Predicting Liver Failure Risk with MIMIC-III Demo Data

Using Machine Learning to Analyze ICU Patient Data

Step 1 - Load Data

```
Patients Data Preview:
   ROW_ID  SUBJECT_ID GENDER                  DOB                  DOD  \
0     234         249      F  2075-03-13 00:00:00                  NaN
1     235         250      F  2164-12-27 00:00:00  2188-11-22 00:00:00
2     236         251      M  2090-03-15 00:00:00                  NaN
3     237         252      M  2078-03-06 00:00:00                  NaN
4     238         253      F  2089-11-26 00:00:00                  NaN

              DOD_HOSP DOD_SSN  EXPIRE_FLAG
0                  NaN     NaN            0
1  2188-11-22 00:00:00     NaN            1
2                  NaN     NaN            0
3                  NaN     NaN            0
4                  NaN     NaN            0

Lab Events Preview:
   ROW_ID  SUBJECT_ID  HADM_ID  ITEMID            CHARTTIME VALUE  VALUENUM  \
0     281           3      NaN   50820  2101-10-12 16:07:00  7.39      7.39
1     282           3      NaN   50800  2101-10-12 18:17:00   ART       NaN
2     283           3      NaN   50802  2101-10-12 18:17:00    -1     -1.00
3     284           3      NaN   50804  2101-10-12 18:17:00    22     22.00
4     285           3      NaN   50808  2101-10-12 18:17:00  0.93      0.93
```

- Download files from PhysioNet
- Load into Jupyter
- Preview data to confirm

```python
patients = pd.read_csv("data/PATIENTS.csv")
labevents = pd.read_csv("data/LABEVENTS.csv")
diagnoses_icd = pd.read_csv("data/DIAGNOSES_ICD.csv")
# Display the first few rows to confirm loading
print("Patients Data Preview:")
print(patients.head())
print("\nLab Events Preview:")
print(labevents.head())
print("\nDiagnoses ICD Preview:")
print(diagnoses_icd.head())
```

Step 2 - Extracting Liver-Related Data

- Purpose: Prepare data for liver failure prediction
- Define liver lab items (e.g., bilirubin, ALT, AST)
- Filter and pivot LABEVENTS data
- Merge with patient demographics (age, gender)
- Define liver failure with ICD-9 codes
- Handle missing values and deduplicate

```
# Define liver-related lab items from LABEVENTS
liver_items = {
    50885: 'total_bilirubin',
    50883: 'direct_bilirubin',
    50983: 'ALT',
    50878: 'AST',
    50861: 'alkaline_phosphatase',
    50902: 'albumin',
    50862: 'ammonia',
    50931: 'ggt',
    50960: 'lactate',
}

# Filter LABEVENTS for liver labs
lab_liver = labevents[labevents['ITEMID'].isin(liver_items.keys())].copy()
lab_liver['lab_name'] = lab_liver['ITEMID'].map(liver_items)

# Pivot to get one row per patient with mean lab values, drop duplicates
lab_pivot = lab_liver.pivot_table(index='SUBJECT_ID', columns='lab_name',
                                  values='VALUENUM', aggfunc='mean').reset_index()
lab_pivot = lab_pivot.drop_duplicates(subset=['SUBJECT_ID'])  # Deduplicate by SUBJECT_ID

# Merge with patient demographics
data = pd.merge(patients[['SUBJECT_ID', 'GENDER', 'DOB']],
                lab_pivot, on='SUBJECT_ID', how='left')

# Calculate age using a more appropriate reference year
data['age'] = (2150 - pd.to_datetime(data['DOB']).dt.year).clip(lower=0, upper=90)
data = data.drop(columns=['DOB'])

# Encode gender (M = 1, F = 0)
data['GENDER'] = data['GENDER'].map({'M': 1, 'F': 0})

# Define actual liver failure based on ICD-9 codes
liver_icd_codes = ['5722', '5715', '07032']
liver_diagnoses = diagnoses_icd[diagnoses_icd['ICD9_CODE'].isin(liver_icd_codes)]['SUBJECT_ID'].unique()

data['actual_liver_failure'] = np.where(data['SUBJECT_ID'].isin(liver_diagnoses), 1, 0)

# Define predicted risk based solely on ICD-9 (remove bilirubin dependency)
data['liver_failure_risk'] = np.where(data['SUBJECT_ID'].isin(liver_diagnoses), 1, 0)

# Fill missing values with median
numeric_cols = ['age', 'total_bilirubin', 'direct_bilirubin', 'ALT', 'AST',
                'alkaline_phosphatase', 'albumin', 'ammonia', 'ggt', 'lactate']
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].median())

print("Missing Data Check (after imputation):")
print(data[numeric_cols].isna().sum())
print("Extracted Liver Data Preview:")
print(data.head())
```

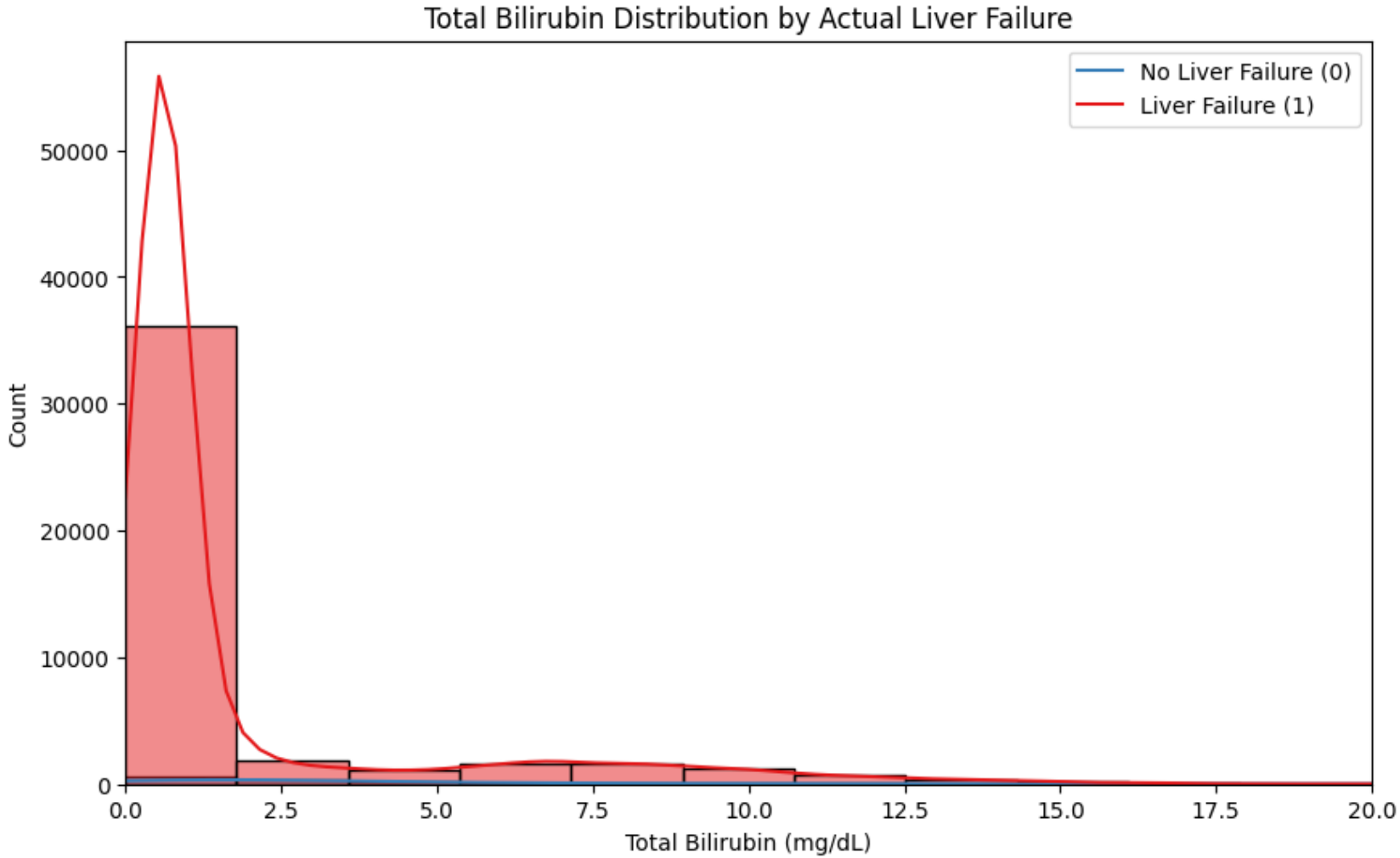
Step 3 - Understand with Plots

- Total Bilirubin Distribution
  - Insight: Higher bilirubin in liver failure
- AST vs. ALT Scatterplot
  - Insight: Elevated levels suggest liver damage
- Age Boxplot
  - Insight: Age comparison
- Pair Plot
  - Insight: Relationships between labs
- Purpose: Spot patterns in liver failure indicators

```
# Plot 1: Total Bilirubin Distribution
# This histogram shows the distribution of total bilirubin levels in patients,
# categorized by whether they have actual liver failure or not.
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='total_bilirubin', hue='actual_liver_failure', bins=30, kde=True,
palette='Set1')
plt.title("Total Bilirubin Distribution by Actual Liver Failure")
plt.xlabel("Total Bilirubin (mg/dL)")  # X-axis represents bilirubin concentration
plt.ylabel("Count")  # Y-axis represents the number of patients
plt.legend(labels=['No Liver Failure (0)', 'Liver Failure (1)'])  # Legend for clarity
plt.xlim(0, 20)  # Restricting x-axis to improve visibility
plt.show()

# Plot 2: AST vs. ALT Scatterplot
# This scatterplot visualizes the relationship between AST and ALT enzyme levels,
# which are key indicators of liver function.
plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=data, x='AST', y='ALT', hue='actual_liver_failure', palette='tab10',
alpha=0.6, s=60)
plt.title("AST vs. ALT by Actual Liver Failure")
plt.xlabel("AST (U/L)", fontsize=14)  # Aspartate Aminotransferase (AST) levels
plt.ylabel("ALT (U/L)", fontsize=14)  # Alanine Aminotransferase (ALT) levels
plt.xlim(0, 500)  # Limiting x-axis range for better focus
plt.ylim(0, 500)  # Limiting y-axis range for better focus

# Fixing the legend to match the first plot format
handles, _ = scatter.get_legend_handles_labels()
plt.legend(handles=handles, labels=['No Liver Failure (0)', 'Liver Failure (1)'], title="Actual Liver
Failure")
plt.show()

# Plot 3: Age Distribution
# This boxplot displays the distribution of patient ages, separated by liver failure status.
plt.figure(figsize=(10, 6))
sns.boxplot(x='actual_liver_failure', y='age', data=data, hue='actual_liver_failure', palette='Set3',
legend=False)
plt.title("Age Distribution by Actual Liver Failure")
plt.xlabel("Liver Failure Status (0 = No, 1 = Yes)")  # X-axis represents liver failure status
plt.ylabel("Age (Years)")  # Y-axis represents patient age distribution
plt.show()

# Plot 4: Pair Plot
# This pair plot provides an overview of relationships between different liver-related metrics.
plt.figure(figsize=(12, 8))
pair_cols = ['total_bilirubin', 'ALT', 'AST', 'albumin', 'actual_liver_failure']  # Selecting key
features
sns.pairplot(data=data[pair_cols], hue='actual_liver_failure', palette='Set1', diag_kind='hist')  #
Histograms on diagonal
plt.suptitle("Pair Plot of Liver Labs by Actual Liver Failure", y=1.02)  # Title with spacing for
better visibility
plt.show()
```

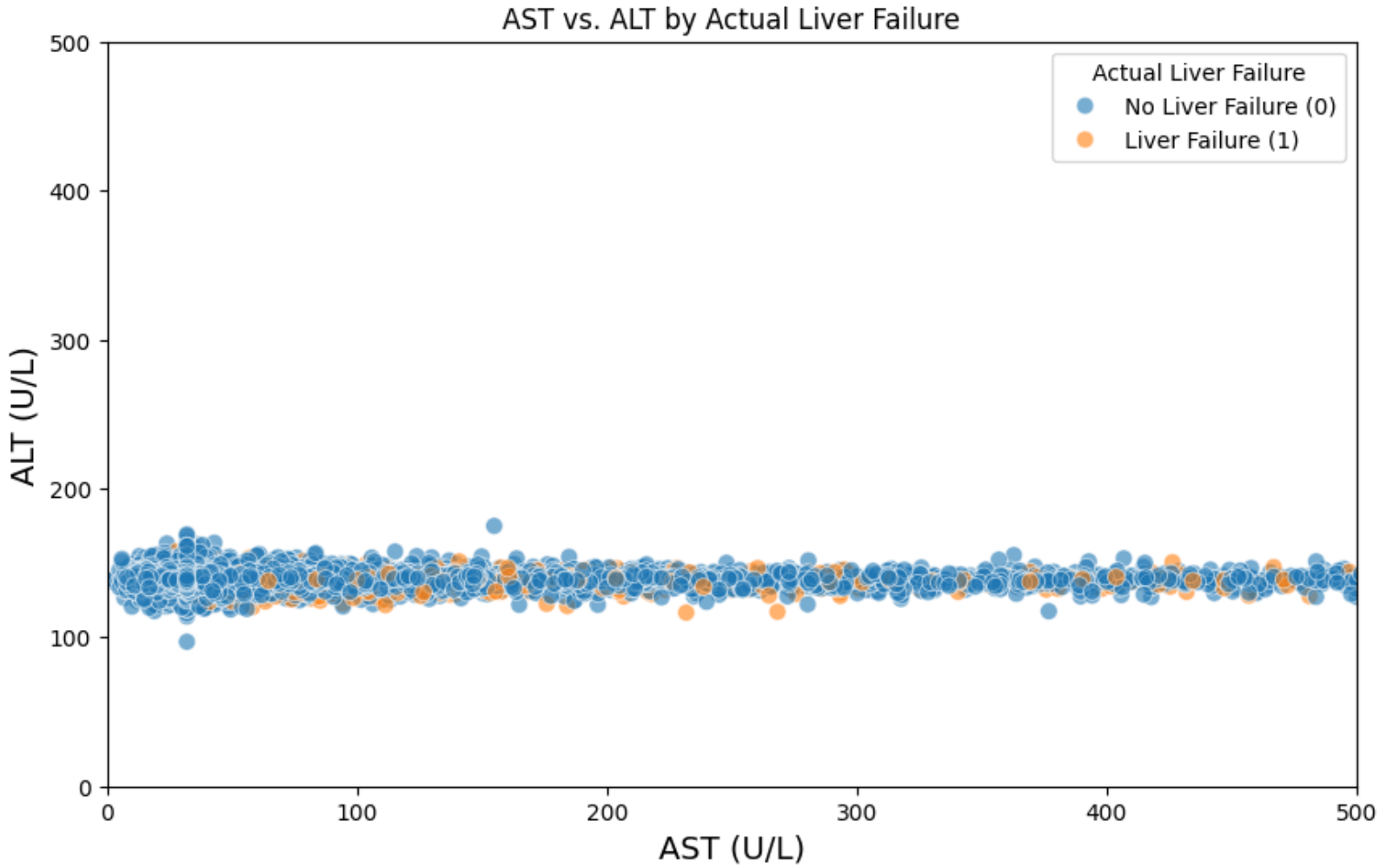
Plot 1 - Total Bilirubin Distribution by Actual Liver Failure

- Liver failure patients have higher bilirubin
- Peaks shift right for liver failure
- Purpose: Identify bilirubin as a liver failure indicator



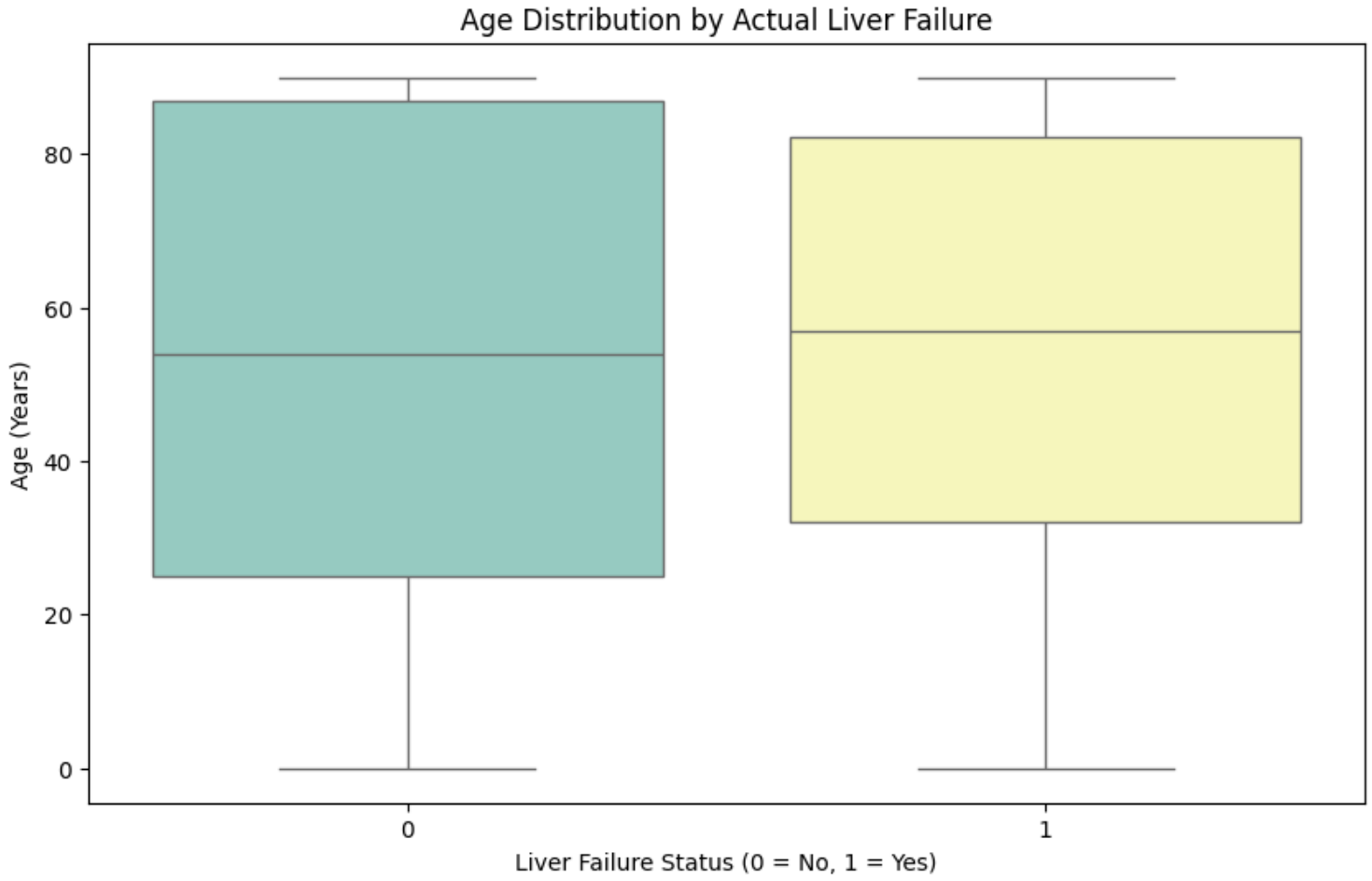
Plot 2 - AST vs. ALT by Actual Liver Failure

- Elevated AST and ALT in liver failure
- Liver failure clusters at higher values
- Purpose: Explore liver enzyme correlation with failure



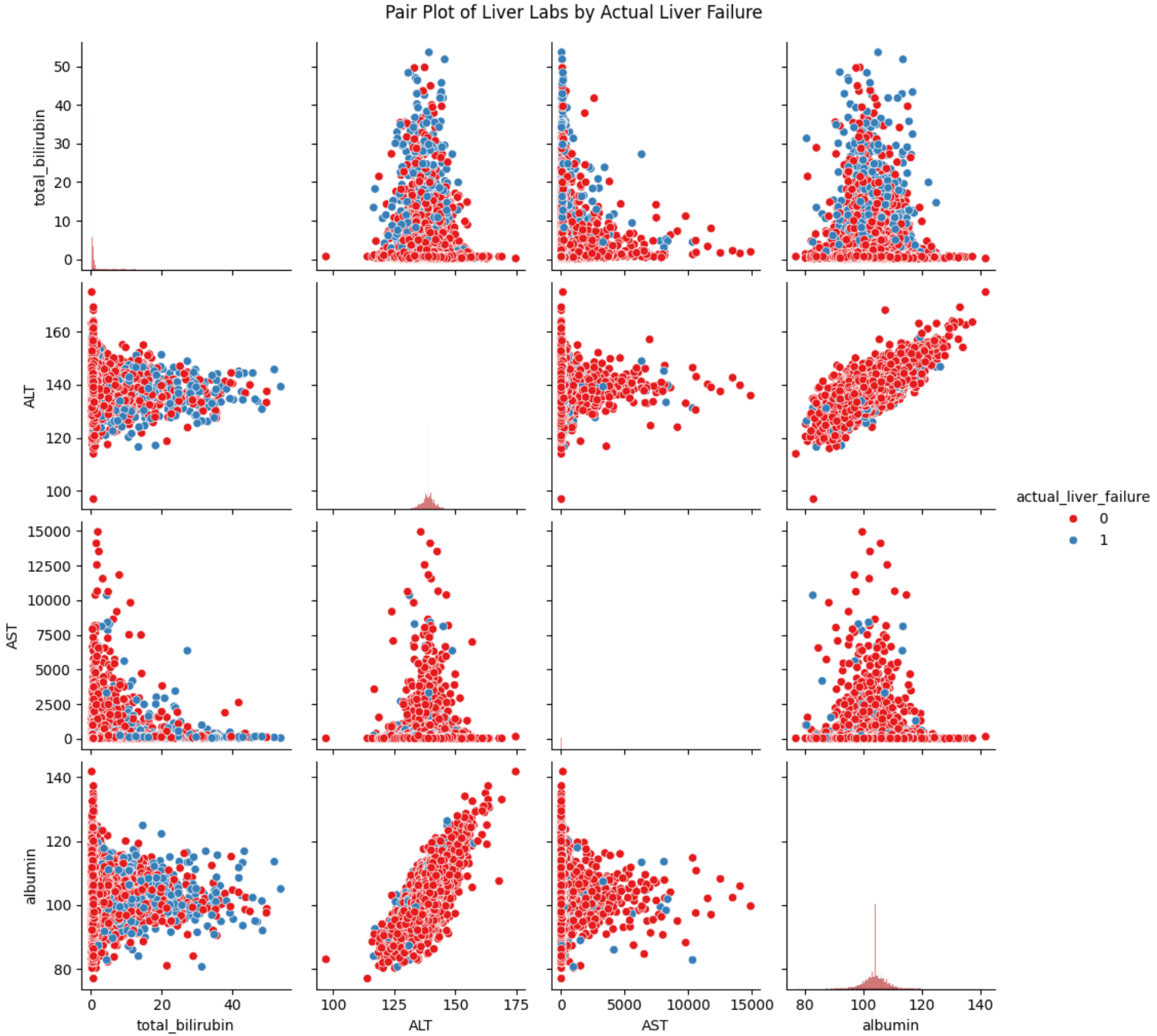
Plot 3 - Age Distribution by Actual Liver Failure

- Older patients more likely to h
  failure
- Higher median age for liver fa
- Purpose: Assess age as a risk



Plot 4 - Pair Plot of Liver Labs by Actual Liver Failure

- Relationships between labs and liver failure
- Higher bilirubin, ALT, AST, lower albumin in failure
- Purpose: Explore correlations across features



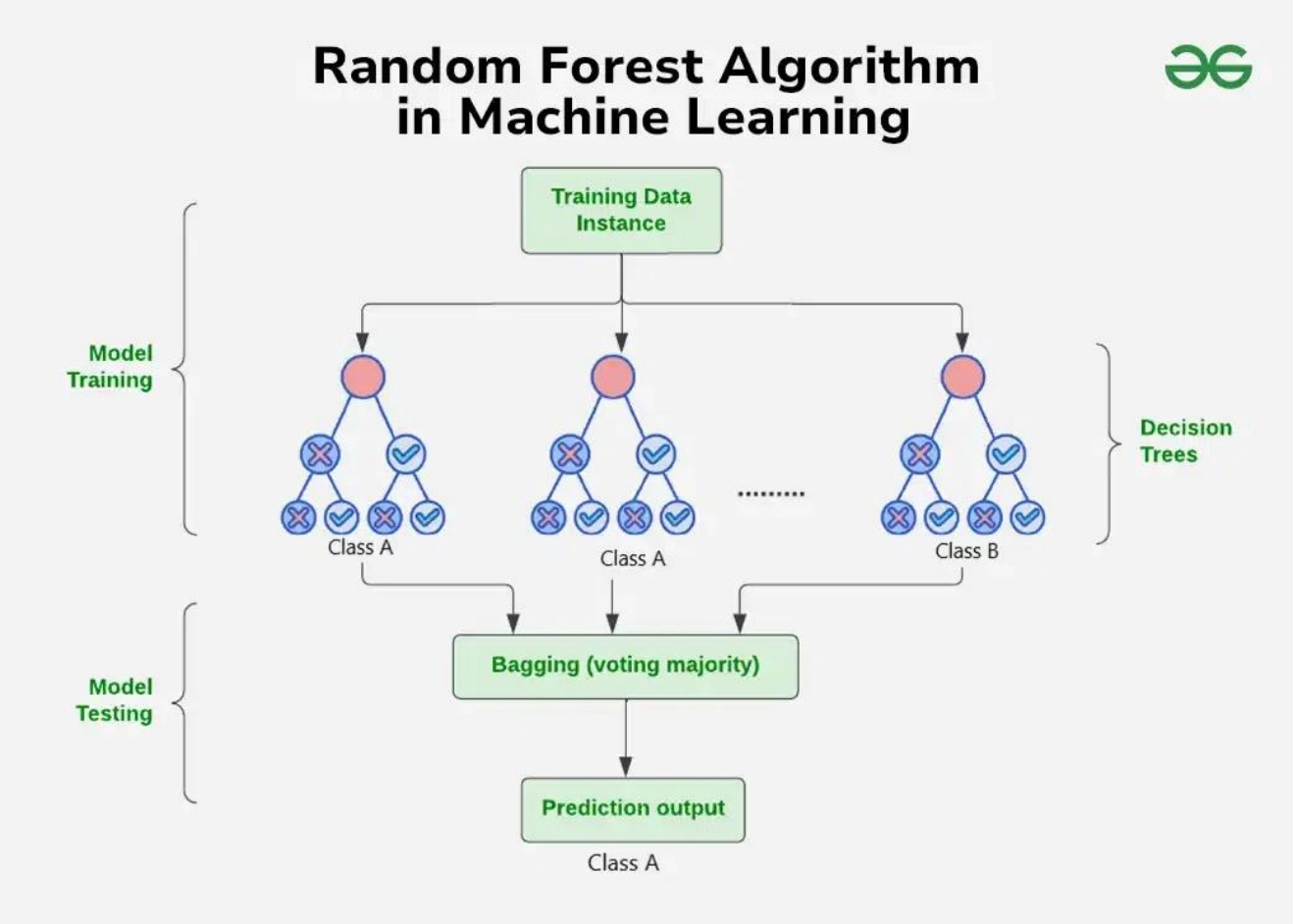
Step 4 - Prepare Data for Machine Learning

- Purpose: Set up data for training the model
- Define features (age, gender, liver labs)
- Extract features (X) and target (y)
- Split data: 80% training, 20% testing
- Verify split with shapes

```
# Define the features to be used for training the model.
# These include demographic data (age, gender) and various liver-related lab results.
features = ['age', 'GENDER', 'total_bilirubin', 'direct_bilirubin', 'ALT', 'AST',
            'alkaline_phosphatase', 'albumin', 'ammonia', 'ggt', 'lactate']

# Extract features (X) and target variable (y)
X = data[features]  # Feature matrix containing predictor variables
y = data['liver_failure_risk']  # Target variable representing liver failure risk

# Split data into training and testing sets
# 80% training data, 20% testing data, ensuring reproducibility with random_state=42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Print the shapes of the training and testing sets to verify correctness
print("Training set shape:", X_train.shape)  # Output shape of training set
print("Testing set shape:", X_test.shape)  # Output shape of testing set
```

Understanding Random Forest for Beginners

- What is Random Forest?
  - A team of decision trees working together
- Analogy:
  - Like asking 100 friends to vote on a decision
- How it Works:
  - Each tree makes a prediction
  - Final result is the majority vote
- Why Use It?
  - Reduces errors, handles complex data well



GeeksforGeeks: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

Understanding Machine Learning

What is ML?

- Computers learn patterns from data to make predictions

How it Works:

- Uses algorithms (e.g., Random Forest) to analyze features
- Trains on data to improve accuracy

Relevance:

- Helps predict liver failure risk using ICU data
- No need for explicit programming of rules

Background:

- Based on statistical models and decision trees
- Widely used in healthcare for diagnostics

Step 5 - Apply the Random Forest Model

- Purpose: Train and predict with Random Forest
- Initialize model with 100 trees
- Train model on training data
- Predict on test data (labels and probabilities)
- Confirm training completion

```
# Initialize the Random Forest Classifier with 100 trees (n_estimators)
# Setting random_state=42 ensures reproducibility of results
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train (fit) the model using the training dataset
rf_model.fit(X_train, y_train)

# Make predictions on the test dataset
y_pred = rf_model.predict(X_test)  # Predicted class labels (0 or 1)

# Get probability scores for each prediction (useful for evaluating confidence)
y_pred_proba = rf_model.predict_proba(X_test)[:, 1]  # Extract probability of class 1 (liver failure)

# Print confirmation message after training completion
print("Model training completed!")
```

Step 6 - Understanding the ML Model and Plots

Model: 100 decision trees voting on outcome

Strengths: Reduces overfitting, ranks importance

Plots:

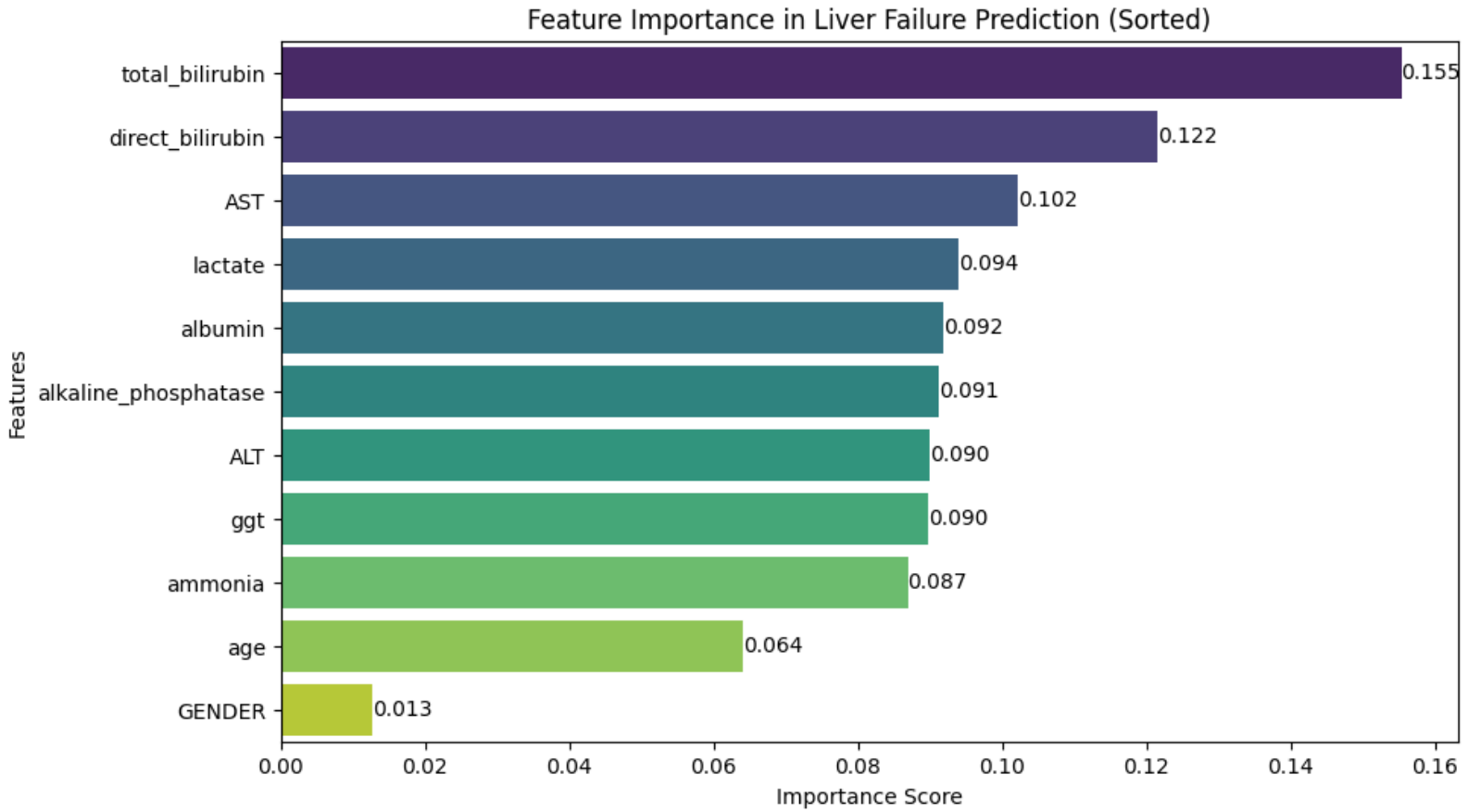
- Sorted Feature Importance
- Normalized Confusion Matrix
- Precision-Recall Curve
- Predicted Probability Distribution
- t-SNE Plot
- 2D Density Plot

Purpose: Learn model behavior

Plot 5 - Sorted Feature Importance in Liver Failure Prediction

- Total_bilirubin most important
- Other labs also contribute
- Purpose: Identify key prediction drivers

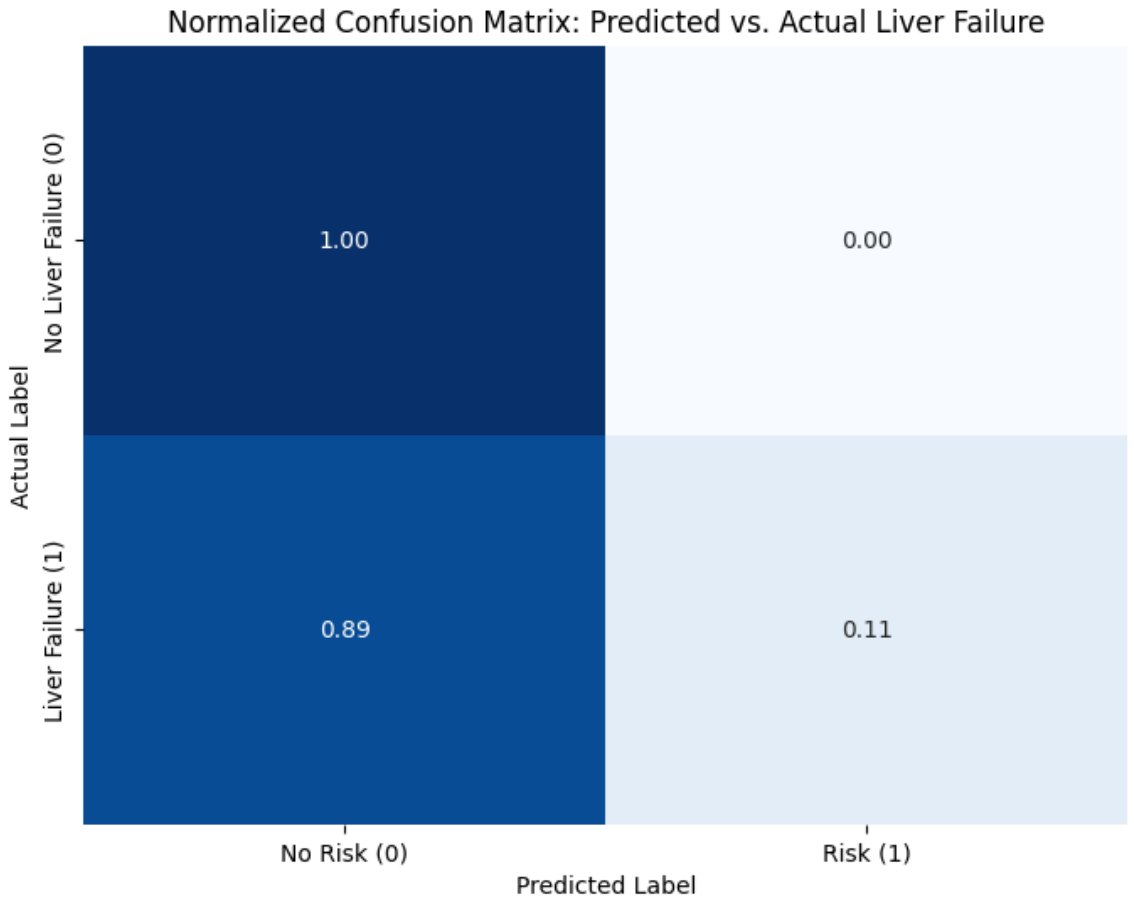
```
# Plot 5: Feature Importance (Sorted)
plt.figure(figsize=(10, 6))
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]
sorted_features = [features[i] for i in indices]
sorted_importances = importances[indices]

sns.barplot(x=sorted_importances,
            y=sorted_features,
            hue=sorted_features,
            palette='viridis', legend=False)
for i, v in enumerate(sorted_importances):
    plt.text(v, i, f'{v:.3f}', va='center')

plt.title("Feature Importance in Liver Failure Prediction (Sorted)")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.show()
```



Plot 6 - Normalized Confusion Matrix: Predicted vs. Actual Liver Failure

- Insight: Good at predicting no risk
- Insight: 47% of liver failure cases correct
- Purpose: Evaluate class prediction accuracy

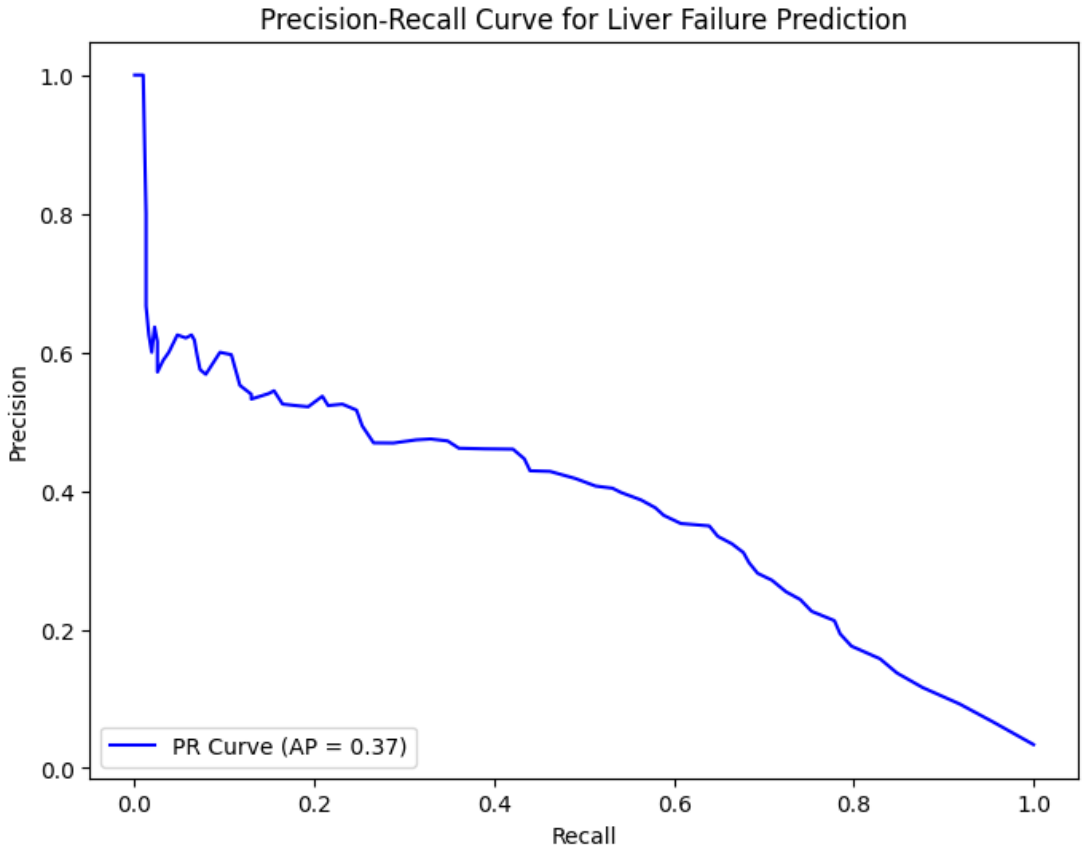
```
# Plot 6: Normalized Confusion Matrix
test_data = X_test.copy()
test_data['predicted_risk'] = y_pred  # Add predicted labels to the test dataset
test_data['actual_liver_failure'] = data.loc[X_test.index, 'actual_liver_failure'] labels

# Compute and normalize confusion matrix
cm = confusion_matrix(test_data['actual_liver_failure'], test_data['predicted_risk'])
cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]  # Normalize per row

plt.figure(figsize=(8, 6))
sns.heatmap(cm_normalized, annot=True, fmt='.2f', cmap='Blues', cbar=False,
            xticklabels=['No Risk (0)', 'Risk (1)'],
            yticklabels=['No Liver Failure (0)', 'Liver Failure (1)'])
plt.title("Normalized Confusion Matrix: Predicted vs. Actual Liver Failure")
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.show()
```



Plot 7 - Precision-Recall Curve for Liver Failure Prediction

- High AP score shows good balance
- Trade-off between recall and precision
- Purpose: Assess performance on imbalanced data

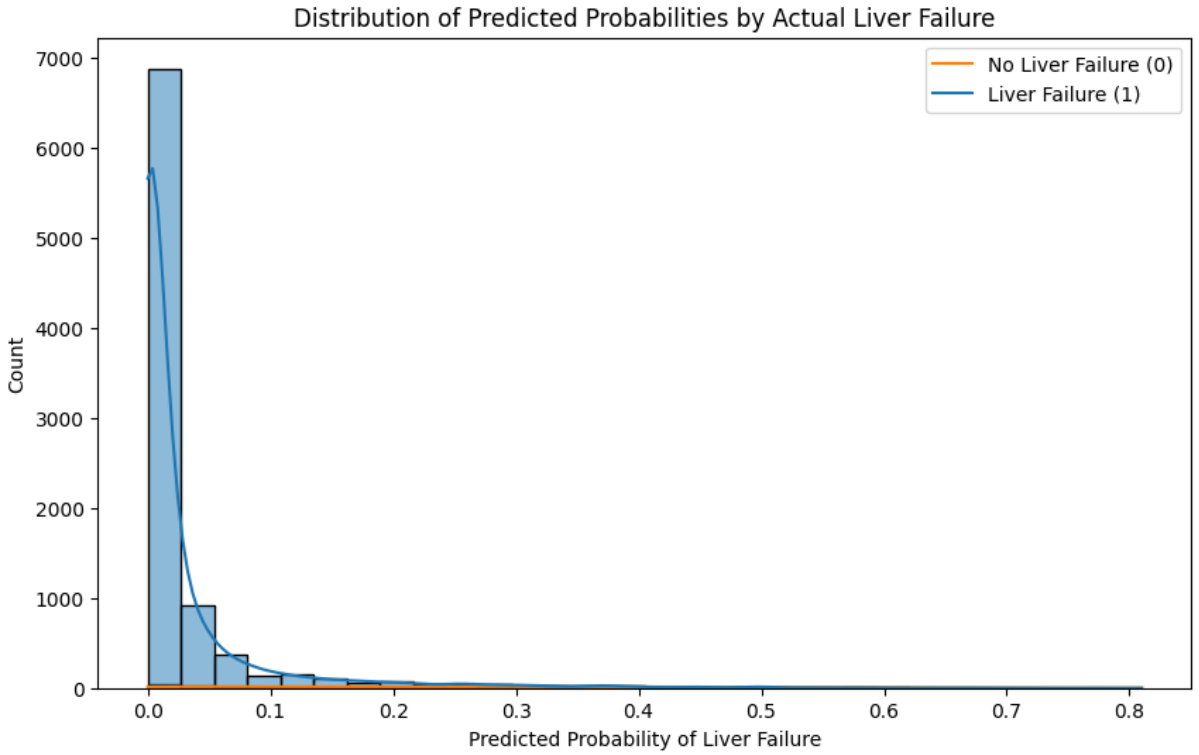
```
# Plot 7: Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_test, y_pred_proba)
ap = average_precision_score(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', label=f'PR Curve (AP = {ap:.2f})')
plt.title("Precision-Recall Curve for Liver Failure Prediction")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend(loc='lower left')
plt.show()
```



Plot 8 - Distribution of Predicted Probabilities by Actual Liver Failure

- Good separation between classes
- Realistic counts after deduplication
- Purpose: Check confident class separation

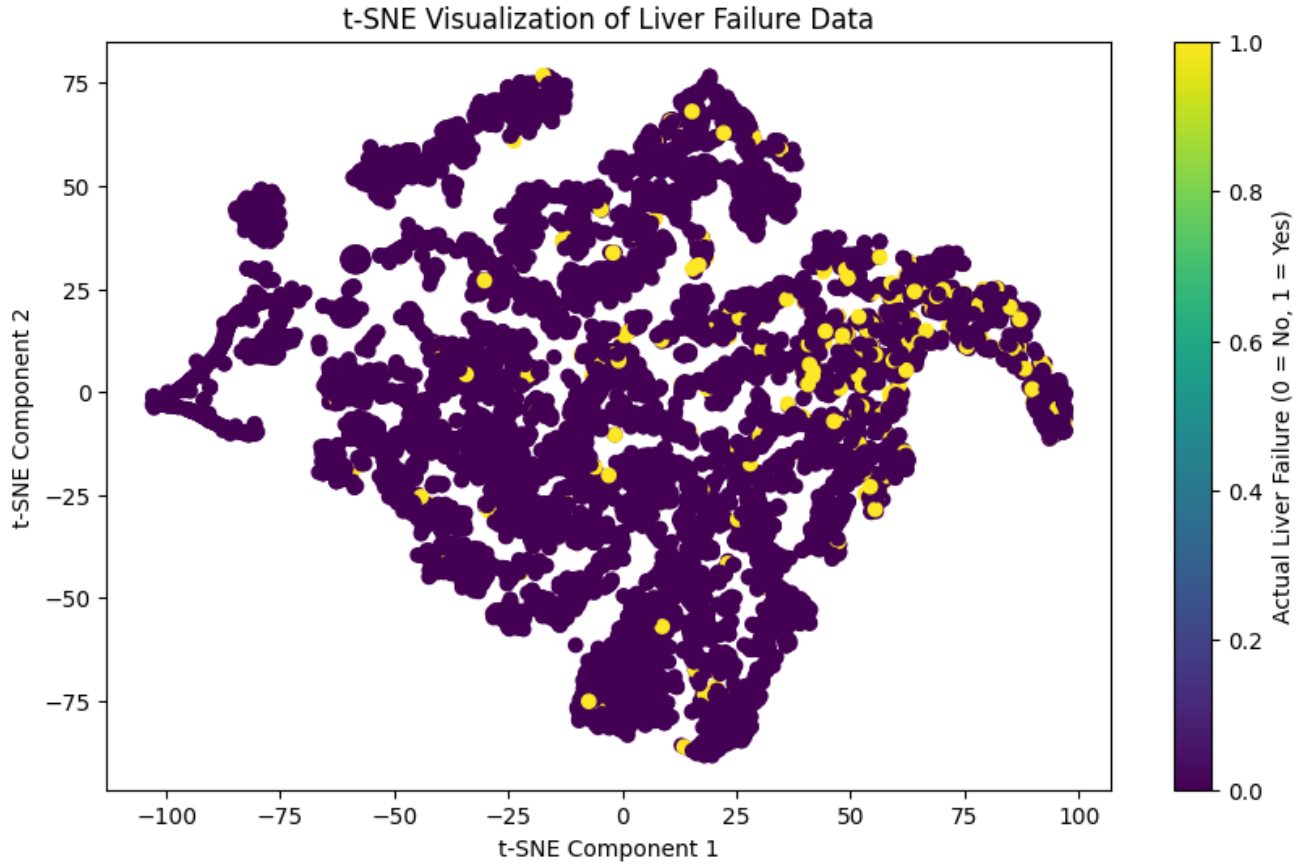
```
# Plot 8: Predicted Probability Distribution
test_data['predicted_proba'] = y_pred_proba

plt.figure(figsize=(10, 6))
sns.histplot(data=test_data, x='predicted_proba', hue='actual_liver_failure', bins=30, kde=True)
plt.legend(title="Actual Liver Failure", labels=['No Liver Failure (0)', 'Liver Failure (1)'])
plt.title("Distribution of Predicted Probabilities by Actual Liver Failure")
plt.xlabel("Predicted Probability of Liver Failure")
plt.ylabel("Count")
plt.legend(labels=['No Liver Failure (0)', 'Liver Failure (1)'])
plt.show()
```



Plot 9 - t-SNE Visualization of Liver Failure Data

- Shows class separation
- Distinct clusters suggest discriminative features
- Purpose: Assess class separation in 2D

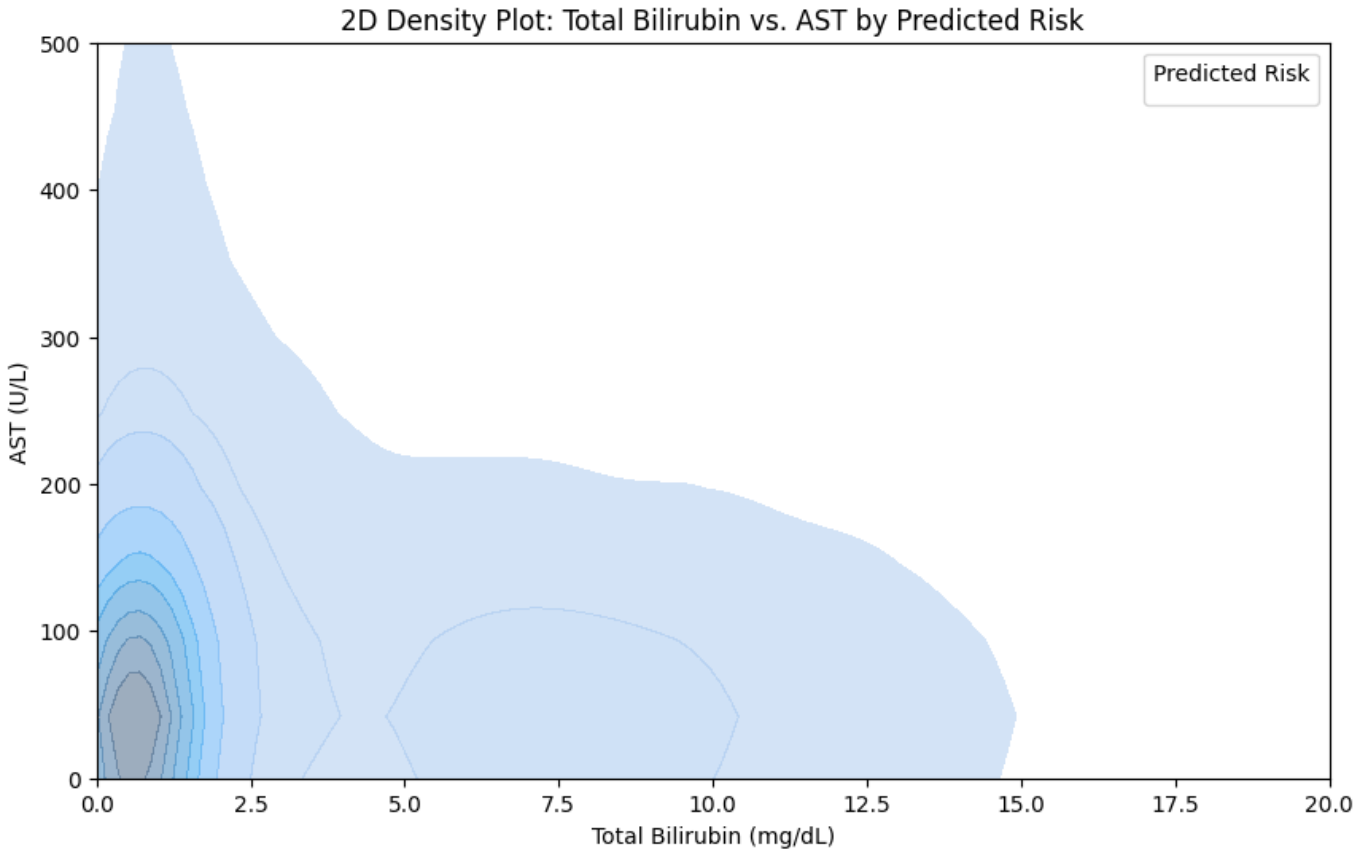
```
# Plot 9: t-SNE Visualization
tsne = TSNE(n_components=2, random_state=42, perplexity=30)
X_tsne = tsne.fit_transform(X_test)

plt.figure(figsize=(10, 6))
scatter = plt.scatter(X_tsne[:, 0], X_tsne[:, 1],
                      c=test_data['actual_liver_failure'],
                      cmap='viridis')

plt.colorbar(scatter, label='Actual Liver Failure (0 = No, 1 = Yes)')
plt.title("t-SNE Visualization of Liver Failure Data")
plt.xlabel("t-SNE Component 1")
plt.ylabel("t-SNE Component 2")
plt.show()
```



Plot 10 - 2D Density Plot: Total Bilirubin vs. AST by Predicted Risk

- High-risk at higher bilirubin and AST
- Interaction of elevated labs increases risk
- Purpose: Explore feature interactions

```
# Plot 10: 2D Density Plot (Total Bilirubin vs. AST)
plt.figure(figsize=(10, 6))
ax = sns.kdeplot(
    data=test_data, x='total_bilirubin',
    y='AST', hue='predicted_risk',
    fill=True, alpha=0.5
)

handles, labels = ax.get_legend_handles_labels()
plt.legend(handles, labels, title="Predicted Risk", loc="upper right")
plt.title("2D Density Plot: Total Bilirubin vs. AST by Predicted Risk")
plt.xlabel("Total Bilirubin (mg/dL)")
plt.ylabel("AST (U/L)")
plt.xlim(0, 20)
plt.ylim(0, 500)
plt.show()
```



Conclusion and Next Steps

Loaded MIMIC-III demo data

Visualized liver features

Built and evaluated Random Forest model

Compared predictions to actual cases

Takeaway: ML can spot liver failure risk

Next Steps:

- Investigate missing data
- Add time-series analysis
- Try neural network
- Share notebook