosen to be 2. Choosing the number of clusters to be two will result in a ***silhouette score of 0.1741***, that optimal ***K value = 2*** has been determined by performing the Elbow Method by calculating the Within-Cluster Sum of Squares. ***K-Means*** assumes the underlying data naturally splits into the number of clusters provided, and forces certain data points into clusters even if they do not belong to the cluster. K-means also assumes that the shape of the clusters are spherical (***convex***) and may not perform as well if the clusters have a more irregular shape (***non-convex***).

DBSCAN : The silhouette score for DBSCAN is calculated to be ***0.3068***. The parameters chosen for this algorithm are ***5 for 'eps' and 5 for 'min_samples_split'***. (To read more about the

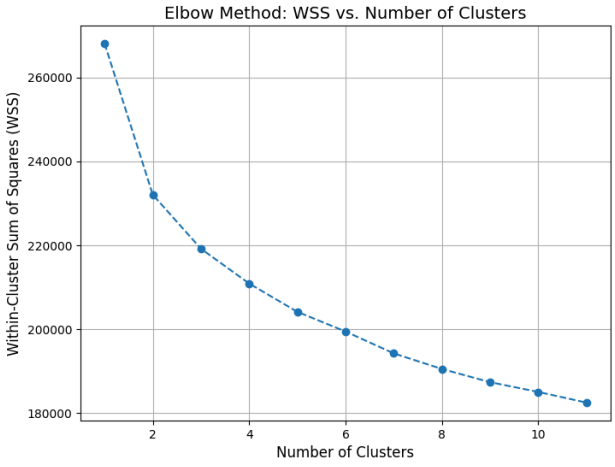
hyperparameter tuning for this algorithm refer to the Notes section down below). ***DBSCAN*** uses density as a measure of determining whether a particular data point falls into a cluster and even specifies some of the data points as noise if they do not show similarities with the points already in the clusters. By not forcing the outlier points into clusters.Thus, ***DBSCAN*** is more effective for datasets that do not follow spherical (non-convex) structures for the clusters.

Algorithm	Silhouette scores
K-Means	***0.1741***
DBSCAN	***0.3068***

The silhouette scores for K-MEANS and DBSCAN under the optimal parameters mentioned above

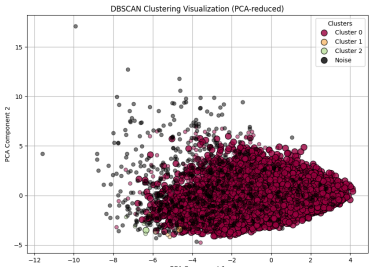
Notes:

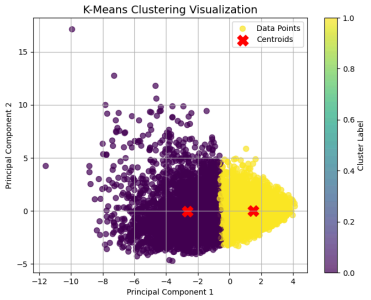
The figure below shows the ***Within-Cluster Sum of Squares*** vs. Number of Clusters chosen for ***K-Means***. This figure is used for the ***Elbow Methods*** to determine the optimal number of clusters to be chosen for the K-Means algorithm. By observing the graph, it can be determined that at ***2 on the x-axis the smooth curve breaks***, specifying that 2 is the optimal number of clusters to be chosen. We have calculated the Silhouette Scores for higher numbers of clusters. With each incrementation in the number of clusters, the Silhouette Score drops dramatically indicating that ***we can not do better than 0.1741 for the Silhouette Score provided by 2 clusters***.

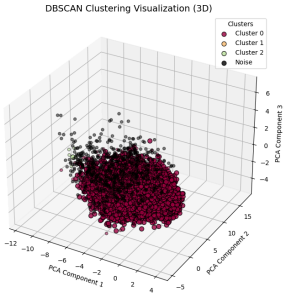


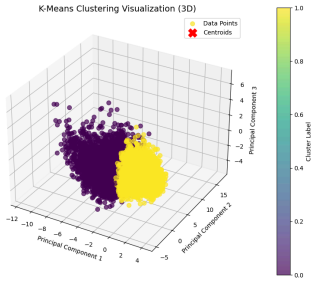
Initially, we performed a ***manual grid search*** for ranges of possible values to determine the optimal values for DBSCAN parameters. ***The silhouette scores looked quite promising with the best value around 0.66.*** However after observing the ***visualization*** of the clusters, we noticed the ***parameters provided by the grid search results in a single clustering of all data points.*** The conclusion we made was ***that the objective of the manual grid search is to maximize the***

silhouette score of the clustering, and sometimes this results in all the data points being put into the same cluster. Therefore, we have decided to experiment with different values for the DBSCAN parameters and decided to choose ***5 for both 'eps' and 'min_sampes_split' as it shows the most promising Silhouette Score*** without putting all the data points in the same cluster.









5. Classification

Overview

- Trains and evaluates multiple machine learning models:

  Baselines : Decision Tree , SVM , k-NN , Naive Bayes
  Ensambles : Random Forest , XGboost
  Bonus : FeedForwd Nural Network , Neural Oblivious Decision Ensembles

- Performs hyperparameter tuning using Grid Search for XGBoost.
- Evaluates models with ROC curves and confusion matrices.
- Logs training and evaluation metrics for each model.

Optimal Baseline results after CV were obtained on the following hyperparameters:

> ***Decision Tree***: criterion="gini", splitter="best", max_depth=None, min_samples_split=2, random_state=13
> ***SVM***: kernel="rbf", C=1.0, gamma="scale", class_weight=None
>
> **Naive Bayes**: model_type="gaussian"
> **k_NN**: n_neighbors=10, metric="manhattan"

Model	Training Accuracy	Mean CV Accuracy	Validation Accuracy	Total Execution time (seconds)
Decision Tree	1.0000	0.7615	0.7809	1.86
SVM	0.7906	0.7111	0.7002	227.49
k-NN	0.7490	0.6890	0.6892	4.08
Naive Bayes	0.6410	0.6363	0.6442	0.68

Decision Tree: Shows overfitting behavior, as evidenced by its high training accuracy compared to cross-validation and validation scores.

SVM: Demonstrates high bias, but despite using the RBF kernel, it fails to effectively capture the complexities of the training data. This suggests that SVM may not perform well on multi-class classification tasks. Attempts to use a one-vs-rest wrapper did not significantly improve the results and considerably increased execution time.

k-NN: Struggles with high-dimensional features, which impacts its performance.

Naive Bayes: This classifier was chosen due to observations of weak overall correlations in the data (as shown in the heatmap) and the near-normal distribution of the features, which align well with Naive Bayes assumptions. However, it surprisingly performed poorly in these conditions.

For more detailed evaluation metrics [***Precision , Recall , F1-score***]. Please refer to the execution_log of the data_ classification.py script

Optimal* Ensambles *results after CV were obtained on the following hyperparameters:

Random Forest: n_estimators=100, min_samples_split=2, max_depth=None, random_state=13, n_jobs=-1,

XGBoost: n_estimators=100, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8, random_state=13, eval_metric="mlogloss",

Model	Training Accuracy	Mean CV Accuracy	Validation Accuracy	Total Execution time (seconds)
Random Forest	1.0000	0.8214	0.8435	5.09
XGBoost	**0.9852**	**0.8793**	**0.8907**	**17.25**

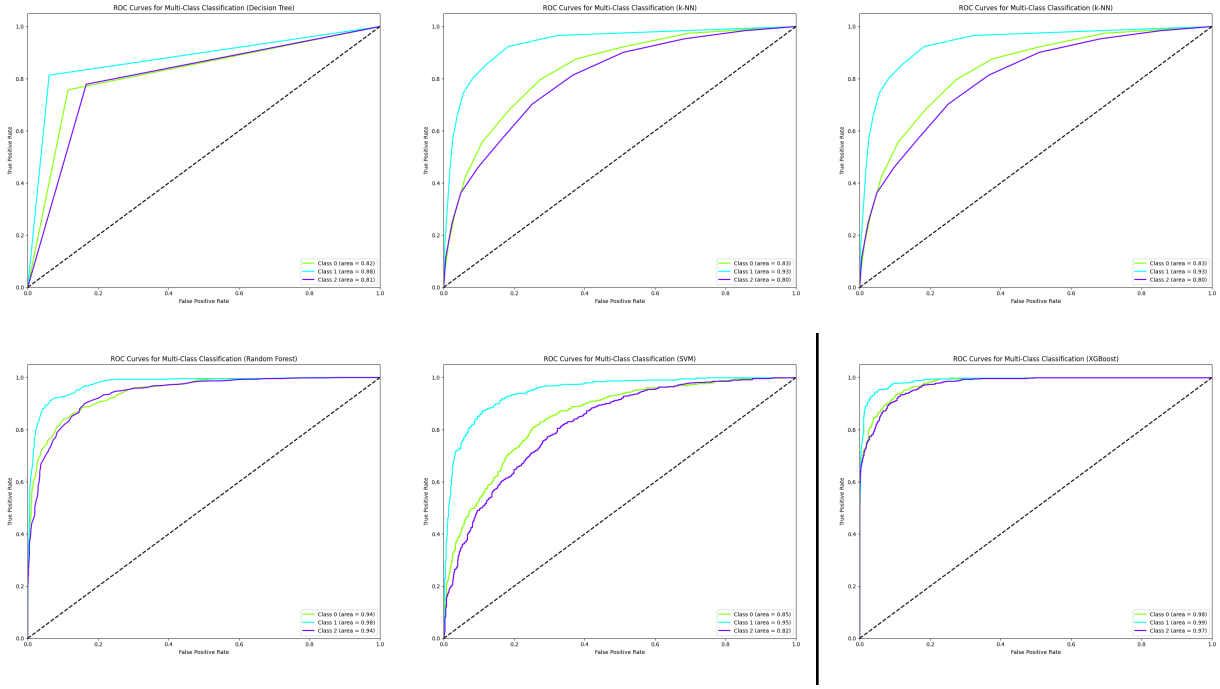
As demonstrated by the baseline results, tree-based models perform well in multi-class classification tasks. To improve validation accuracy, we leverage ***tree-based ensemble methods***. We begin with a ***Random Forest***, which, as a ***bagging technique, helps reduce variance***. This approach yields noticeable improvements in the validation score.
Next, we apply XGBoost, which brings a significant improvement in validation accuracy, at a slight reduction in training accuracy. This trade-off results in ***lower variance***. We relate these results to ***XGBoost's use of L1 and L2 regularization terms,*** which mitigate overfitting. Additionally, the incorporation of a learning rate term allows the model to take more conservative steps during training, further preventing overfitting. Moreover, ***XGBoost's gradient-based optimization*** enables the trees to construct ***complex decision boundaries,*** striking an ***effective balance between bias and variance***. [2]

For more detailed evaluation metrics [***Precision , Recall , F1-score***]. Please refer to the execution_log of the data_ classification.py script

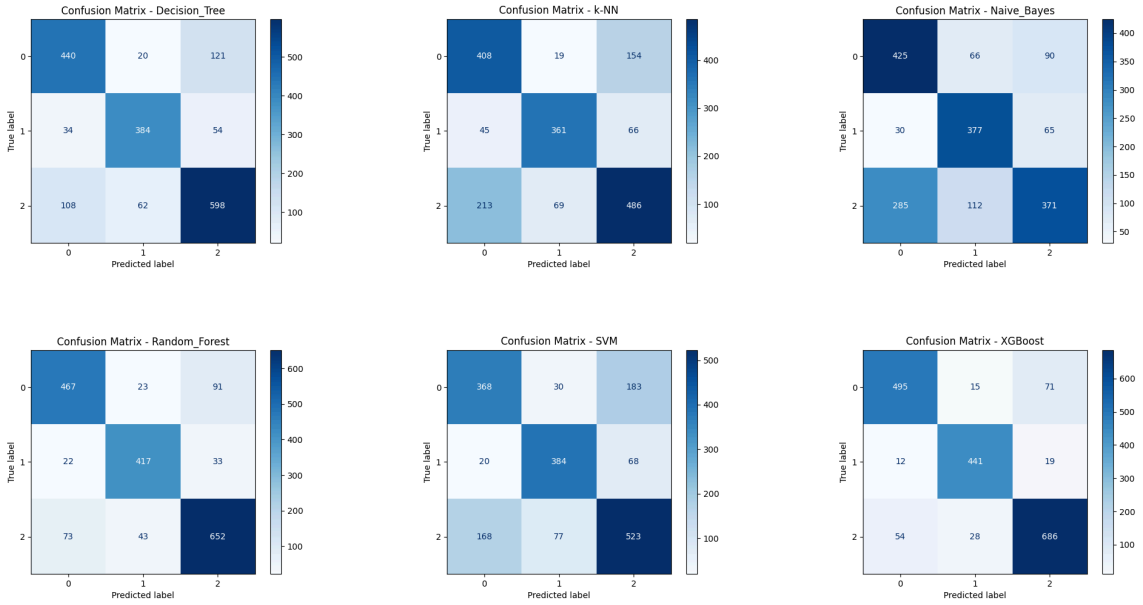
Grid search on XGBoost:

We finally try grid search with the following parameters grid

xgboost_grid_search_params = {'n_estimators': [50, 100], 'max_depth': [3, 4], 'learning_rate': [0.05, 0.1], 'subsample': [0.8], 'colsample_bytree': [0.8], 'gamma': [0], 'min_child_weight': [1, 3], 'reg_alpha': [0, 0.1], 'reg_lambda': [1, 2], 'num_class': [3], 'use_gpu': False, 'objective': ['multi:softprob'], 'eval_metric': ['mlogloss']}



Here, we present a one-vs-rest ROC curve for our selected model. Notably, Class 1, which was identified earlier as the dominant class, is classified the most accurately.

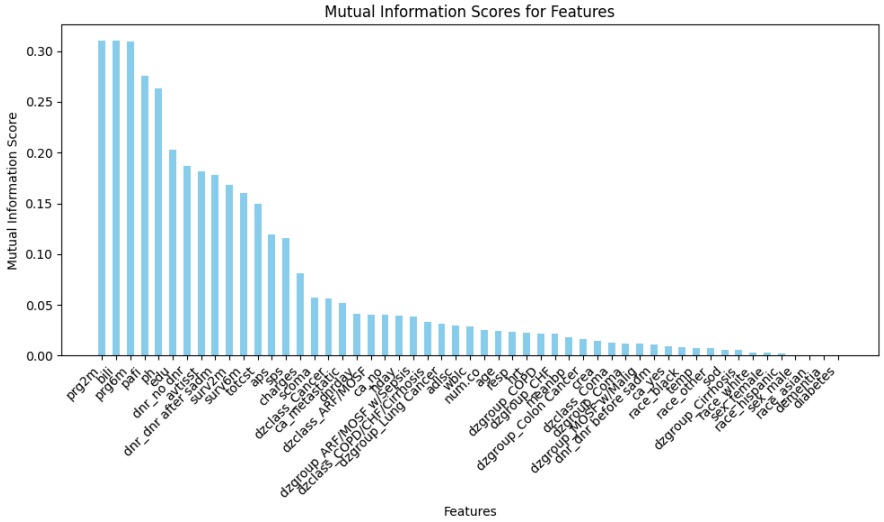


Model	Mean CV Accuracy	Validation Accuracy	Total Execution time (seconds)
XGBoost GS	**0.8828**	**0.8869**	**37.99**

After performing grid search, we observed no improvements; in fact, the validation accuracy decreased slightly. This confirms the first ***XGBoost remains the best-performing model in our analysis.***

Feature Elimination

We employed mutual information to identify the most informative features relative to our target labels, selecting the top N features for further analysis. Comparing classifier performance, we observed that models trained with the reduced feature set generally showed lower validation accuracy. Consequently, we opted to keep all original features in our models.



Bonus XGBoost is all you need ?

This section compares the performance of XGBoost against two neural network-based approaches:

1. **FeedForward Neural Network (FFNN)**
2. **Neural Oblivious Decision Ensembles (NODE)**

Inspired by [3], we examine the performance of neural networks on multi-class classification tasks for tabular data.We begin with a simple feedforward neural network, incorporating categorical embeddings and optimizing with Cross-Entropy loss. Additionally, we evaluate the method proposed in [4], driven by the authors' claim that it outperforms XGBoost in most cases. The findings of our analysis are as follows:

Model	Validation Accuracy	Traning Accuracy	Total Execution time (seconds)
XGBoost	**0.8869**	**0.9852**	**37.99s**
FFNN	**0.7**	**0.770**	**9s on T4**
NODE	**0.75**	**0.78**	**120s on T4**

Despite the claims in [4], XGBoost remains the best-performing method for our classification task. Generally, deep learning methods, while excelling in areas such as latent feature extraction, generative modeling, and self-supervised learning, still fall short when it comes to tasks involving tabular data.

However, NODE [4] offers an end-to-end differentiable approach for training tree-based ensembles, enabling seamless integration with other deep learning architectures, such as convolutional or attention-based models. For instance, a convolutional neural network (ConvNet) block could be combined with NODE, allowing the model to make decisions based on both tabular and image data simultaneously.

Deliverables

- **Logs**: Tracks preprocessing, modeling, and evaluation steps.
- **Plots**: Provides visual insights into data, clusters, and classification results.
- **Processed Dataset**: A cleaned and fully prepared dataset for further analysis or deployment.
- **Reports**: Includes Silhouette scores, training accuracy, and validation performance.

Use Cases

- Exploratory analysis of healthcare datasets like SUPPORT2.
- Identifying patterns, outliers, and clusters in high-dimensional data.
- Training and comparing machine learning models for predictive analysis.

Conclusion

Despite their high dimensionality, medical record features generally exhibit weak correlations, as demonstrated in our heatmap. However, our mutual information histogram reveals that the physician survival estimate (**prg2m**) has significant predictive power. While this is notable, including such features could introduce substantial bias into the dataset and hinder the classifier's ability to generalize to out-of-distribution data. As the quality and experience of medical staff that differ across regions.

Another observation is that whether a patient signs a Do Not Resuscitate (DNR) order is a strong predictor of survival rates. This might reflect the psychological aspects influencing a patient's treatment journey . Additionally, **Total Bilirubin (Tbil)** emerged as a critical feature for predicting patient survival rates, highlighting its relevance in clinical prognostic models.

From a methodological perspective, Support Vector Machines (SVMs) are not well-suited for multi-class classification tasks, while **XGBoost** proved to be highly effective. In fact, XGBoost outperformed **state-of-the-art tabular neural network methods**

A notable limitation of our dataset is its age and geographic specificity. **Collected between 1989 and 1994**, primarily from **U.S. medical centers**, the data may reflect biases, especially in variables like physician estimates, due to evolving medical practices and regional differences. Future research should apply similar data mining techniques to more recent and diverse datasets to enhance the generalizability and applicability of the findings across various healthcare settings. Alternatively, **employing transfer learning methods** could adapt models trained on this dataset to more recent, **out-of-distribution data.**

Contributions

Arvin Bayat Manesh :

- Data preprocessing
- Data Clustering (K-means) + The Elbow Method + Silhouette Score Analysis + Manual Grid Search for DBSCAN
- Outlier Detection (Isolation Forest)
- Data Classification (Decision Trees, Support Vector Machines, Random Forests)
- Main script of Exploratory Data Analysis (EDA)
- Main script of clustering,
- Main script of outlier detection.
- README file
- ½ Report
- ½ Poster

Amr Sharafeldin :

- Data visualization utils
- Logger utils
- Data Clustering (DBSCAN)
- Outlier Detection (Elliptic Envelope)
- Data Classification (XGBOOST, Naive Bayes , k_NN , XGBOOST grid search)
- Mutual information
- Main script of Data Classification.
- Main script of Feature Elimination
- Data Classification execution instruction on the README file
- Bonus part - XGBoost vs FFNN and NODE
- ½ Report.
- ½ Poster

References

---

[1] Knaus, W., Connors, A. F., Jr., Dawson, N., Desbiens, N. A., Fulkerson, W. J., Goldman, L., Lynn, J., Oye, R., Bergner, M., Damiano, A., Hakim, R., Murphy, D. J., Teno, J., Virnig, B., Wagner, D. P., Wu, A., Yasui, Y., Robinson, D. K., Kreling, B., & Ransohoff, D. (1995). A *controlled trial to improve care for seriously ill hospitalized patients: The study to understand prognoses and preferences for outcomes and risks of treatments (SUPPORT). JAMA, 274(20), 1591–1598.*

[2] A guide on XGBoost hyperparameters tuning [Notebook]. Kaggle. *https://www.kaggle.com/code/prashant111/a-guide-on-xgboost-hyperparameters-tuning*

[3]Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on tabular data? Advances in Neural Information Processing Systems, 35, 1–14

Popov, S., Morozov, S., & Babenko, A. (2019). Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. arXiv preprint arXiv:1909.06312. Retrieved from https://arxiv.org/abs/1909.06312

[5] Manu Joseph, PyTorch Tabular: A Framework for Deep Learning with Tabular Data.