DATA SET SUPPORT II

THE SUPPORT (STUDY TO UNDERSTAND PROGNOSES AND PREFERENCES FOR OUTCOMES AND RISKS OF TREATMENTS) DATASET IS AN EXTENSIVE COLLECTION OF DATA FROM CRITICALLY ILL PATIENTS. IT ENCOMPASSES RECORDS FROM 9105 PATIENTS ACROSS FIVE MEDICAL CENTERS IN THE UNITED STATES, COVERING TWO STUDY PERIODS: 1989-1991 AND 1992-1994. THIS DATASET IS UNIQUE IN ITS FOCUS ON NINE CRITICAL DISEASE CATEGORIES, INCLUDING ACUTE RESPIRATORY FAILURE, VARIOUS CANCERS, AND MULTIPLE ORGAN SYSTEM FAILURES.

https://archive.ics.uci.edu/dataset/880/support2

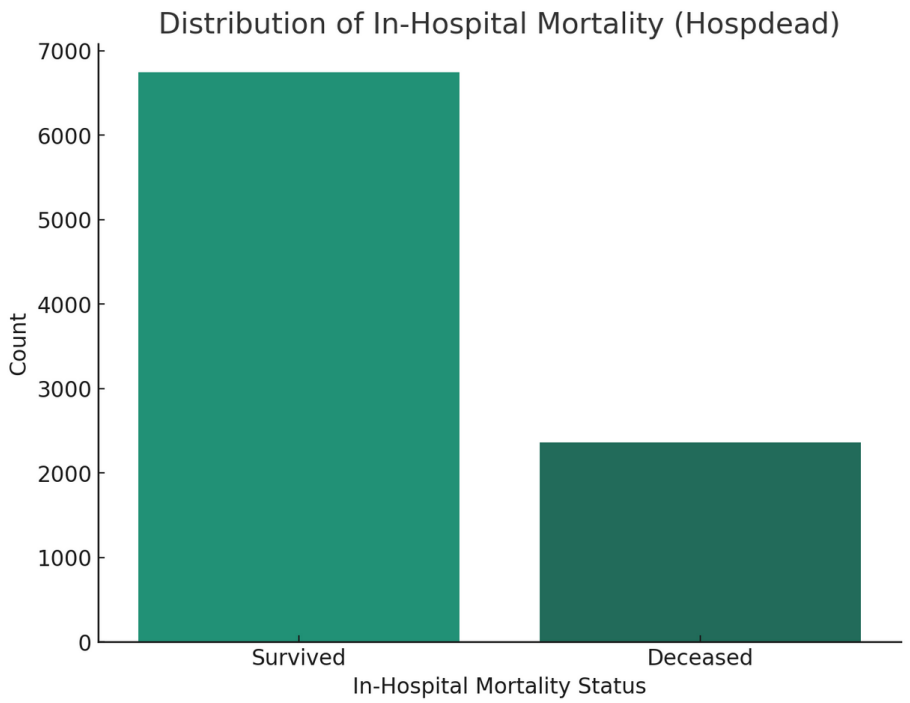
DATA SET STRUCTURE

```
print("Shape ofthe data", df.shape)

df.head()
```

✓ 0.0s Python

```
Shape ofthe data (9105, 47)
```

	age	death	sex	hospdead	slos	d.time	dzgroup	dzclass	num.co	edu	income	scoma	charges	totcst	totmcst	avtisst	race	sps	aps	surv2m	
1	62.84998	0	male	0	5	2029	Lung Cancer	Cancer	0	11.0	$11-$25k	0.0	9715.0	NaN	NaN	7.000000	other	33.898438	20.0	0.262939	0.(
2	60.33899	1	female	1	4	4	Cirrhosis	COPD/CHF/Cirrhosis	2	12.0	$11-$25k	44.0	34496.0	NaN	NaN	29.000000	white	52.695312	74.0	0.001000	0.(
3	52.74698	1	female	0	17	47	Cirrhosis	COPD/CHF/Cirrhosis	2	12.0	under $11k	0.0	41094.0	NaN	NaN	13.000000	white	20.500000	45.0	0.790894	0.
4	42.38498	1	female	0	3	133	Lung Cancer	Cancer	2	11.0	under $11k	0.0	3075.0	NaN	NaN	7.000000	white	20.097656	19.0	0.698975	0
5	79.88495	0	female	0	16	2029	ARF/MOSF w/Sepsis	ARF/MOSF	1	NaN	NaN	26.0	50127.0	NaN	NaN	18.666656	white	23.500000	30.0	0.634888	0.!



PREDICTING IN-HOSPITAL MORTALITY - A BINARY CLASSIFICATION CHALLENGE

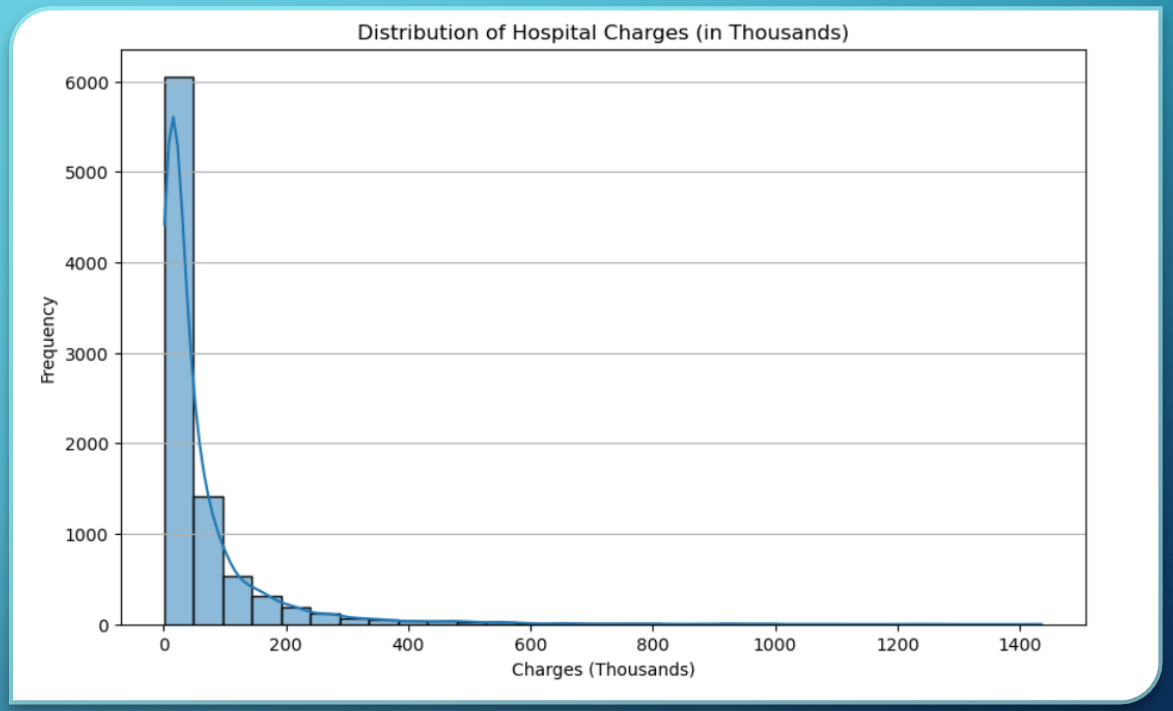
- The primary objective of our classification analysis is to predict in-hospital mortality, a critical outcome in healthcare settings. This binary classification problem focuses on determining whether a critically ill patient will survive (0) or succumb during their hospital stay (1).

IMPORTANCE

The ability to predict mortality can significantly aid in clinical decision-making, resource allocation, and patient-family counseling, particularly in managing end-of-life care scenarios.

FORECASTING TOTAL HOSPITAL COSTS - A REGRESSION TASK

- The regression aspect of our analysis aims to predict the total hospital costs incurred per patient. This is a continuous variable prediction, requiring the model to estimate the monetary cost associated with a patient's hospital stay. The challenge lies in correlating various factors like treatment procedures, length of stay, severity of illness, and patient demographics with the total cost.



IMPORTANCE

- **Financial Planning**: Accurate cost predictions can be instrumental for hospitals and patients in financial planning and budgeting.
- **Resource Optimization**: Understanding cost drivers enables better resource management and allocation within healthcare facilities.
- **Policy Making**: Insights from cost predictions can inform healthcare policy, particularly in areas related to insurance and patient care funding.

DATA PREPROCESSING: MISSING VALUES

```
nulls =df.columns[df.isnull().any()]
nulls
```

✓ 0.0s

```
Index(['edu', 'scoma', 'charges', 'totcst', 'totmcst', 'avtisst', 'sps', 'aps',
       'surv2m', 'surv6m', 'prg2m', 'prg6m', 'dnrday', 'meanbp', 'wblc', 'hrt',
       'resp', 'temp', 'pafi', 'alb', 'bili', 'crea', 'sod', 'ph', 'glucose',
       'bun', 'urine', 'adlp', 'adls'],
      dtype='object')
```

29 columns out of 47 had missing values (61%)

Depending on how many missing values has been in columns – they were either substituted for mean, empty rows were dropped or the whole column was dropped as feature.

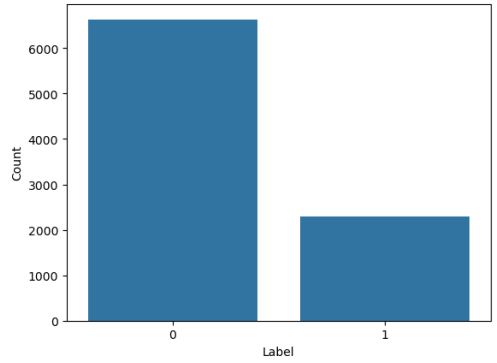
Column Name	Null Count	Column Name	Null Count	Column Name	Null Count	Column Name	Null Count	Column Name	Null Count	Column Name	Null Count
edu	1634	scoma	1	charges	172	totcst	888	totmcst	3475	avtisst	82
sps	1	aps	1	surv2m	1	surv6m	1	prg2m	1649	prg6m	1633
dnrday	30	meanbp	1	wblc	212	hrt	1	resp	1	temp	1
pafi	2325	alb	3372	bili	2601	crea	67	sod	1	ph	2284
glucose	4500	bun	4352	urine	4862	adlp	5641	adls	2867	None	None

DATA PREPROCESSING: CATEGORICAL VALUES

sex	dzgroup	dzclass	income	race	ca	dnr	sfdm2
male	Lung Cancer	Cancer	$11-$25k	other	metastatic	no dnr	NaN
female	Cirrhosis	COPD/CHF/Cirrhosis	$11-$25k	white	no	NaN	<2 mo. follow-up
female	Cirrhosis	COPD/CHF/Cirrhosis	under $11k	white	no	no dnr	<2 mo. follow-up
female	Lung Cancer	Cancer	under $11k	white	metastatic	no dnr	no(M2 and SIP pres)
female	ARF/MOSF w/Sepsis	ARF/MOSF	NaN	white	no	no dnr	no(M2 and SIP pres)
...	...	...	...	...	...	...	...
male	ARF/MOSF w/Sepsis	ARF/MOSF	NaN	white	no	no dnr	NaN
female	Coma	Coma	NaN	white	no	no dnr	NaN
male	ARF/MOSF w/Sepsis	ARF/MOSF	NaN	white	no	no dnr	NaN
male	MOSF w/Malig	ARF/MOSF	NaN	white	yes	dnr after sadm	<2 mo. follow-up
female	ARF/MOSF w/Sepsis	ARF/MOSF	$11-$25k	white	no	no dnr	no(M2 and SIP pres)

DATA PREPROCESSING: CLASS IMBALANCE

```
label
0    6633
1    2299
Name: count, dtype: int64
```




```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE


# Separate input features and target
X = df.drop('label', axis=1)
y = df['label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Summarize class distribution
print("Before SMOTE '1': {}".format(sum(y_train == 1)))
print("Before SMOTE '0': {} \n".format(sum(y_train == 0)))

# Apply SMOTE
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

print("After SMOTE '1': {}".format(sum(y_train_smote == 1)))
print("After SMOTE '0': {}".format(sum(y_train_smote == 0)))
```

✓ 0.0s

```
Before SMOTE '1': 1593
Before SMOTE '0': 4659

After SMOTE '1': 4659
After SMOTE '0': 4659
```

- The Synthetic Minority Oversampling Technique (SMOTE) is used to over-sample the minority class in a dataset and make the class distribution balanced. This technique generates synthetic samples rather than creating copies of existing samples.

CLASSIFICATION FEATURES

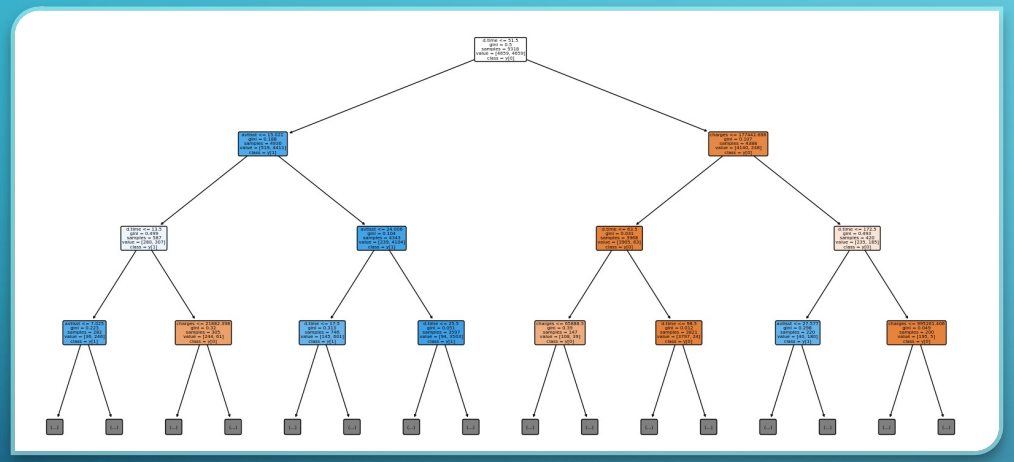
d.time	adlsc	avtisst	dnr	pafi	charges	crea	dzgroup_Coma	hday	aps
1740	6.0	7.000000	0	388.00000	9715.0	1.199951	0	1	20.0
5	1.0	29.000000	0	100.00000	34496.0	5.299805	0	3	74.0
47	0.0	13.000000	0	231.65625	41094.0	2.000000	0	4	45.0
133	0.0	7.000000	0	333.30000	3075.0	0.799927	0	1	19.0
1740	2.0	18.666656	0	173.31250	50127.0	0.799927	0	3	30.0

Set #1 Correlation & RT

dnr	avtisst	d.time	aps	hday	dzgroup_Coma	dzgroup_CHF	charges	dzgroup_MOSF w/Malig	bili	dzgroup_ARF/MOSF w/Sepsis	pafi	adlsc	crea	dzgroup_COPD
0	7.000000	1740	20.0	1	0	0	9715.0	0	0.299988	0	388.00000	6.0	1.199951	0
0	29.000000	5	74.0	3	0	0	34496.0	0	1.010000	0	100.00000	1.0	5.299805	0
0	13.000000	47	45.0	4	0	0	41094.0	0	2.199707	0	231.65625	0.0	2.000000	0
0	7.000000	133	19.0	1	0	0	3075.0	0	1.010000	0	333.30000	0.0	0.799927	0
0	18.666656	1740	30.0	3	0	0	50127.0	0	1.010000	1	173.31250	2.0	0.799927	0

Set #2 Correlation only

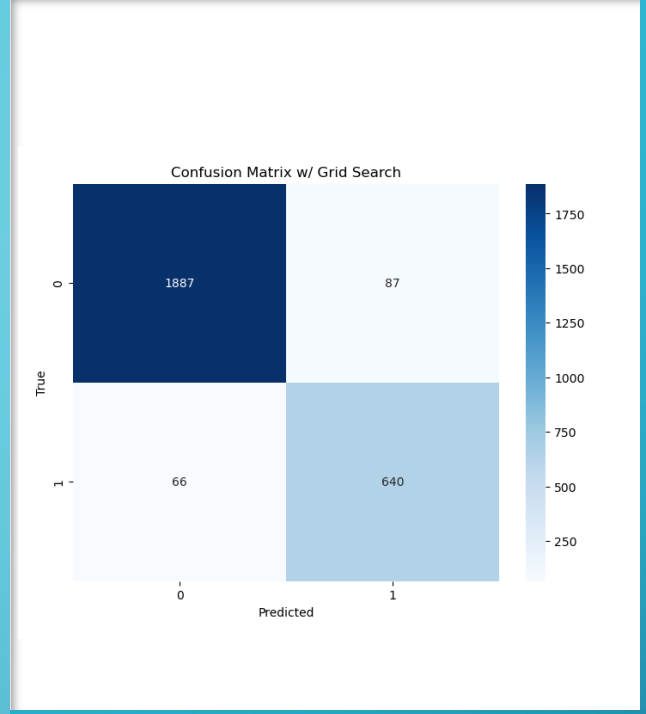
DECISION TREE

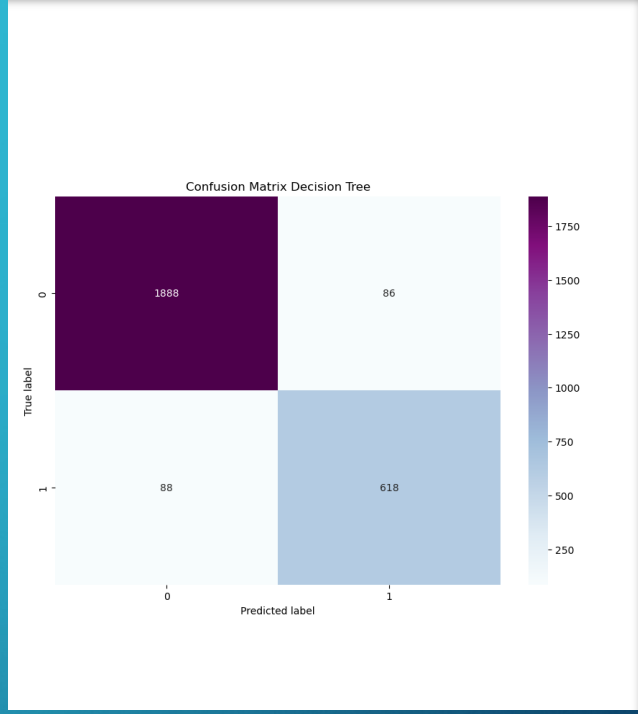


- Decision trees are highly interpretable models.
- Decision trees can capture non-linear relationships between features and the target variable.
- No assumptions about data distribution. This is beneficial when working with real-world medical data, which may not follow specific theoretical distributions.

DECISION TREE RESULTS

- In both models, the accuracy is similar at approximately 94%.
- The precision, recall, and f1-score for predicting class '1' show improvement in the grid-search optimized model. Specifically, the recall for class '1' increased from 0.88 to 0.91, indicating that the optimized model is better at identifying true positive cases of this class.
- The macro and weighted averages across precision, recall, and f1-score are slightly better in the grid-search optimized model, suggesting a more balanced performance across both classes.
- Both models show a high number of true positives and true negatives. However, the grid-search optimized model has fewer false negatives (66 compared to 88) for class '1', which is critical in scenarios like predicting hospital deaths where missing a positive case can be more severe than a false alarm.





Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	1974
1	0.88	0.91	0.89	706
accuracy			0.94	2680
macro avg	0.92	0.93	0.93	2680
weighted avg	0.94	0.94	0.94	2680

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	1974
1	0.88	0.88	0.88	706
accuracy			0.94	2680
macro avg	0.92	0.92	0.92	2680
weighted avg	0.94	0.94	0.94	2680

LOGISTIC REGRESSION

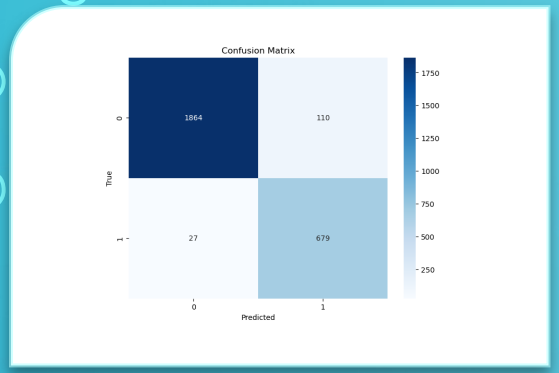
- Great for binary classification
- Logistic regression models offer good interpretability, allowing for an understanding of how each feature influences the likelihood of the outcome. This can be particularly valuable in a healthcare context, where understanding the risk factors associated with hospital mortality is crucial.
- Dataset includes a wide range of features, both numerical (like 'age', 'scoma', 'charges') and binary/categorical (such as 'sex', 'diabetes'). Logistic regression can handle this mix of variable types effectively

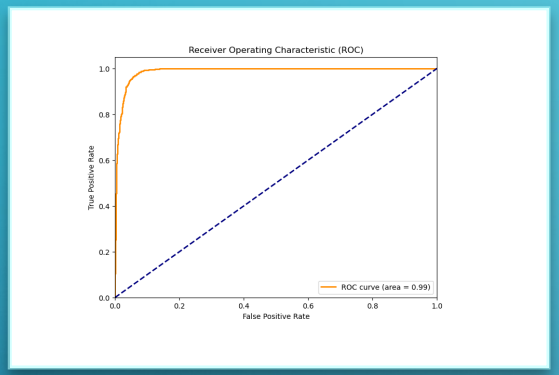
```python
# Separate input features and target
X = df[final_features]
y = df['label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Apply SMOTE to balance the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the scaled training data (after SMOTE) and transform the testing data
X_train_scaled = scaler.fit_transform(X_train_smote)
X_test_scaled = scaler.transform(X_test)

# Initialize the Logistic Regression model
log_reg = LogisticRegression(random_state=42)

# Fit the model on the scaled and balanced training data
log_reg.fit(X_train_scaled, y_train_smote)

# Making predictions on the test set
y_pred = log_reg.predict(X_test_scaled)

# Evaluating the model
# Printing the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Printing the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Printing the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

(variable) y_train: Unknown

✓ 0.1s

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.93      0.96      1974
           1       0.83      0.95      0.89       706

    accuracy                           0.94      2680
   macro avg       0.91      0.94      0.92      2680
weighted avg       0.94      0.94      0.94      2680

Confusion Matrix:
[[1839  135]
 [  38  668]]
Accuracy: 0.9354477611940298
```



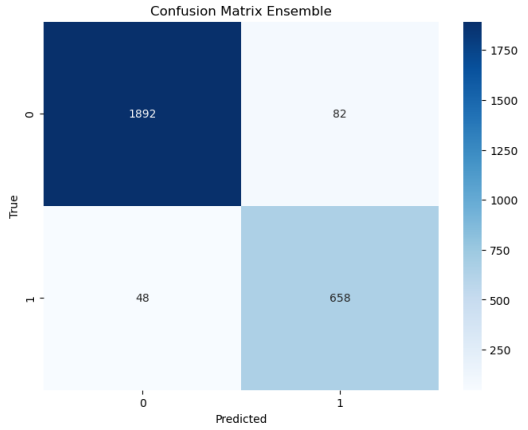



```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      1974
           1       0.86      0.96      0.91       706

    accuracy                           0.95      2680
   macro avg       0.92      0.95      0.94      2680
weighted avg       0.95      0.95      0.95      2680

Confusion Matrix:
[[1864  110]
 [  27  679]]
Accuracy: 0.9488805970149253
```

LOGISTIC REGRESSION RESULTS

- The optimized logistic regression model displays a high accuracy of 95% in predicting hospital mortality. It is exceptionally precise in identifying patients at low risk (precision: 99%), while also highly sensitive in detecting high-risk cases (recall: 96%). The balance between precision and recall is reflected in the strong F1-scores of 0.96 and 0.91 for the negative and positive classes, respectively. The model shows a tendency to predict mortality more conservatively, preferring to err on the side of caution with a slightly higher false positive rate. Overall, the ROC curve demonstrates an excellent AUC of 0.99, indicating superior discriminative ability. This model is highly effective for clinical decision-making, prioritizing the identification of patients at risk of hospital death.

ENSEMBLE MODEL OF DT, LOGISTIC REGRESSION AND RANDOM FOREST




```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1974
           1       0.89      0.93      0.91       706

    accuracy                           0.95      2680
   macro avg       0.93      0.95      0.94      2680
weighted avg       0.95      0.95      0.95      2680

Confusion Matrix:
[[1892   82]
 [  48  658]]
Accuracy: 0.9514925373134329
```

- An ensemble model harnesses the collective strength of multiple machine learning algorithms to improve predictive performance. By combining the decisions from distinct models—here, a Decision Tree, Logistic Regression, and Random Forest—the ensemble reduces the likelihood of errors that any single model might make. Our soft voting scheme, which considers the predicted probabilities from each model, ensures that the final prediction reflects a consensus with due weight to the confidence of each model's prediction.
- The ensemble's efficacy is evident from its accuracy of 95.1%, showcasing its robustness in making correct predictions. The model achieves a commendable balance between precision and recall, affirming its capacity to accurately identify patients at risk while minimizing false alerts. The ROC curve, yielding an AUC of 0.95, substantiates the ensemble's excellent ability to differentiate between the patient outcomes. This confirms the ensemble's suitability for deployment in critical healthcare settings, providing a dependable tool for medical professionals to prioritize care and interventions.

COMPARISON OF CLASSIFICATION MODELS

Model	Precision	Recall	F1-Score	Accuracy	AUC
Decision Tree (Grid)	0.88	0.91	0.89	0.94	0.93
Logistic Regression (Grid)	0.86	0.96	0.91	0.95	0.99
Ensemble	0.89	0.93	0.91	0.95	0.95

- In analyzing the performance metrics of the three predictive models, we observe that both the Logistic Regression with Grid Search and the Ensemble model exhibit superior accuracy and AUC scores, indicating their strong discriminative ability in predicting hospital mortality. The Logistic Regression stands out with the highest recall and AUC, suggesting it is particularly effective in identifying true positive cases without significantly increasing false positives. The Ensemble model, while closely matching the Logistic Regression in accuracy and F1-score, offers a balance between precision and recall.
- The Logistic Regression with Grid Search emerges as the most robust model for this application, with its exceptional ability to identify patients at risk accurately. Its high recall is crucial in a medical context, ensuring that fewer cases at risk go unnoticed. The high AUC further reinforces its status as the best model, providing confidence in its use as a reliable tool for hospital mortality prediction.

REGRESSION FEATURES

Set #1 Intersection between correlating features & random forest choice

sps	bun	scoma	hday	hrt	avtisst	bili	aps	slos	label
33.898438	23.0	0.0	1	69.0	7.000000	0.299988	20.0	5	9715.0
42.500000	23.0	44.0	3	112.0	29.000000	1.010000	74.0	4	34496.0
20.500000	23.0	0.0	4	88.0	13.000000	2.199707	45.0	17	41094.0
20.097656	23.0	0.0	1	88.0	7.000000	1.010000	19.0	4	3075.0
23.500000	23.0	26.0	3	112.0	18.666656	1.010000	30.0	16	50127.0

Set # 2 Only correlation top features

	slos	hday	avtisst	dzgroup_ARF/MOSF w/Sepsis	aps	sps	bili	hospdead	scoma	hrt	temp	ca_no	crea	edu	bun	label
0	5	1	7.000000	0	20.0	33.898438	0.299988	0	0.0	69.0	36.00000	0	1.199951	11.000000	23.0	9715.0
1	4	3	29.000000	0	74.0	42.500000	1.010000	1	44.0	112.0	35.50000	1	5.299805	12.000000	23.0	34496.0
2	17	4	13.000000	0	45.0	20.500000	2.199707	0	0.0	88.0	37.39844	1	2.000000	12.000000	23.0	41094.0
3	4	1	7.000000	0	19.0	20.097656	1.010000	0	0.0	88.0	35.50000	0	0.799927	11.000000	23.0	3075.0
4	16	3	18.666656	1	30.0	23.500000	1.010000	0	26.0	112.0	37.89844	1	0.799927	11.744707	23.0	50127.0

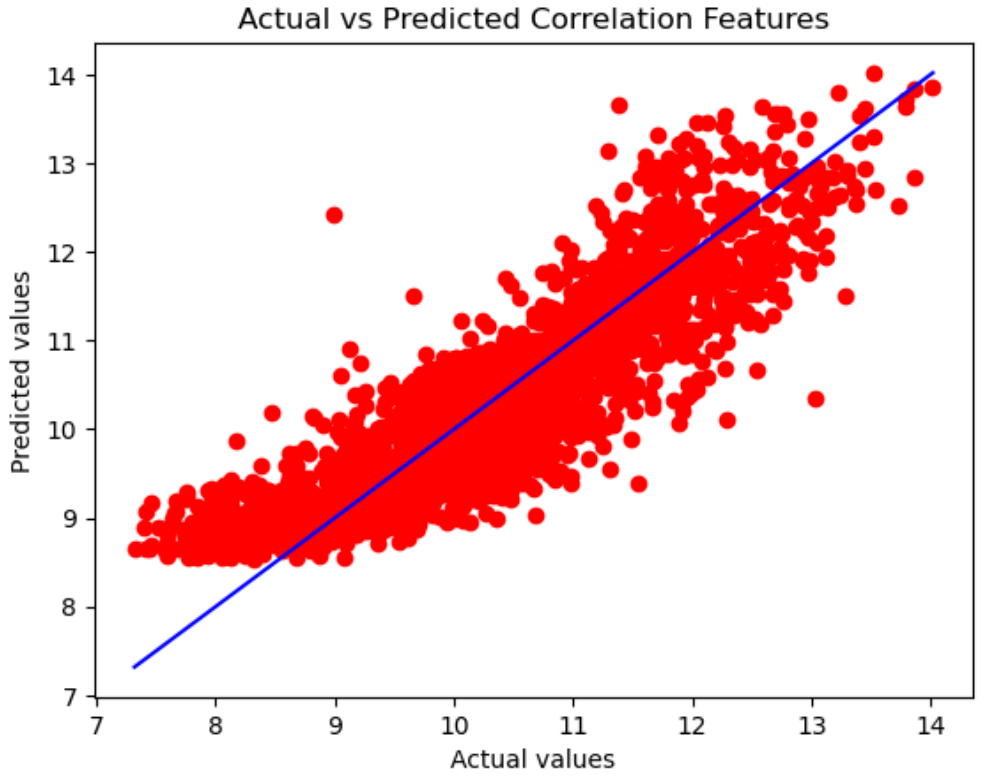
LINEAR REGRESSION

```
"""Linear regression with Intersect Features"""
X = df.drop(['label'], axis=1)  # Drop non-feature columns
y = df['label']  # Use the log-transformed label



# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Fit the scaler to the training data and transform both training and test data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Fit the Linear Regression model
reg = LinearRegression()
reg.fit(X_train_scaled, y_train)

# Predictions can be made with the model using the scaled test data
y_pred_reg = reg.predict(X_test_scaled)

# Scatter plot for actual vs predicted values
plt.scatter(y_test, y_pred_reg, color='red')
plt.title('Actual vs Predicted Intersectioned Features')
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='blue')  # A reference line
plt.show()


# The coefficients
print('Coefficients:\n', reg.coef_)

# The mean squared error
print('Mean squared error: %.2f' % mean_squared_error(y_test, y_pred_reg))

# The mean absolute error
print('Mean absolute error: %.2f' % mean_absolute_error(y_test, y_pred_reg))

# The root mean squared error
print('Root mean squared error: %.2f' % np.sqrt(mean_squared_error(y_test, y_pred_reg)))

# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f' % r2_score(y_test, y_pred_reg))
```

- Captures Linear Relationship: several features in the dataset, such as length of stay and severity of illness, likely have a linear relationship with medical charges, making linear regression a suitable model for capturing these relationships.
- Linear regression provides clear interpretability, allowing healthcare professionals to understand how different factors, like age or medical procedures, are associated with the total charges, which is invaluable for cost analysis and management.
- It is computationally efficient and straightforward to implement, making it an excellent starting point for modeling and understanding the underlying structure of the data before considering more complex models.
- As charges are a continuous variable, linear regression is well-suited for making quantitative predictions, which can be directly used in financial planning and budgeting within the healthcare system.

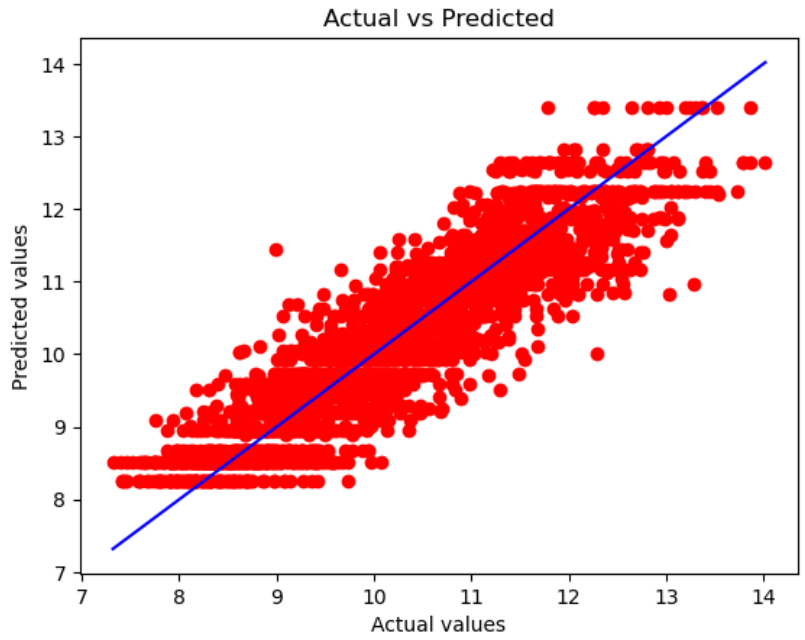
LINEAR REGRESSION RESULTS

- variable, with a coefficient of determination (R^2) of 0.79. This suggests that approximately 79% of the variability in the target can be explained by the model's inputs. The model's coefficients indicate the relative influence of each feature on the target variable, with the first feature having the most substantial positive impact on the prediction.
- However, the mean squared error (MSE) and root mean squared error (RMSE) values are 0.34 and 0.58, respectively, which points to the average squared difference and the standard deviation of the prediction errors. The mean absolute error (MAE) of 0.45 reflects the average absolute difference between predicted values and actual values. These error metrics suggest that the model's predictions are reasonably close to the true values but could potentially be improved with further model tuning or feature selection.




```
Coefficients:
 [ 2.28695621  1.04044264  1.29970034  0.27352079  0.0946163   0.18694846
  0.20147505  0.01502122  0.07991532  0.02205944  0.12074003  0.05029923
 -0.07321796  0.41821698  0.21061349]

Mean squared error (MSE): 0.34
Mean absolute error (MAE): 0.45
Root mean squared error (RMSE): 0.58
Coefficient of determination (R^2): 0.79
```

REGRESSION DECISION TREE RESULTS

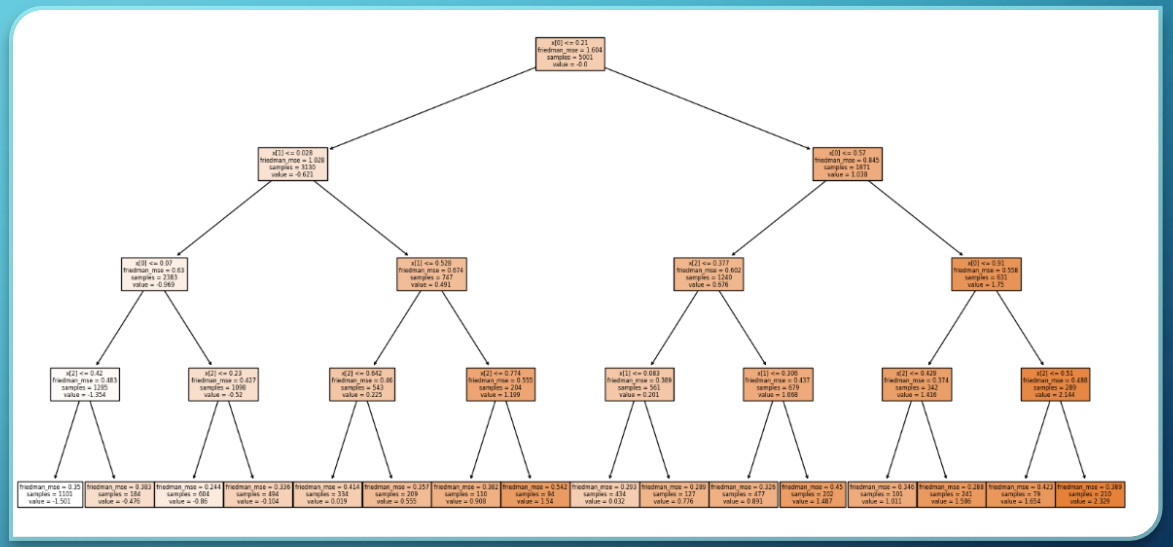



```
Coefficients:
 [ 2.28695621  1.04044264  1.29970034  0.27352079  0.0946163   0.18694846
  0.20147505  0.01502122  0.07991532  0.02205944  0.12074003  0.05029923
 -0.07321796  0.41821698  0.21061349]

Mean squared error (MSE): 0.30
Mean absolute error (MAE): 0.43
Root mean squared error (RMSE): 0.55
Coefficient of determination (R^2): 0.82
```

- The Decision Tree Regressor exhibits promising results with a coefficient of determination (R^2) of 0.82, indicating that it successfully captures 82% of the variance in the target variable.
- The model's predictive accuracy is further evidenced by relatively low error metrics: a mean squared error (MSE) of 0.30 and a root mean squared error (RMSE) of 0.55, both of which imply that the model's predictions are fairly close to the actual values.
- The mean absolute error (MAE) of 0.43 reinforces this, suggesting that, on average, the model's predictions deviate from the actual values by a moderate margin. These metrics collectively signify that the Decision Tree Regressor is a reliable and effective model for our data.

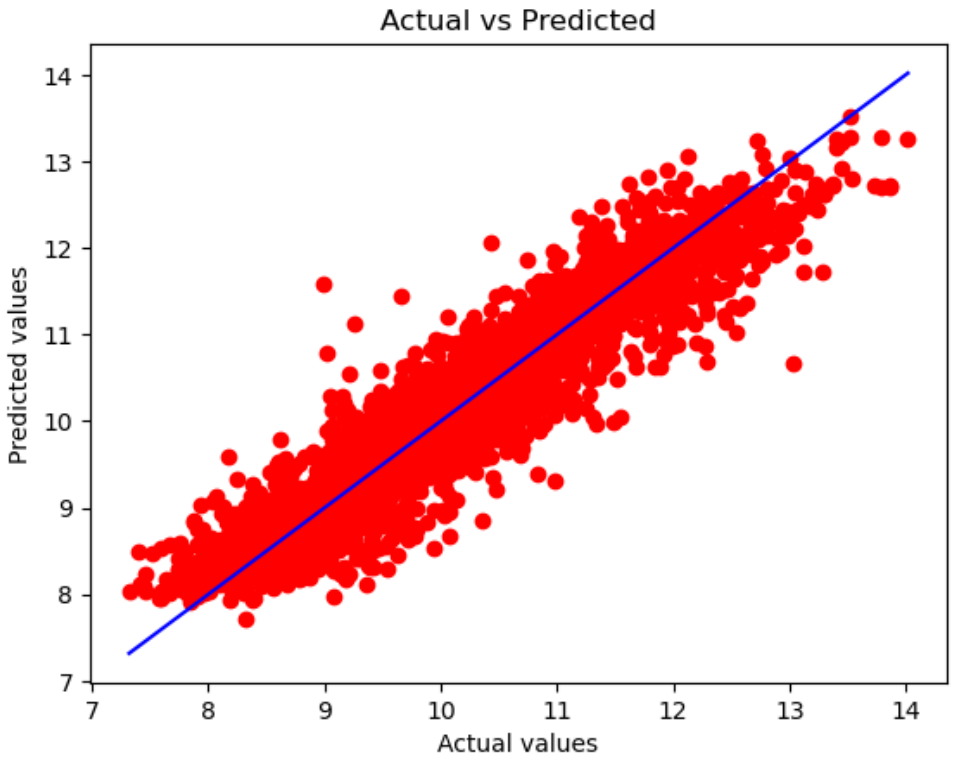
GRADIENT BOOSTING REGRESSOR WITH PARAMETRS FROM GRIDSEARCHCV

- Gradient Boosting Regressor is well-suited for our dataset as it builds upon an ensemble of weak prediction models, typically decision trees, to iteratively correct errors made by prior models, leading to a more robust and accurate overall model. Its ability to handle complex, non-linear relationships in the data can capture deeper insights and improve predictions beyond what simpler models like linear regression can achieve. Furthermore, its flexibility with various hyperparameters, including tree characteristics and learning rate, allows for fine-tuning to closely fit our specific dataset, effectively reducing overfitting and enhancing prediction accuracy.



GDBR W/ GRID SEARCH CV RESULTS

- The Mean Squared Error (MSE) is 0.21, suggesting that the model's predictions are generally close to the actual values, with a low average squared deviation.
- The Root Mean Squared Error (RMSE) of 0.46 further indicates a good fit, as it represents the standard deviation of the residuals (prediction errors) and is relatively low.
- The Mean Absolute Error (MAE) of 0.35 reinforces the model's accuracy, showing that, on average, the absolute difference between predicted and actual values is small.
- An R^2 value of 0.87 is particularly noteworthy, implying that the model explains 87% of the variance in the target variable. This high value indicates a strong model that effectively captures the underlying patterns in the data.
- The coefficients of the model, ranging from negative values to values above 2, reflect the varying degrees of influence each feature has on the target variable. Positive coefficients indicate a direct relationship, while negative coefficients indicate an inverse relationship.
- In summary, the Gradient Boosting Regressor demonstrates robust performance with high predictive accuracy and a strong ability to explain the variability in your data. The low error values coupled with a high R^2 score indicate that this model is well-suited for predicting the target variable in your dataset.




```
Coefficients:
 [ 2.28695621  1.04044264  1.29970034  0.27352079  0.0946163   0.18694846
  0.20147505  0.01502122  0.07991532  0.02205944  0.12074003  0.05029923
 -0.07321796  0.41821698  0.21061349]

Mean squared error (MSE): 0.21
Mean absolute error (MAE): 0.35
Root mean squared error (RMSE): 0.46
Coefficient of determination (R^2): 0.87
```

REGRESSION MODEL COMPARISON

Model	MSE	MAE	RMSE	R2
Linear Regression	0.34	0.45	0.58	0.79
Decision Tree	0.30	0.43	0.55	0.82
Gradient Boosting	0.21	0.35	0.46	0.87

The Gradient Boosting model outperforms both the Linear Regression and Decision Tree models across all metrics, indicating it is the best model for this dataset. It has the lowest MSE, MAE, and RMSE, suggesting more accurate and consistent predictions. Additionally, its R^2 value is the highest, indicating the model explains a larger proportion of variance in the data. These results suggest that Gradient Boosting's complex ensemble approach is more effective for this particular dataset than the simpler linear and tree-based models.

QUESTIONS?