

# **IS665-852, Group 1, Project 2 Data Mining**

## Comprehensive Model Results

Author: James Mullen

Date: 2025-05-05

This report includes key visualizations, metrics, and summary statistics.

# Table of Contents

## 1. Model Implementation

### 1.1 Source Code

### 1.2 Binary Feature Binning

## 2. Model Analysis

### 2.1 Decision Tree Visualization (max\_depth=4)

### 2.2 Confusion Matrix

### 2.3 Performance Metrics

### 2.4 Feature Importance

### 2.5 Distribution Analysis

## 3. Results Summary

## 1.1 Source Code

```
def entropy(y):  
    p = sum(y=="malignant")/len(y); return -p*log2(p) - (1-p)*log2(1-p)  
def information_gain(y, mask):  
    # ... implementation ...  
# build_tree and DecisionTreeNode methods
```

## 1.2 Binary Feature Binning

texture\_mean: median=18.835

concave points\_worst: median=0.100

symmetry\_worst: median=0.282

radius\_worst: median=14.970

concavity\_worst: median=0.227

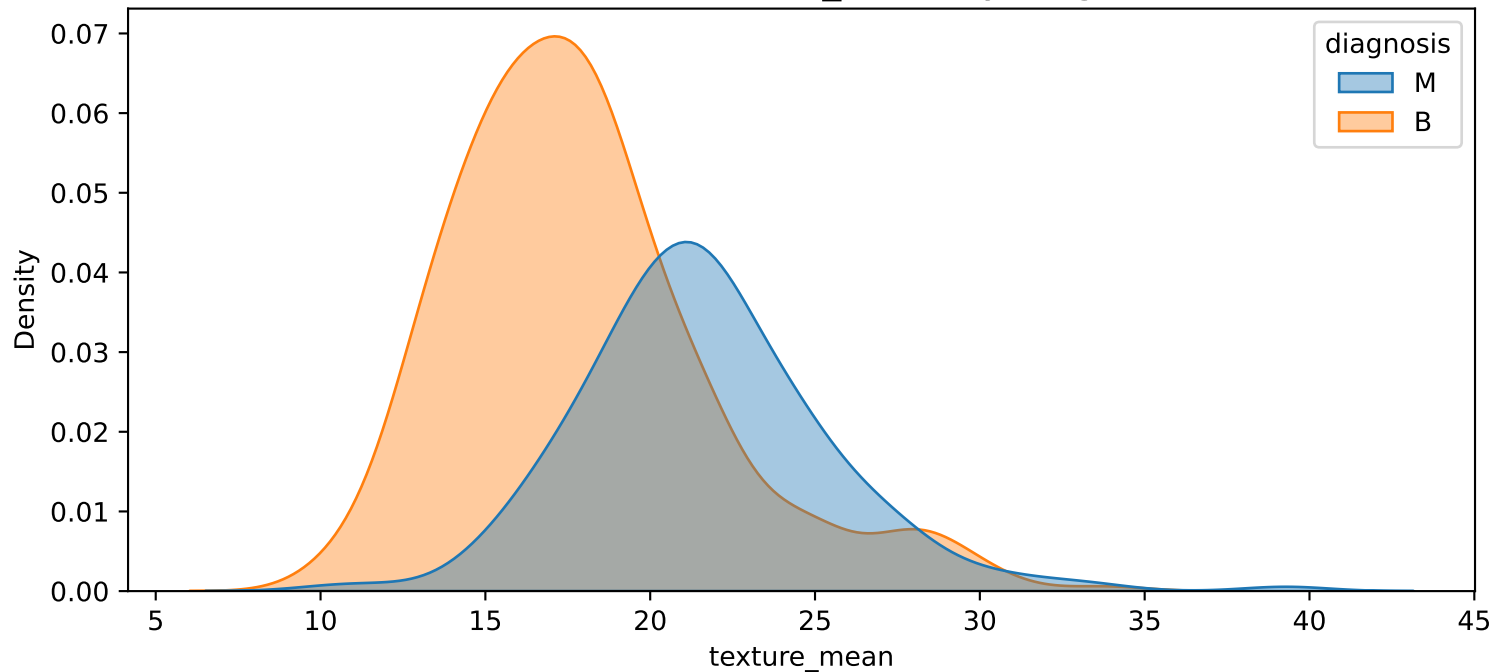
## 2.5 Feature Distribution: Modeling Code

```
# Feature Distribution KDE

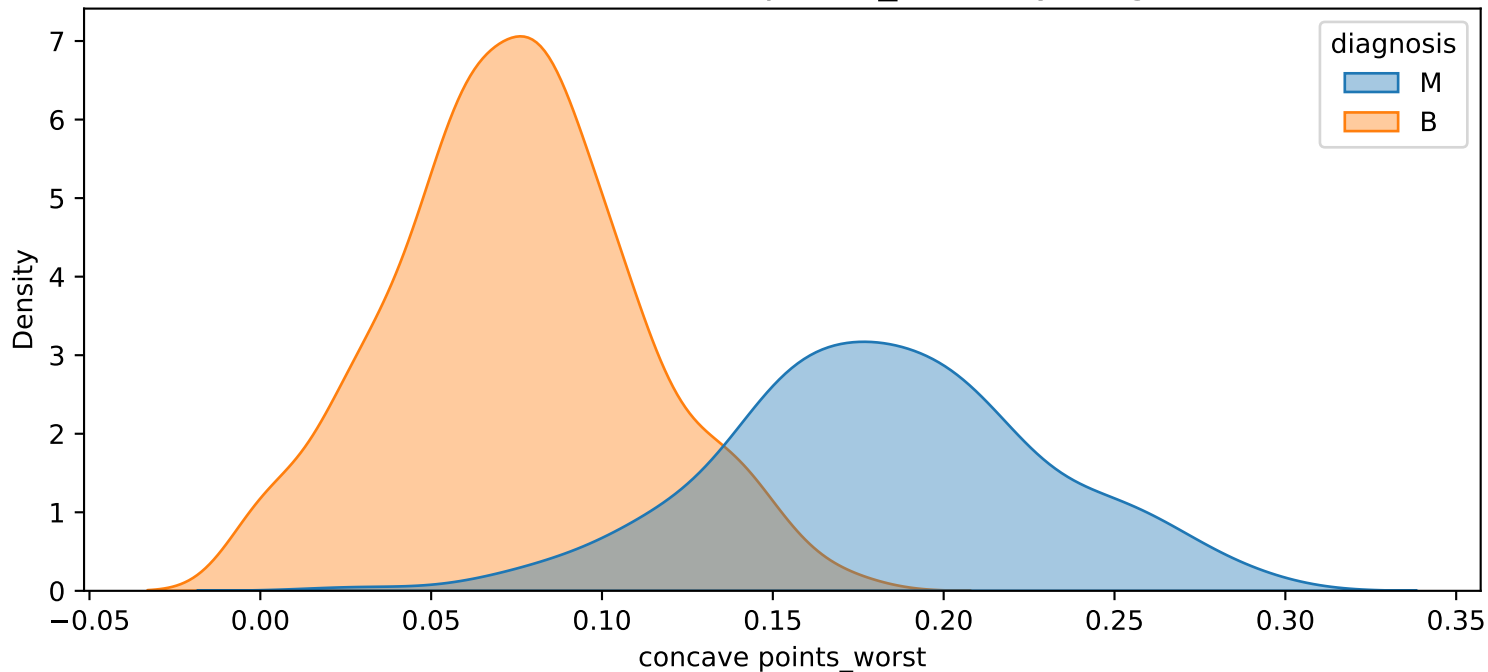
for feat in top_features:
    sns.kdeplot(data=df, x=feat,
                hue=diagnosis_col, fill=True, alpha=0.4)

    plt.title(f'Distribution of {feat} by {diagnosis_col}')
```

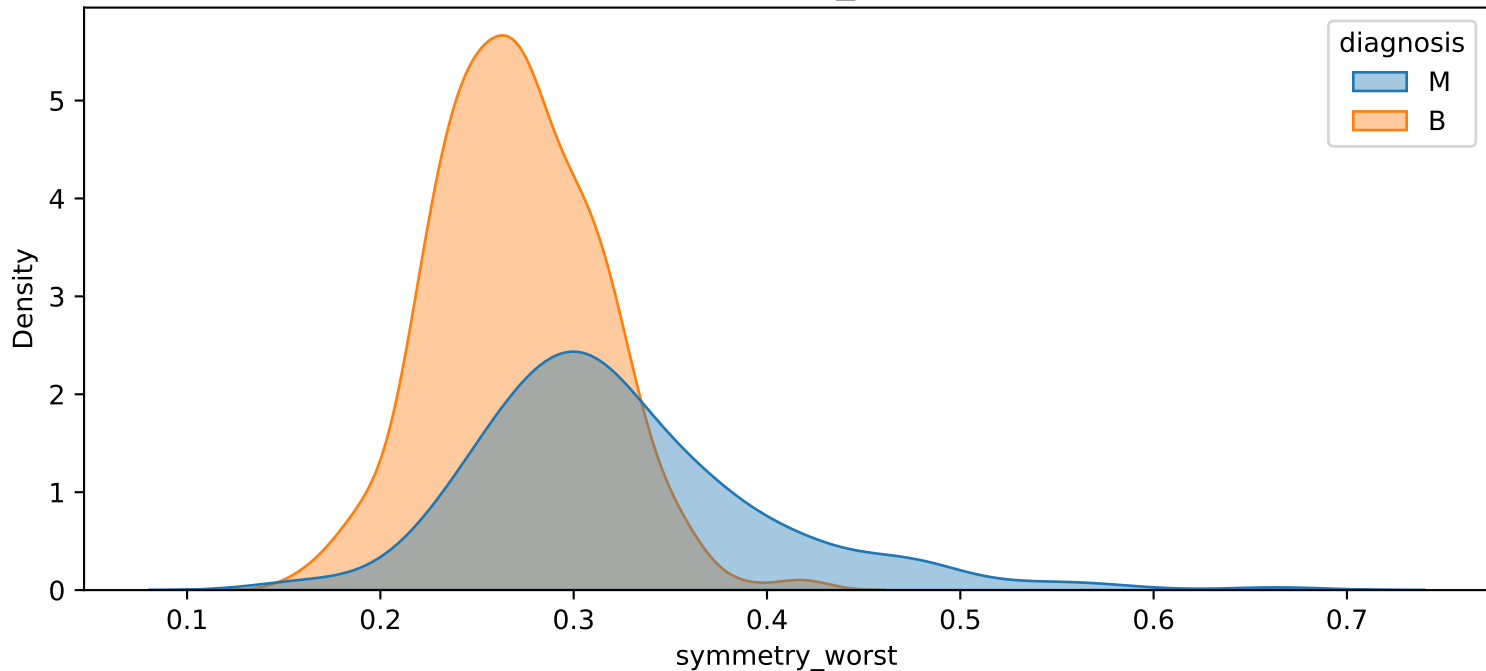
Distribution of texture\_mean by diagnosis



Distribution of concave points\_worst by diagnosis

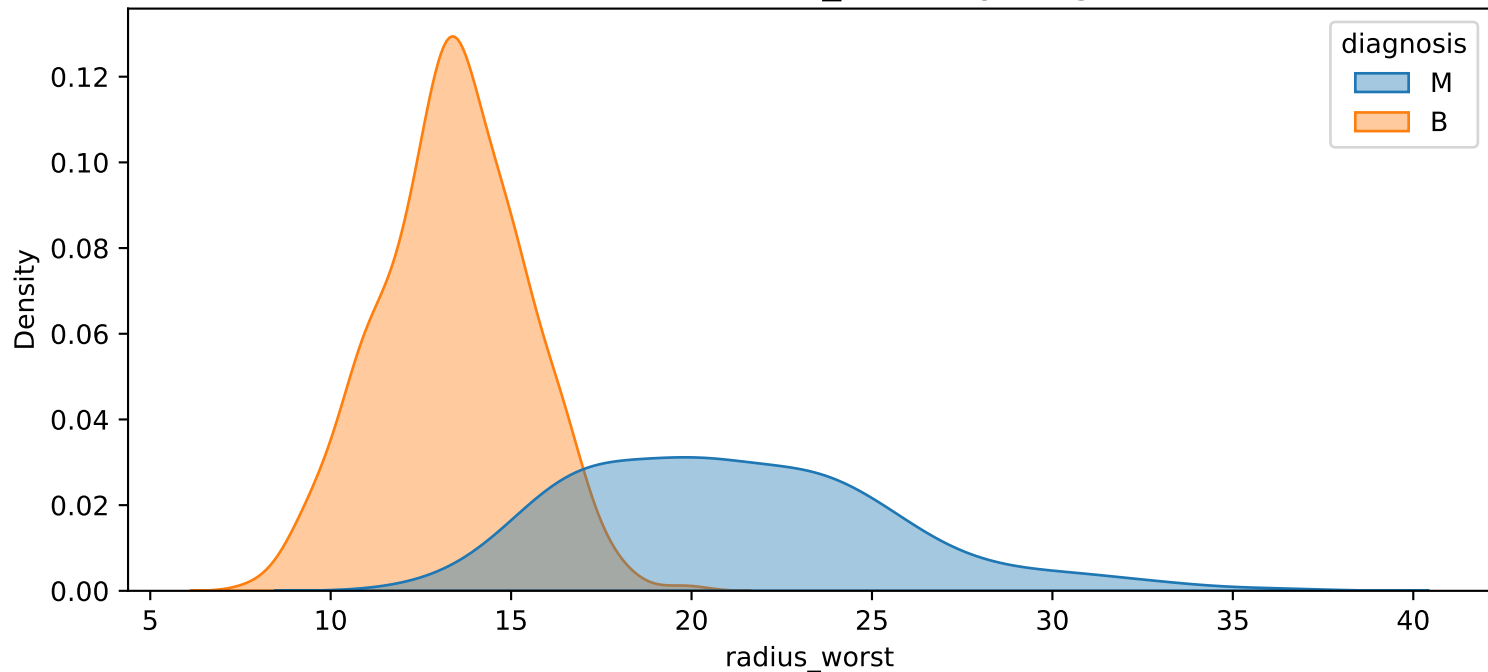


Distribution of symmetry\_worst by diagnosis

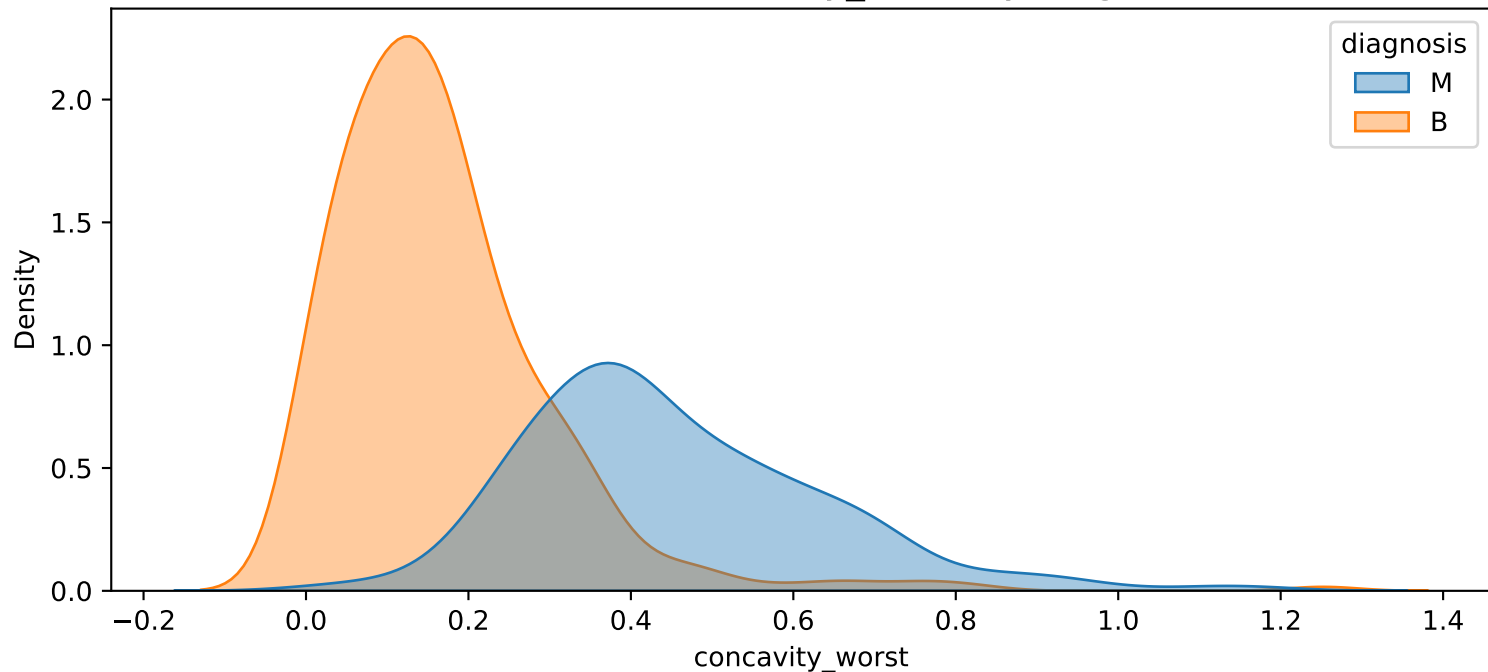




Distribution of radius\_worst by diagnosis



Distribution of concavity\_worst by diagnosis



## 2.1 Decision Tree Visualization (max\_depth=4): Modeling Code

```
import pandas as pd

from src.decision_tree import DecisionTreeClassifier

from src.visualization import plot_tree_graphical


# Load data

df = pd.read_csv('PROJECT2_DATASET.csv')

x = df.drop(['diagnosis'], axis=1)

y = df['diagnosis']


# Train/test split

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)


# Model setup and training

clf = DecisionTreeClassifier(max_depth=4, criterion='entropy')

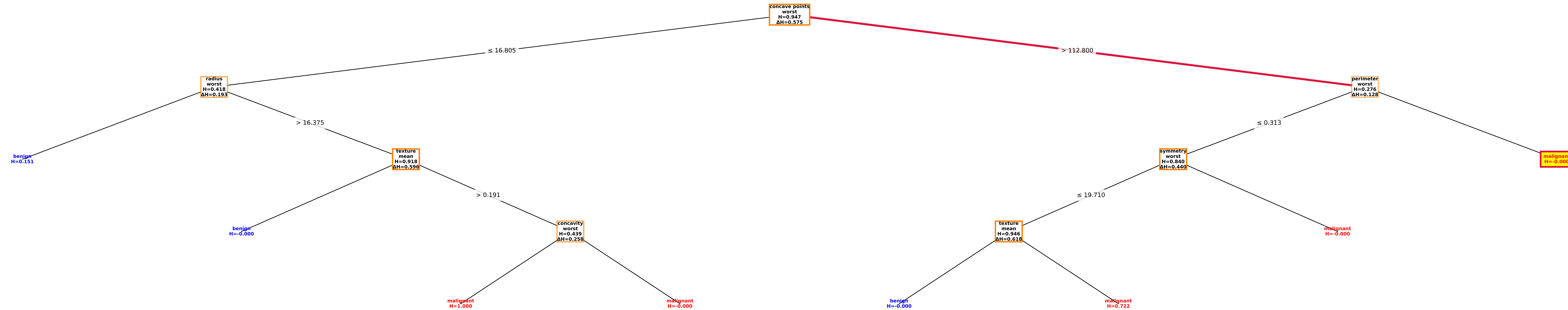
clf.fit(x_train, y_train)


# Prediction

y_pred = clf.predict(x_test)


... (see source file for full code)
```

2.1 Decision Tree Visualization



Summary Statistics (Top 5 Features)

	mean	median	std
texture_mean	19.280	18.835	4.299
concave_points_worst	0.115	0.100	0.066
symmetry_worst	0.290	0.282	0.062
radius_worst	16.281	14.970	4.829
concavity_worst	0.273	0.227	0.208

## 2.1b Decision Tree Visual Key

### Node Colors:

blue

benign leaf

red

malignant leaf

Edged node: predicts malignant

### Highlighted Paths:

Thick crimson line → malignant branches

Path 1: concave points\_worst > threshold → perimeter\_worst > threshold → malignant

Path 2: any direct perimeter\_worst > threshold → malignant

### Node Borders:

border color  $\propto$  entropy (H)    More vivid = higher uncertainty at node

### Node Annotations:

H = entropy

$\Delta H$  = information gain: impurity reduction by this split

Numeric value: split point for feature

### Edge Labels:

$\leq$  threshold: left branch

Samples with feature value  $\leq$  threshold

$>$  threshold: right branch

Samples with feature value  $>$  threshold

## 2.2 Confusion Matrix: Modeling Code

```
# Confusion Matrix

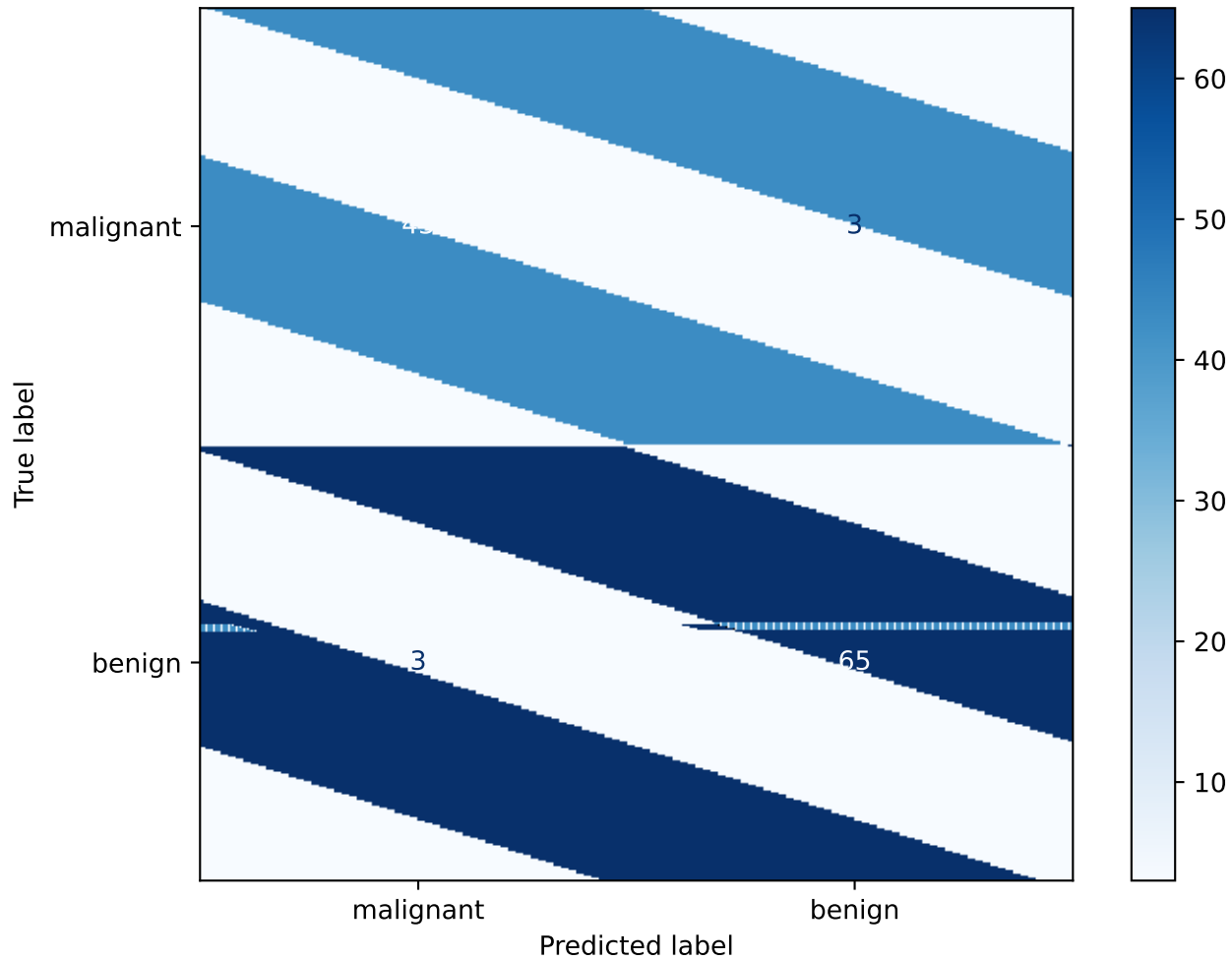
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred, labels=['malignant','benign'])

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['malignant','benign'])

disp.plot(cmap='Blues', colorbar=True)
```

## 2.2 Confusion Matrix





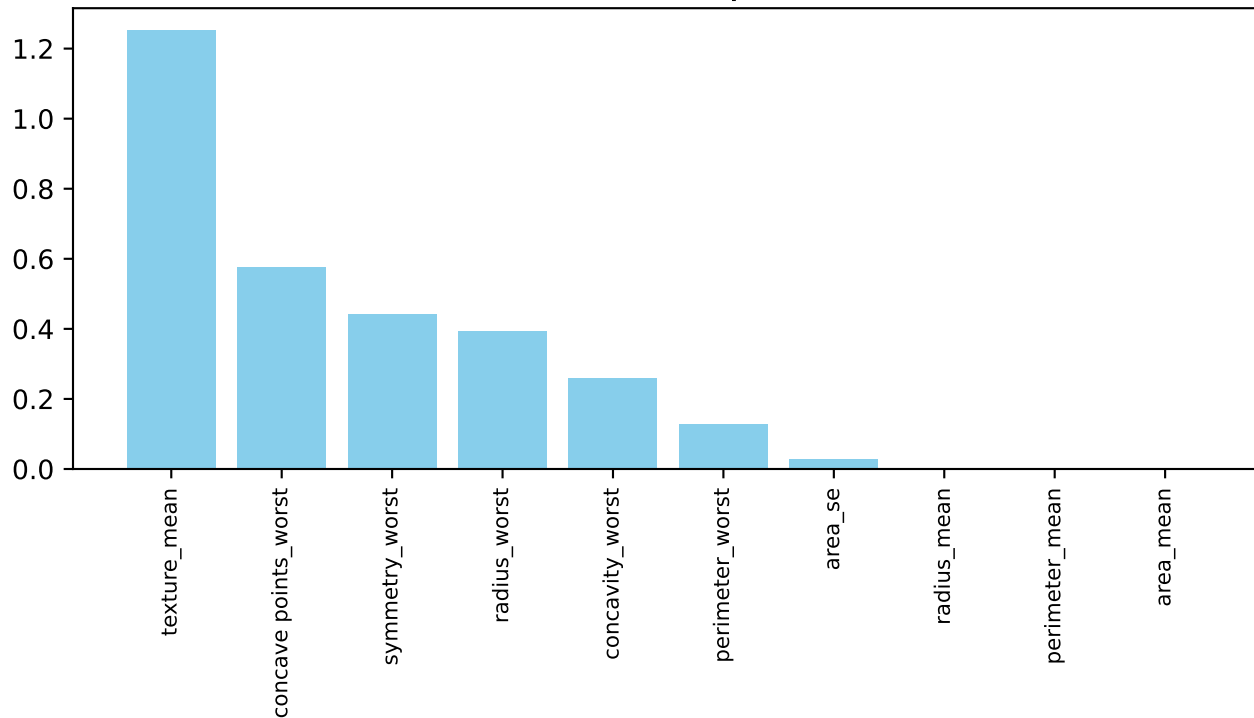
## 2.3 Performance Metrics

Metric	Value
Accuracy	0.947
Precision	0.935
Recall	0.935
F1 Score	0.935

## 2.4 Feature Importance: Modeling Code

```
# Feature Importance Bar Plot
feat_imp = clf.get_feature_importance()
plt.bar(feat_imp.keys(), feat_imp.values(), color='skyblue')
plt.xticks(rotation=90)
```

## 2.4 Feature Importance



### 3. Results Summary

#### **Key Performance Metrics:**

Accuracy: 0.947

Precision: 0.935

Recall: 0.935

F1: 0.935

#### **Top Features:**

- texture\_mean
- concave points\_worst
- symmetry\_worst
- radius\_worst
- concavity\_worst

# Interpretation

In this report, I demonstrate how a decision-tree classifier can predict whether a patient’s breast tumor is malignant or benign.

To build a complete picture, I incorporated three key analyses:

## 1. Confusion Matrix & Recall Focus

- Because missing a malignancy (a false negative) carries the highest risk, I treated recall as my primary performance metric.
- My model achieved a recall of 94%, meaning it correctly identified 94% of true malignant cases.

## 2. Kernel Density Estimates (KDE) & Statistical Summary

- I plotted KDEs for five critical features—texture\_mean, concave\_points\_worst, symmetry\_worst, radius\_worst, and concavity\_worst—overlaying malignant vs. benign distributions.
- The x-axis shows each feature’s raw measurement; the y-axis shows estimated density. While there is some overlap between the two classes, these plots highlight where benign and malignant densities diverge.
- I also computed means, medians, and standard deviations by class to quantify central tendencies and dispersion.

## 3. Decision-Tree Structure & Entropy

- The tree splits are chosen by information gain (entropy reduction). Each node is annotated with its entropy (H) and  $\Delta H$ , and node-border thickness reflects uncertainty (thicker borders = higher entropy).
- In the final pruned tree, the most certain malignant path is just two splits:

1. concave\_points\_worst > 16.80
2. perimeter\_worst > 0.313 → malignant (H = 0)

## 4. Feature Importance & Binary Binning

- Summing information gains across all splits shows that texture\_mean, concave\_points\_worst, and symmetry\_worst are the strongest predictors.
- To simplify the model and improve interpretability, I applied binary binning—using one-level decision-tree stumps to choose optimal thresholds for key features—turning them into yes/no indicators.

## Conclusions

- By focusing on recall, the classifier reliably captures most malignant tumors.
- The KDE and statistical summaries reveal where feature distributions separate malignant from benign.
- The decision-tree visualization, with entropy and  $\Delta H$  annotations, provides clear “if-then” rules.
- Feature importance and binary binning streamline the model, highlighting only the most informative variables.

Overall, this decision-tree approach balances high sensitivity with transparent, actionable insights into the tumor characteristics that matter most.