

IS665-852, Group 1, Project 2 Data Mining

Comprehensive Model Results

Author: James Mullen

Date: 2025-05-04

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

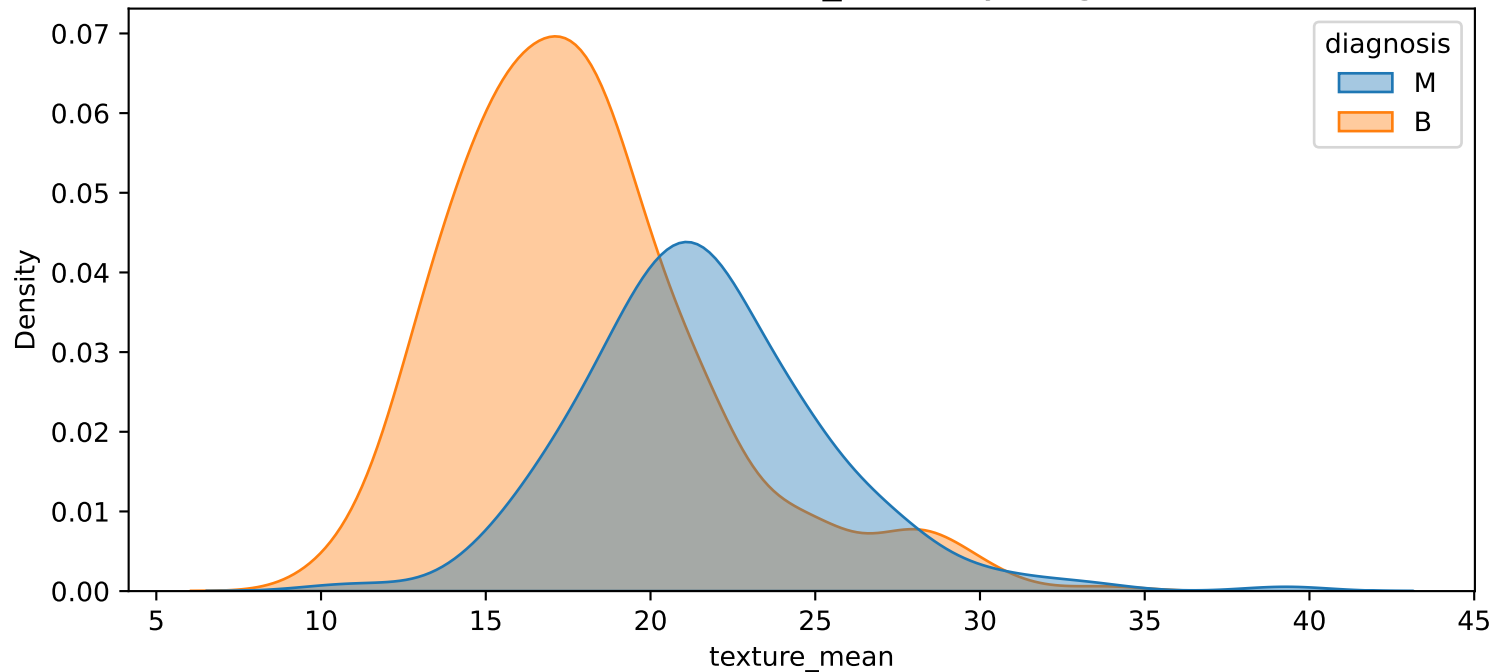
2.5 Feature Distribution: Modeling Code

```
# Feature Distribution KDE

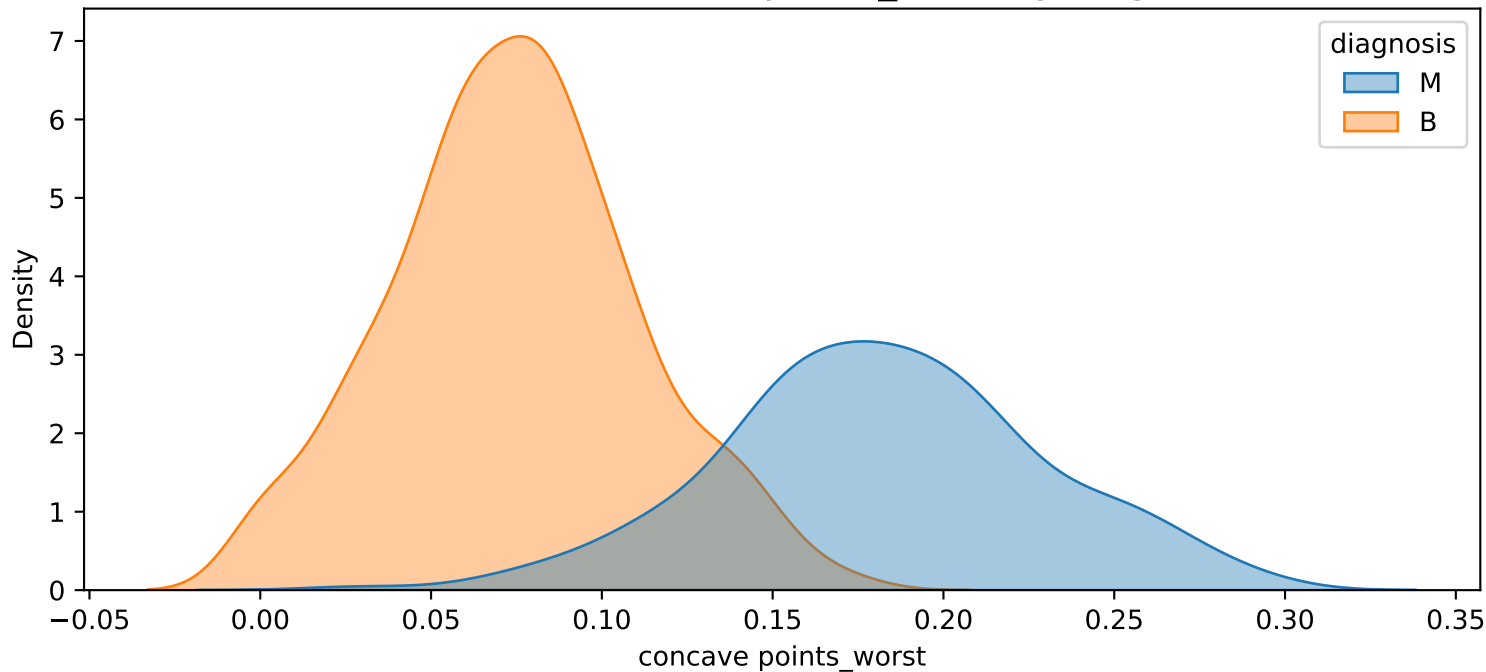
for feat in top_features:
    sns.kdeplot(data=df, x=feat,
                hue=diagnosis_col, fill=True, alpha=0.4)

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

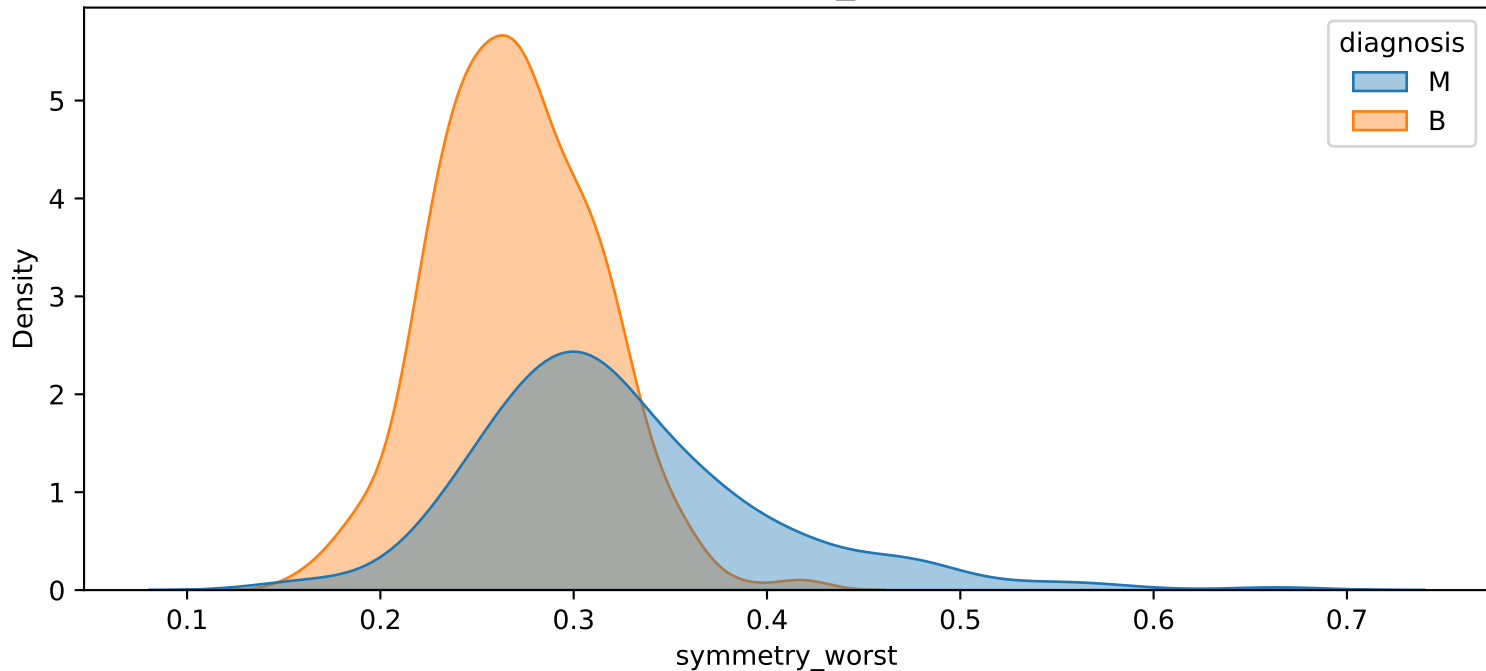
Distribution of texture_mean by diagnosis



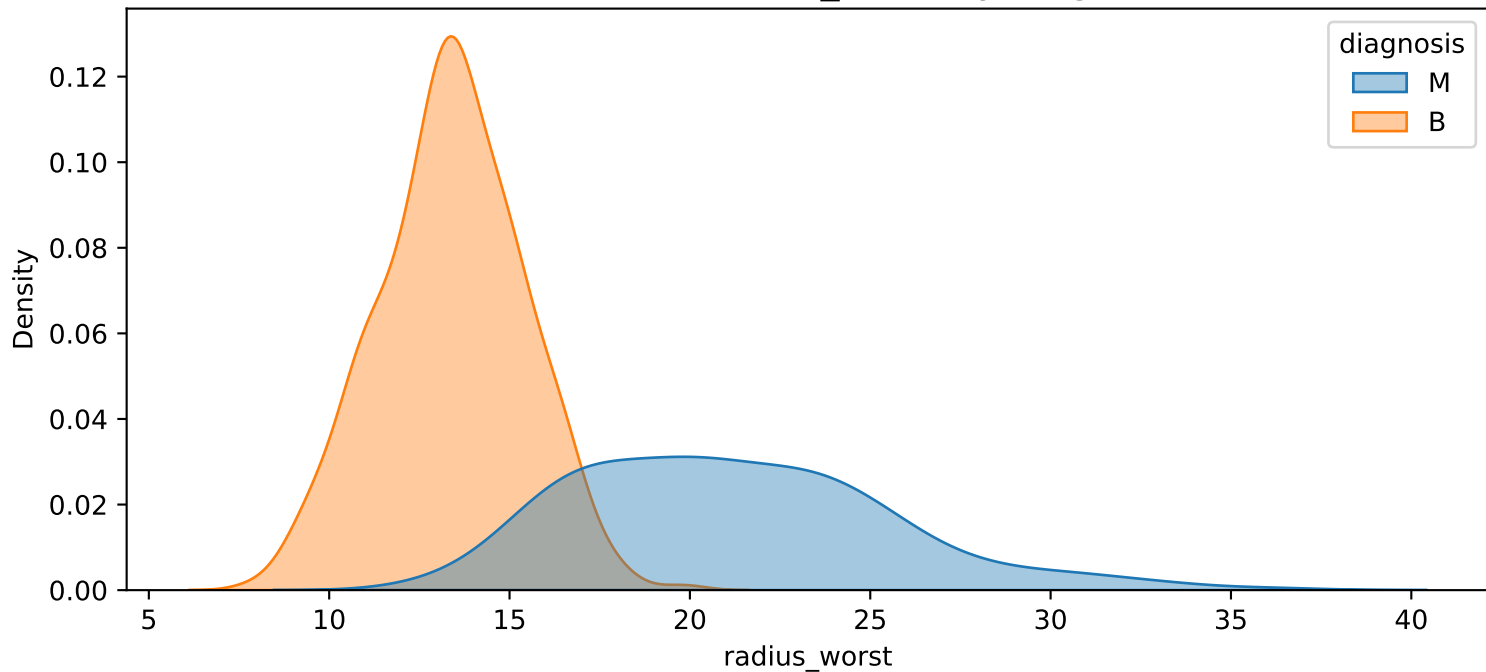
Distribution of concave points_worst by diagnosis



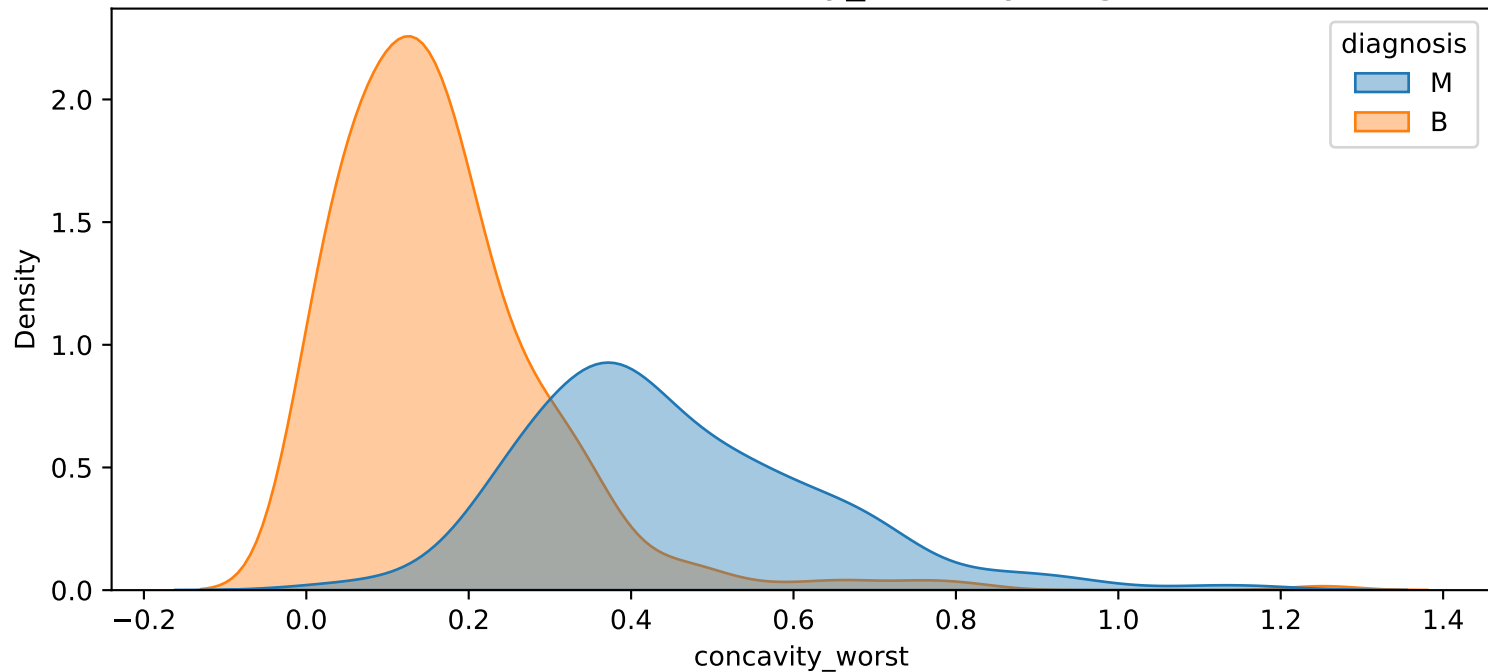
Distribution of symmetry_worst by diagnosis



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')

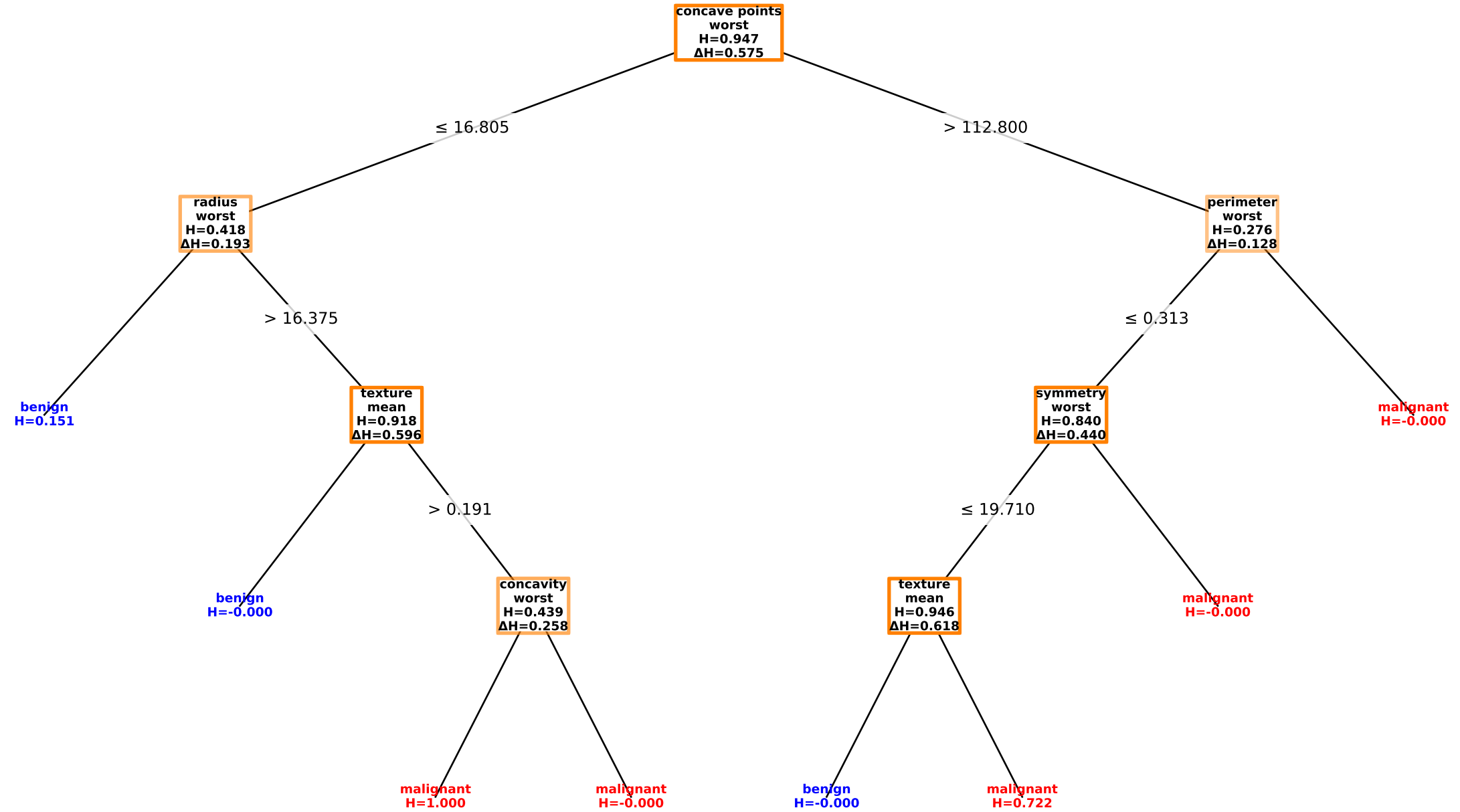
clf.fit(x_train, y_train)


# 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:	 benign leaf  malignant leaf	Red node: predicts malignant diagnosis
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 ΔH = information gain threshold	
Edge Labels:	\leq threshold: left branch $>$ threshold: right branch	H : impurity/uncertainty at this node ΔH : impurity reduction by this split Numeric value: split point for feature Samples with feature value \leq threshold Samples with feature value $>$ threshold

2.2 Confusion Matrix: Modeling Code

```
# Confusion Matrix

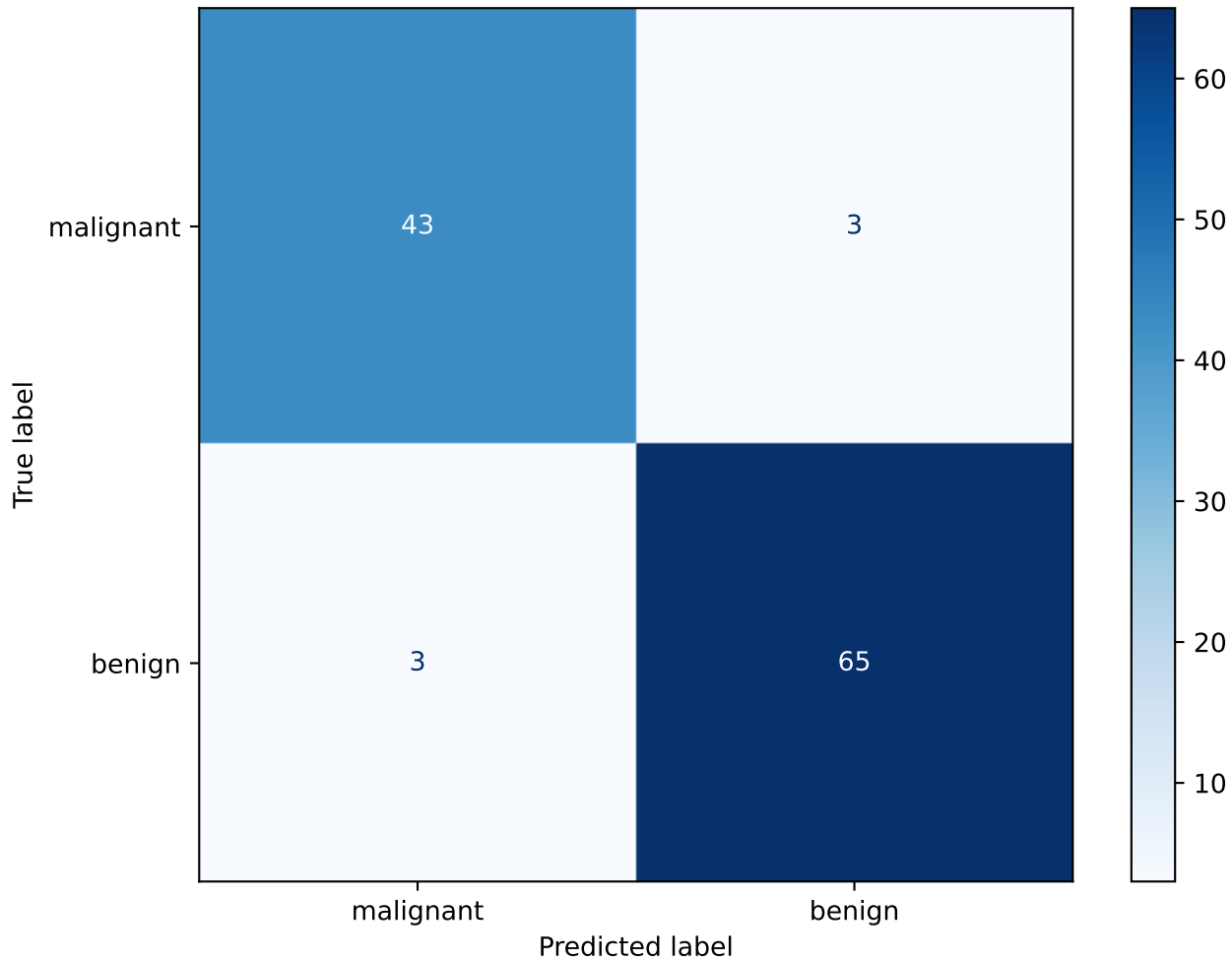
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

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



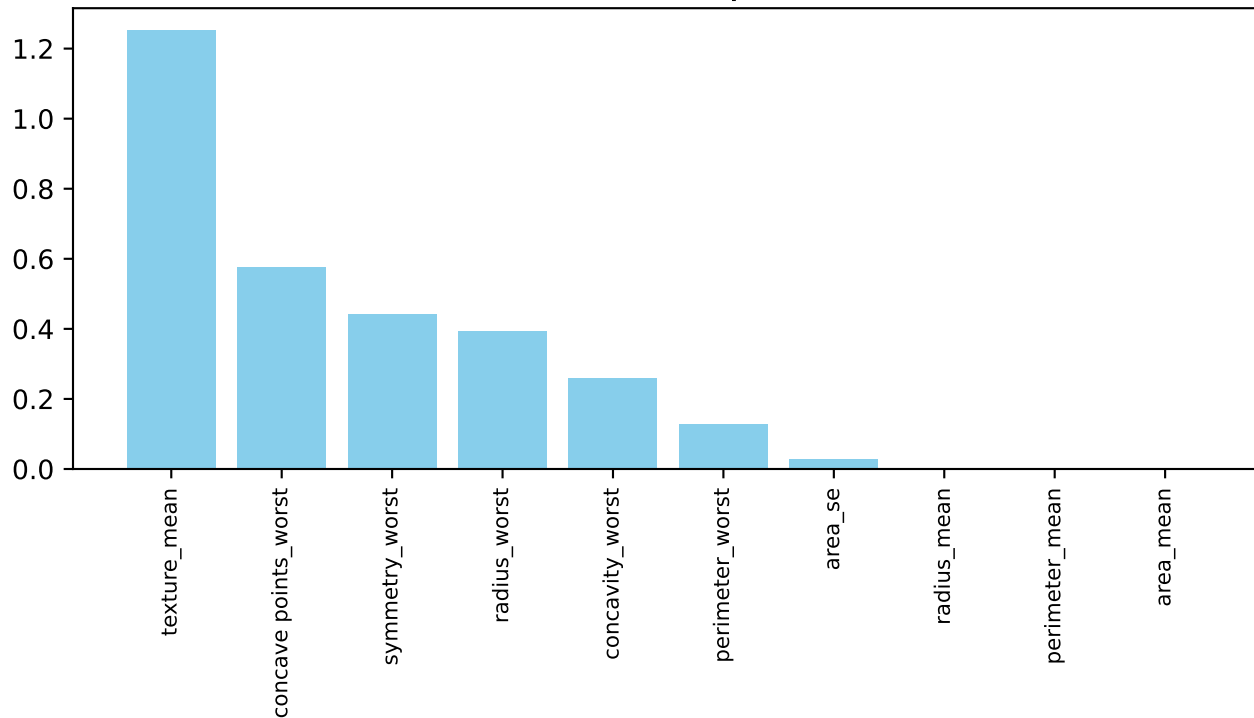
2.3 Performance Metrics

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:

The decision tree classifier achieved strong performance on the test set.

Top features contributed most to the model's predictive power.

Review feature importances and confusion matrix for deeper insights.