Unlocking the Secrets of Heart Transplant Success with Machine Learning

Trevor Leach

Overall Costs

Waiting List

Potential for Failure

Overall Costs

30 Days of Pre-Transplant Medical Care

$49,800

Hospital Admission for Procedure

$1,062,600

180 Days of Post-Transplant Medical Care

$270,300

Grand Total

$1,664,800

Organ Procurement

$131,500

Physician Cost

$111,100

Medications

$39,500

Waiting List

Potential for Failure

Waiting List

Overall Costs

New Candidates Added to Adult Wait List
4,086

Total Number of Candidates on Adult Wait List
7,562

12%
of Candidates Have Been on Adult Wait List for 2+ Years

59.7%
of Adult Candidates Diagnosed with Cardiomyopathy

Median Adult Wait Time
5.1 months

New Candidates Added to Pediatric Wait List
694

Total Number of Candidates on Pediatric Wait List
1,087

26%
of Candidates Have Been on Pediatric Wait List for > 90 Days

30%
of Pediatric Candidates are Younger than 1 Year Old

8.2%
of Pediatric Candidates Died

Potential for Failure

Overall Costs

Waiting List

Potential for Failure

25.1%

of Adult Patients Experienced Acute Rejection in the First Year

7.9%

of Adult Patients Died Within 1 Year Post-Transplant

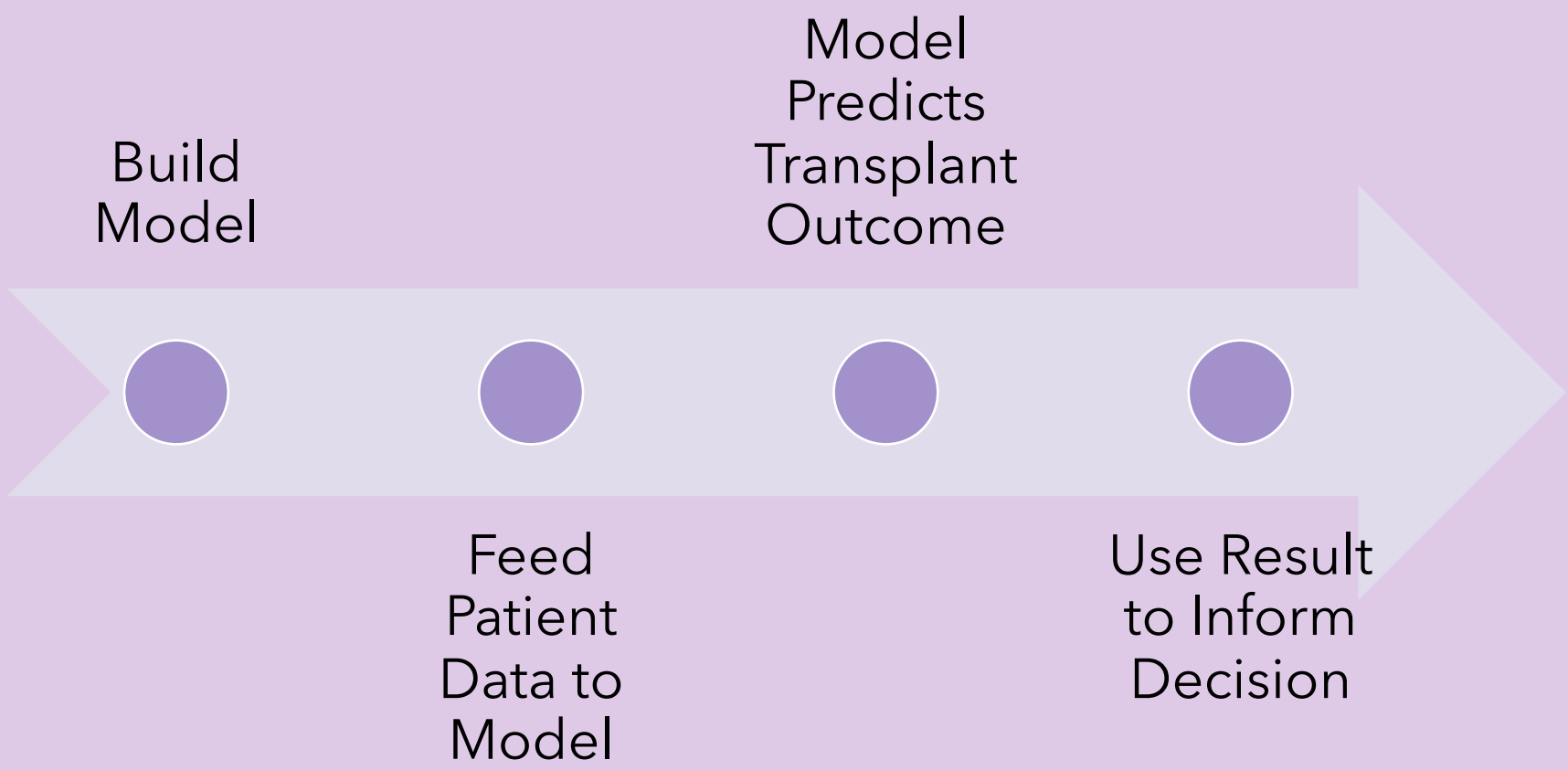
21.2%

of Pediatric Patients Experienced Acute Rejection in the First Year

8.2%

of Pediatric Patients Died Within 1 Year Post-Transplant

Solution: Machine Learning



United Network for Organ Sharing

United **N**etwork for **O**rgan **S**haring

Manages the national transplant waiting list

Maintains database of every transplant in the US

Federally contracted non-profit

Manages each organ match

United **N**etwork for **O**rgan **S**haring

Maintains database of every transplant in the US

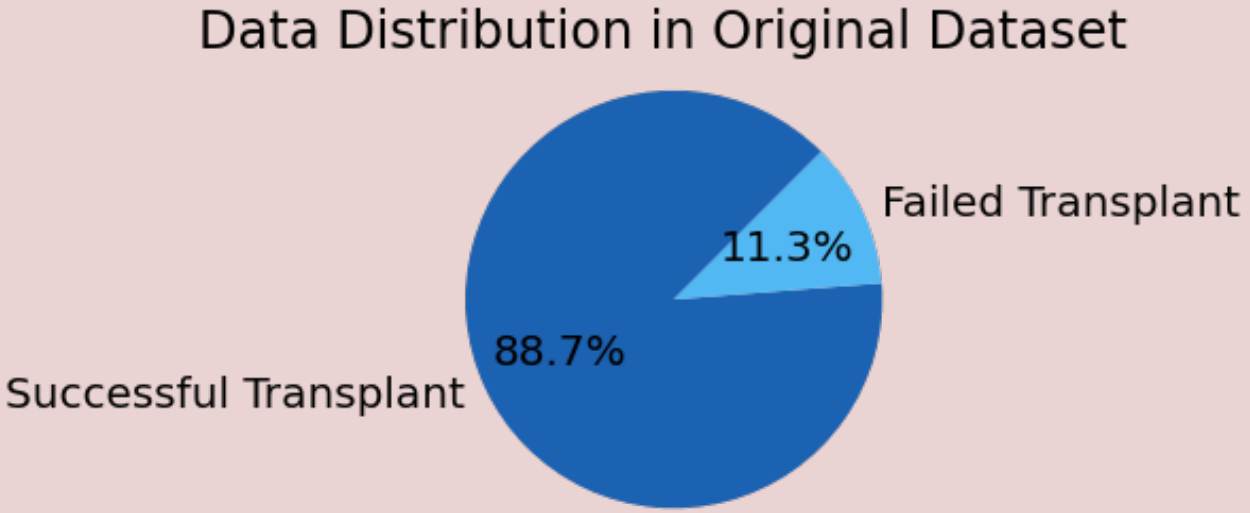
500+ Features

40,000+ Objects

Available to Anyone

United **N**etwork for **O**rgan **S**haring

Maintains database of every transplant in the US



Preprocessing the Data

Preprocessing the Data

Determine Transplant Failure Time

```python
conditions = [
    ((df['GTIME'] <= 365) & (df['GTIME'] > 90) & (df['GSTATUS'] == 0)),
    ((df['GTIME'] <= 365) & (df['GTIME'] > 90) & (df['GSTATUS'] == 1)),
    ((df['GTIME'] > 365) | (df['GSTATUS'] == 0)),
]

choices = [
    np.nan,
    0,
    1
]

df['365DayOutcome'] = np.select(conditions, choices, default = np.nan)
```

GTIME: Graft Lifespan (in days)
GSTATUS: Graft Failed (1 = Yes)

Outcomes

- nan: Last checkup between 90 and 365 days with successful graft
- 0: Graft failed between 90 and 365 days
- 1: Graft successful at 365 days

Preprocessing the Data

Feature Engineering

```
cleanedDataset365['GENDER'] = cleanedDataset365['GENDER'].map({'M':0, 'F':1})
cleanedDataset365['ABOMAT'] = cleanedDataset365['ABOMAT'].map({'ABO incompatible':0, 'ABO compatible':1, 'ABO identical':2})
cleanedDataset365['PROTEIN_URINE'] = cleanedDataset365['PROTEIN_URINE'].astype(bool)
```

GENDER: M/F → 0/1

ABOMAT:
ABO incompatible/ABO compatible/ABO identical → 0/1/2

PROTEIN_URINE: Y/N → True/False

Preprocessing the Data

Null Valu

Filter Features

Feature

Round 1:

```
df_1 = cleanedDataset365.filter(['365DaySurvival','GENDER','LV_EJECT',
                                 'NUM_PREV_TX','DAYSWAIT_CHRON','ABOMAT',
                                 'TX_YEAR','PHMratio','vol_quartile','TBILI',
                                 'PROTEIN_URINE','AGE'], axis=1)
```

Round 2:

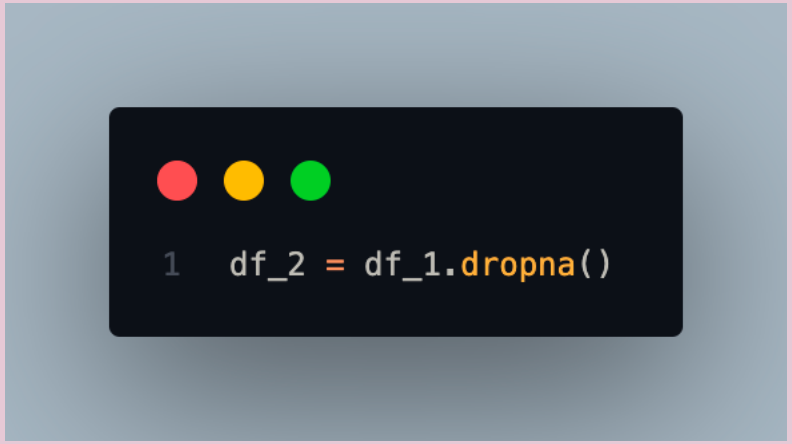
```
filtered_df = cleanedDataset365.select_dtypes(include=['int64','float64','bool'])
```

Preprocessing the Data

echnique (SMOTE)

Remove Objects with Null Values

Filter



Preprocessing the Data

Synthetic Minority Oversampling Technique (SMOTE)

```python
from imblearn.combine import SMOTEENN

X = df_smote.drop(['365DaySurvival'], axis = 1)
y = df_smote['365DaySurvival']

smt = SMOTEENN(random_state=42)
X, y = smt.fit_resample(X, y)
```

- Uses KNN to connect two minority datapoints
- Creates new datapoint along that line
- Continues until classes are balanced

- May result in ambiguous datapoints

Preprocessing the Data

Edited Nearest Neighbor (ENN)

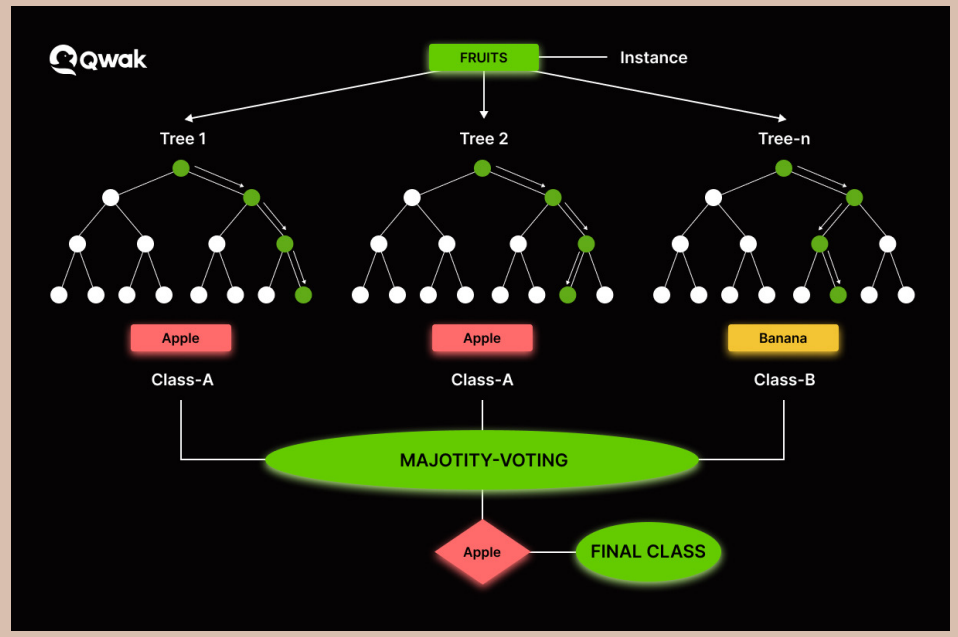
```python
from imblearn.combine import SMOTEENN

X = df_smote.drop(['365DaySurvival'], axis = 1)
y = df_smote['365DaySurvival']

smt = SMOTEENN(random_state=42)
X, y = smt.fit_resample(X, y)
```

- Undersampling technique
- Removes instances of majority class on boundary between classes
- Creates a more obvious distinction between classes

XGBoost vs. Random Forest

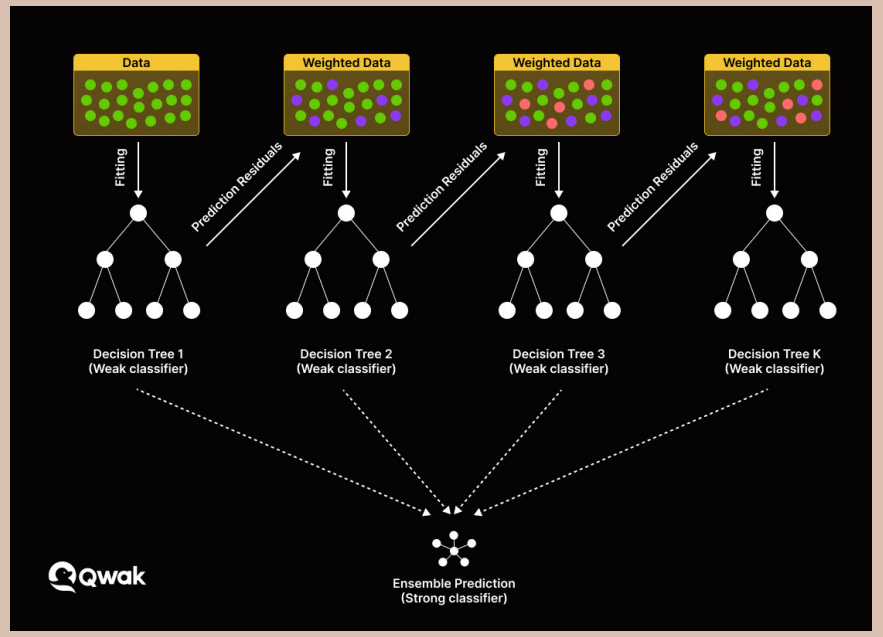
Random Forest

- Bagging algorithm
- Collection of decision trees
- Each tree is a weak learner
- Each tree learns from a subset of the original features
- Majority voting to determine class



XGBoost

- Boosting algorithm
- Based on Gradient Boosting
- Each classifier is a decision tree
- First learner makes predictions
- Error is calculated using loss function
- Residuals passed to next learner



Model Performance

Model Validation

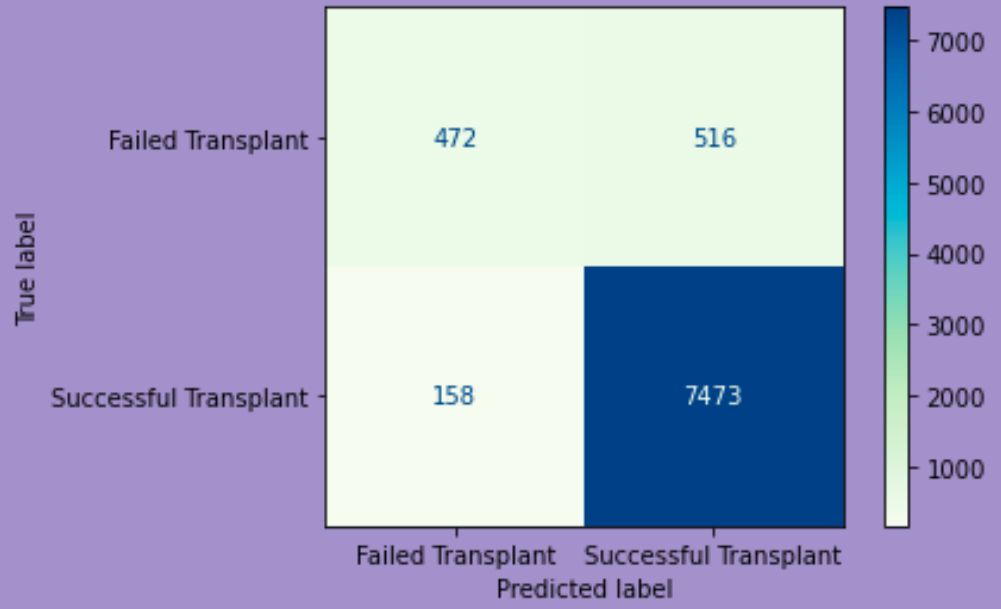
Features

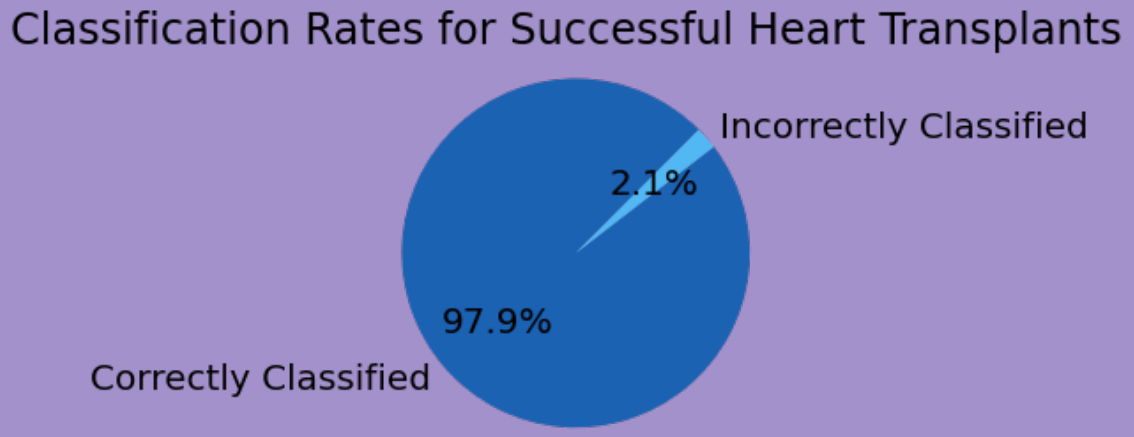
Model Performance

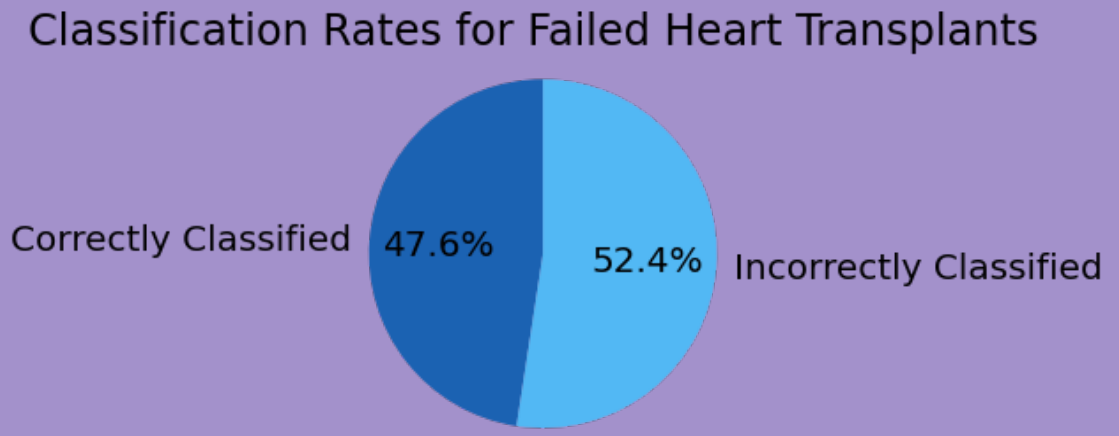
Round 1

Physician-Specified Features with XGBoost

Metric	Score
Accuracy	0.922
Precision	0.935
Recall	0.98
Specificity	0.476
F1 Score	0.957
AUC ROC	0.728





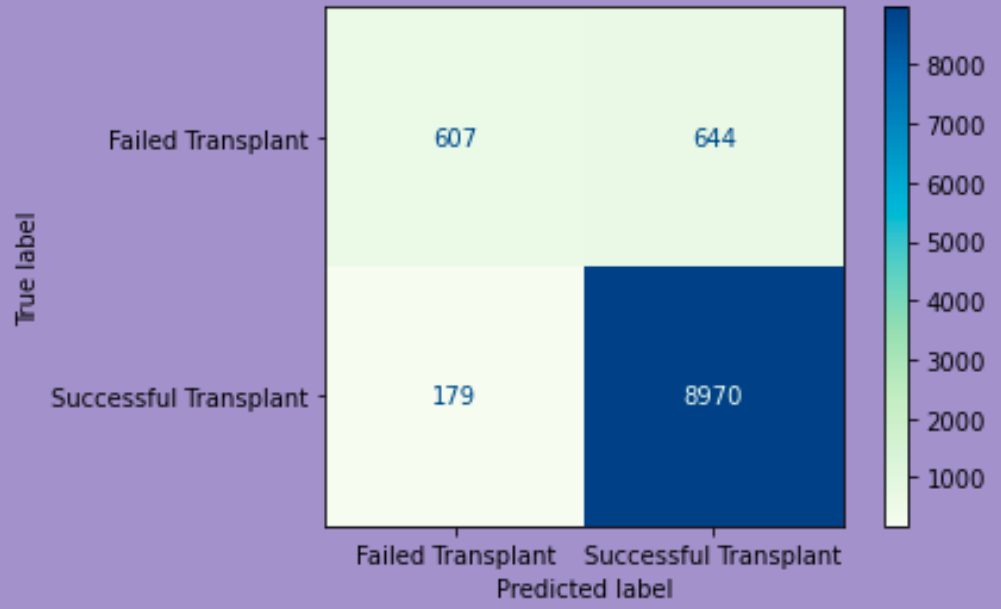


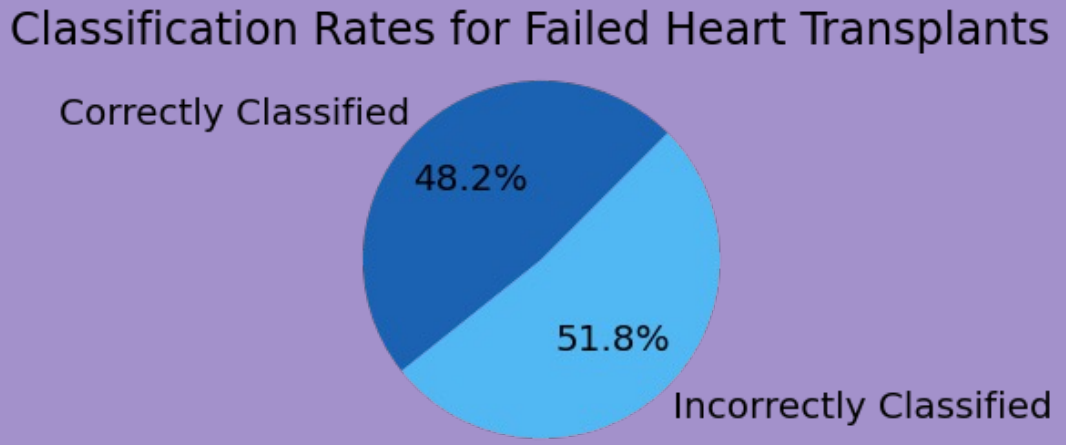
Model Performance

Round 2

Physician-Specified Features with Random Forest

Metric	Score
Accuracy	0.921
Precision	0.933
Recall	0.98
Specificity	0.485
F1 Score	0.956
AUC ROC	0.733



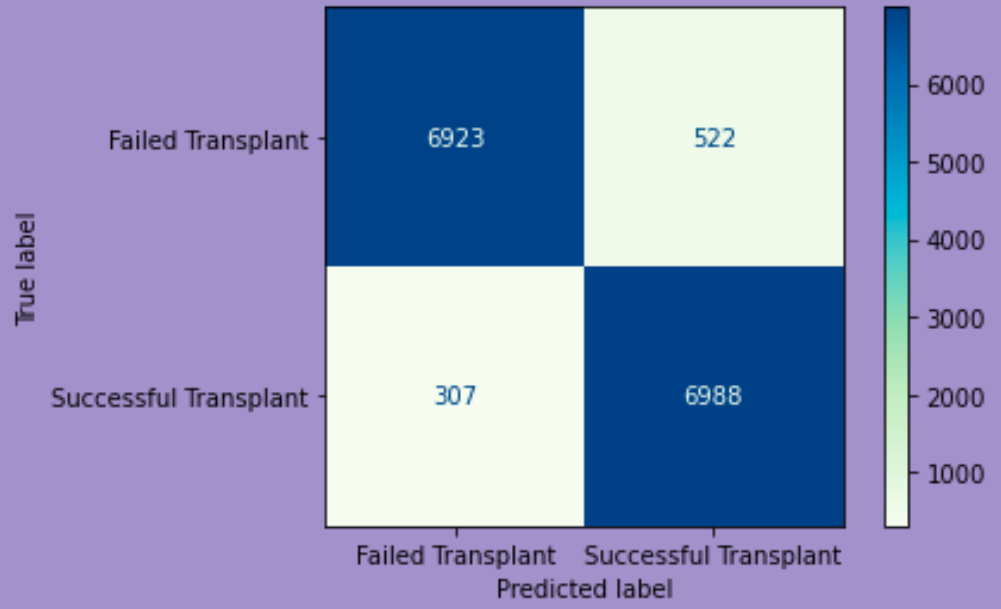


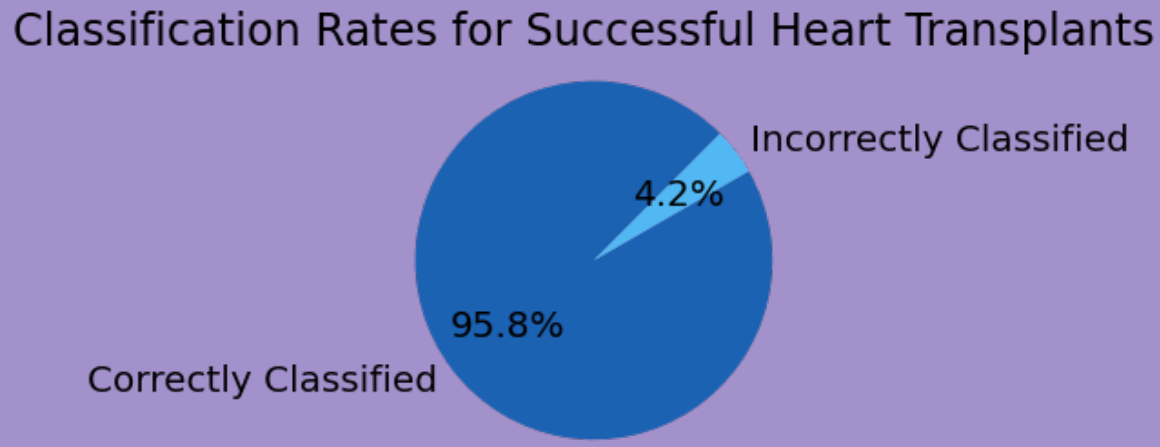
Model Performance

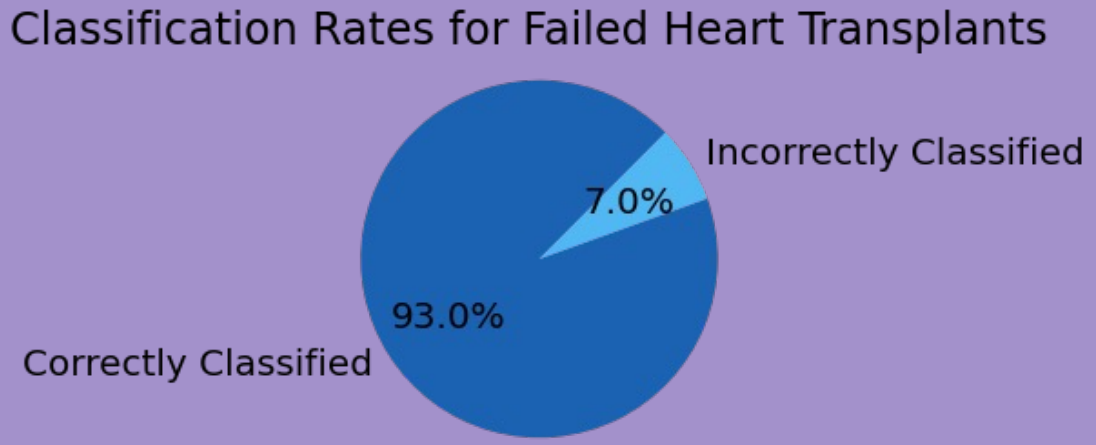
Round 3

Physician-Specified Features with XGBoost and SMOTE

Metric	Score
Accuracy	0.944
Precision	0.93
Recall	0.958
Specificity	0.93
F1 Score	0.944
AUC ROC	0.944





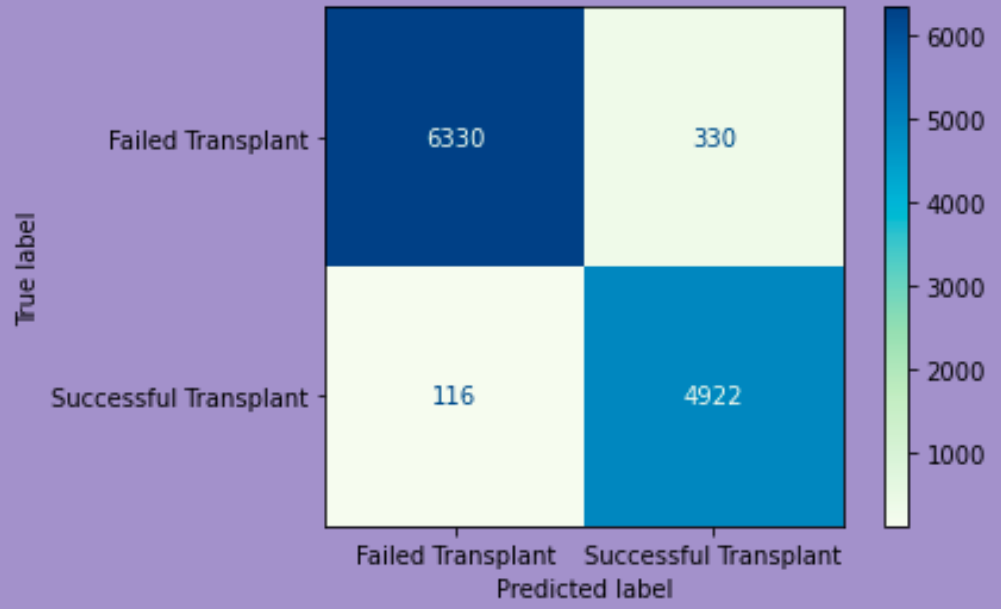


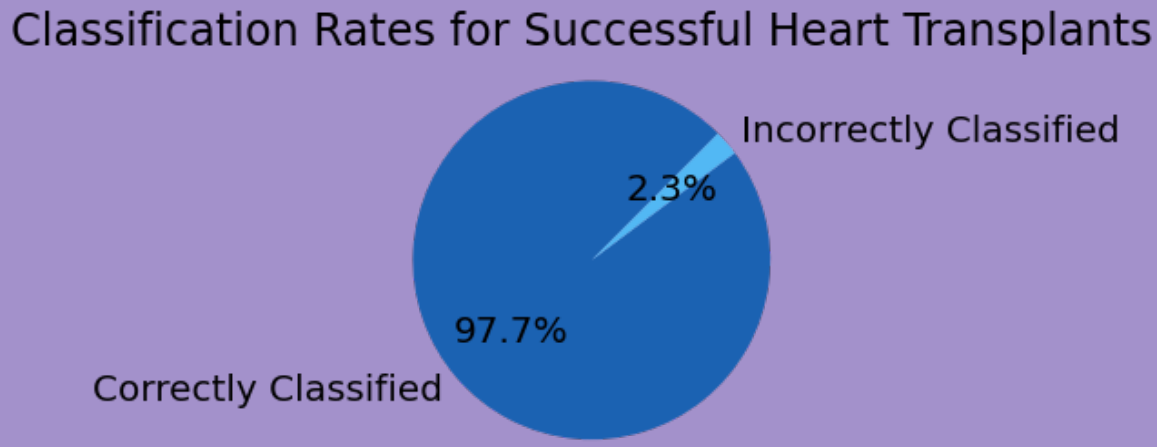
Model Performance

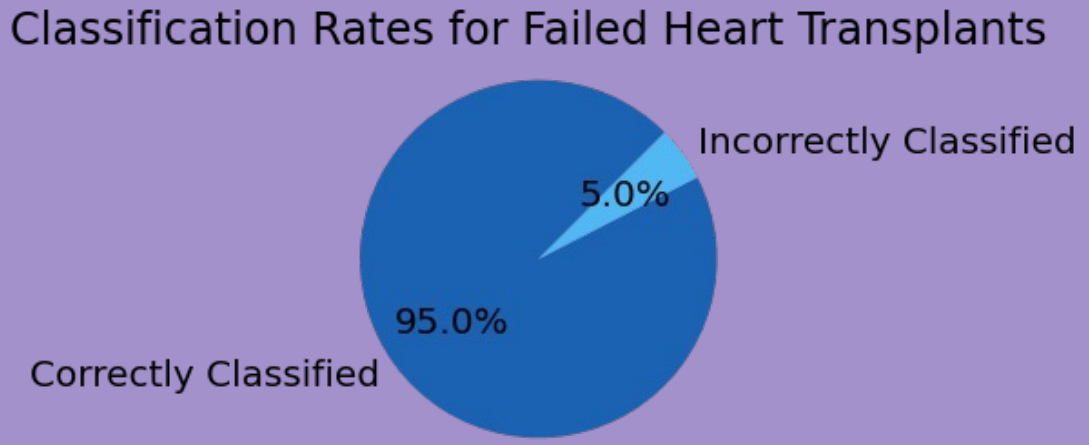
Round 4

Physician-Specified Features with XGBoost, SMOTE, and ENN

Metric	Score
Accuracy	0.962
Precision	0.937
Recall	0.977
Specificity	0.95
F1 Score	0.957
AUC ROC	0.964





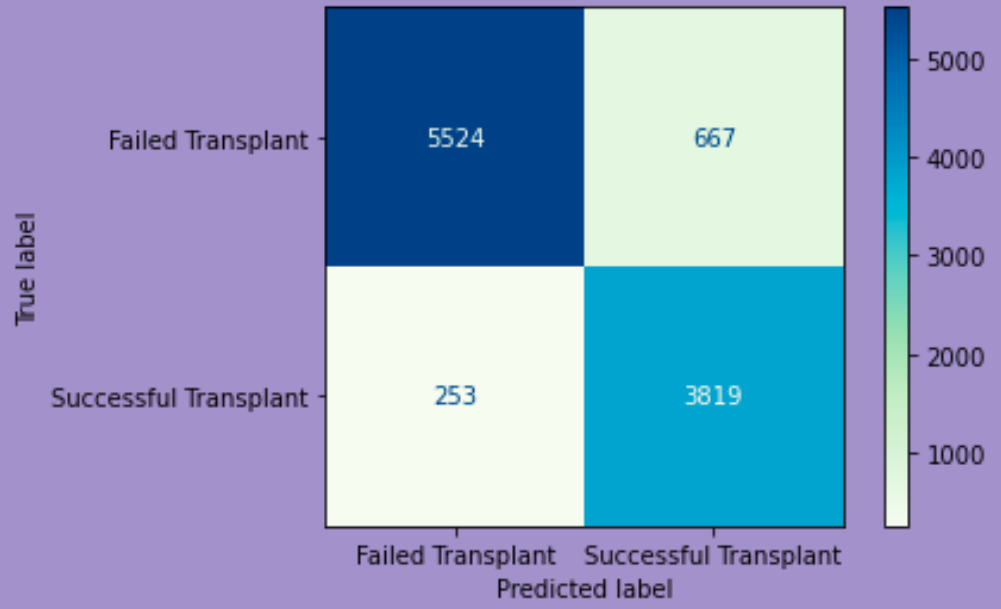


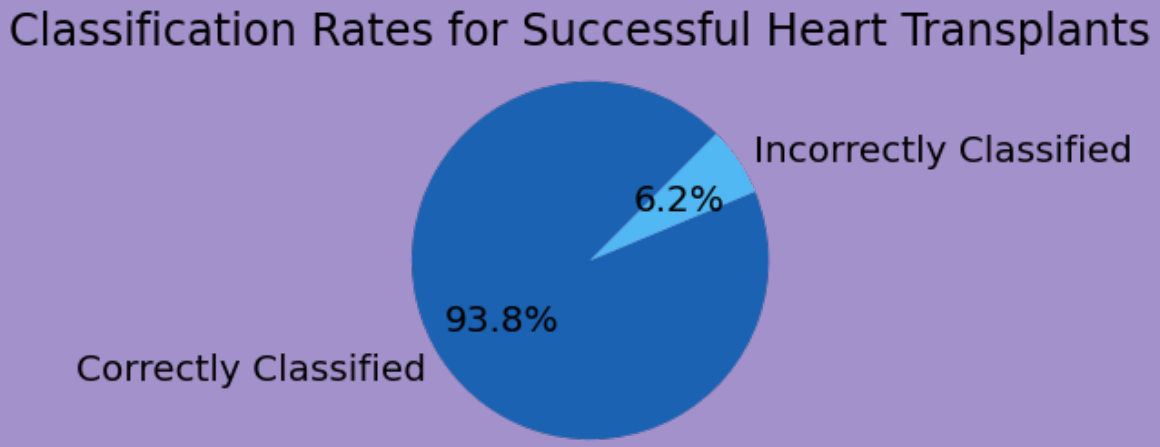
Model Performance

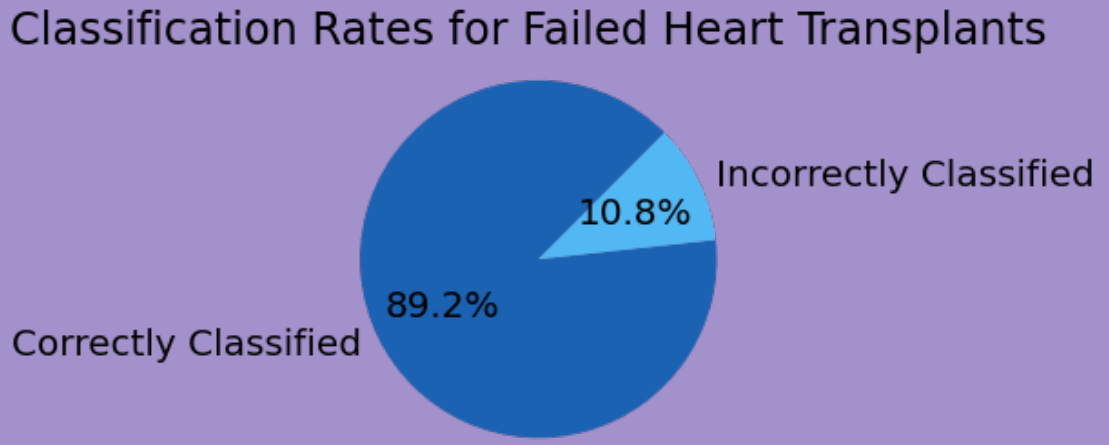
Round 5

Physician-Specified Features with XGBoost, SMOTE, and ENN without TX_YEAR

Metric	Score
Accuracy	0.91
Precision	0.851
Recall	0.938
Specificity	0.892
F1 Score	0.892
AUC ROC	0.915

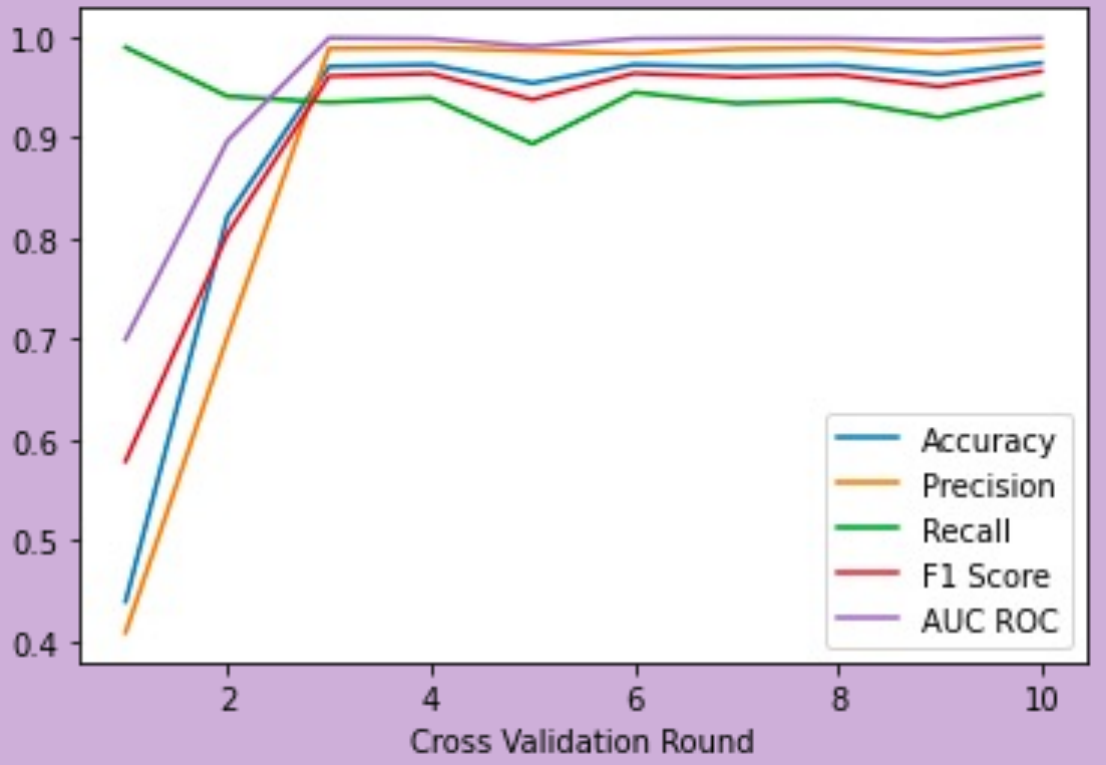






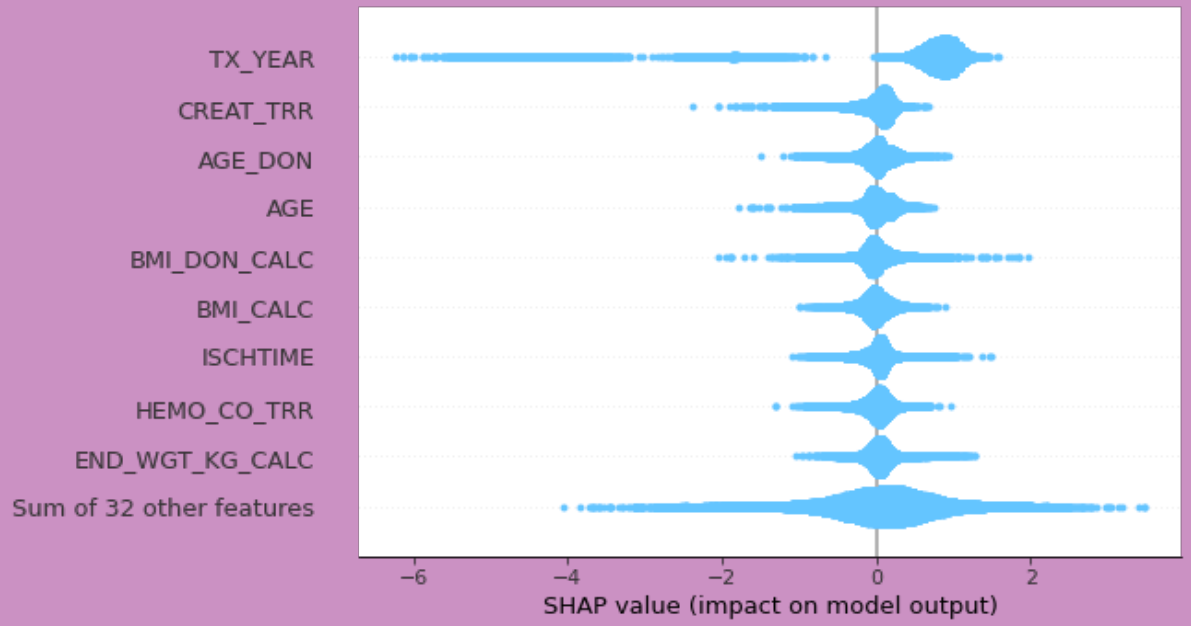
Model Validation

```
1  from sklearn.model_selection import cross_validate
2  scores = cross_validate(model, X, y, cv=10, scoring=['accuracy','precision','recall','f1','roc_auc'])
```

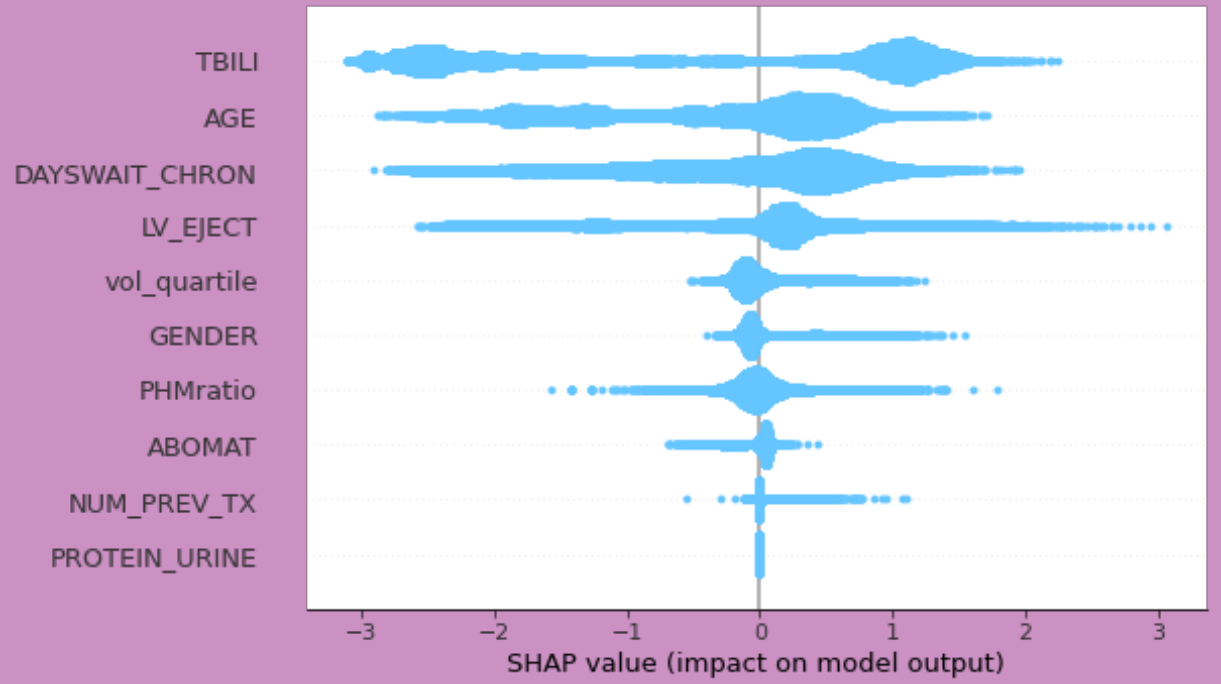


Features

Key Features for XGBoost without SMOTE/ENN Including All Features



Key Features for XGBoost with SMOTE/ENN with Physician Filtered Columns, no TX_YEAR



Conclusions

Model 5

Physician-Specified Features with XGBoost, SMOTE, and ENN without TX_YEAR

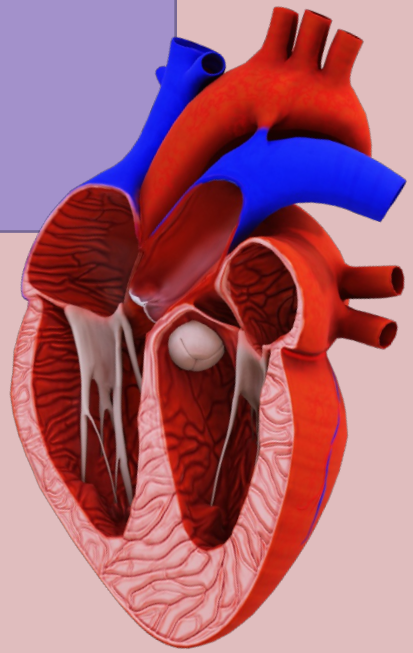
Metric	Score
Accuracy	0.91
Precision	0.851
Recall	0.938
Specificity	0.892
F1 Score	0.892
AUC ROC	0.915

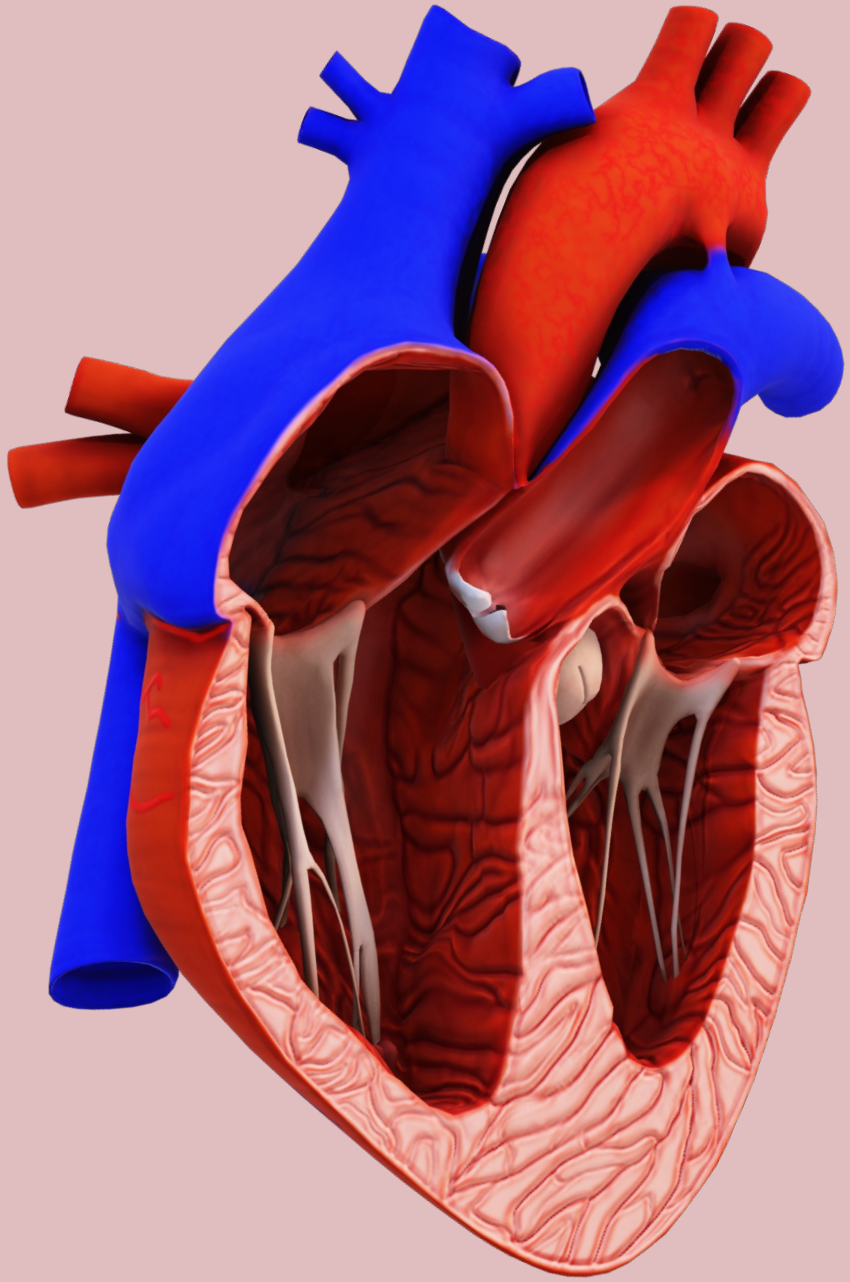
Assumptions

- TX_YEAR shouldn't be included in model
- Physician recommendations

Limitations

- Without SMOTE, failures cannot be predicted





Thank You