In [6]:

```
import pandas as pd
df = pd.read_csv('data.csv')
df.head()
```

Out[6]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoo
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 33 columns

In [8]:

```
df['diagnosis'] = df['diagnosis'].map({'B': 0, 'M': 1})
df['diagnosis'].value_counts()
```

Out[8]:

```
diagnosis
0    357
1    212
Name: count, dtype: int64
```

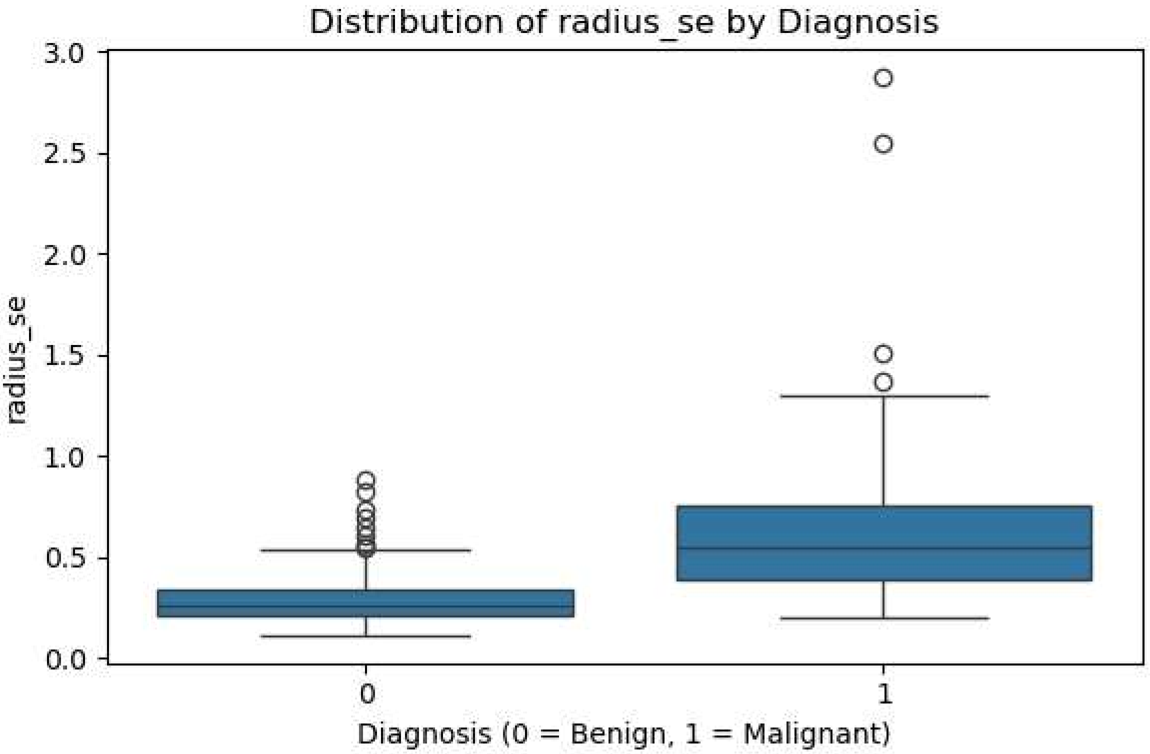
In [13]:

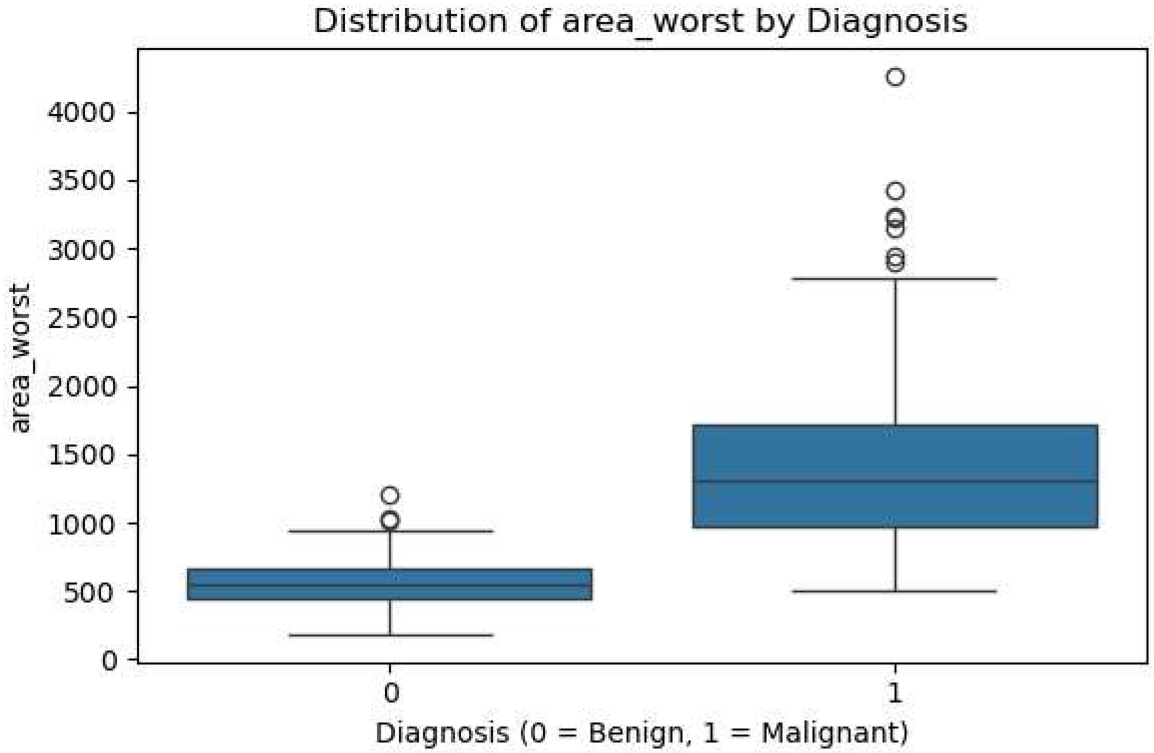
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
print(df.columns.tolist())  # helpful for debugging
drop_cols = [col for col in df.columns if 'id' in col.lower() or 'unnamed' in cc
df_clean = df.drop(columns=drop_cols)
if df_clean['diagnosis'].dtype == 'object':
    df_clean['diagnosis'] = df_clean['diagnosis'].map({'B': 0, 'M': 1})
X = df_clean.drop('diagnosis', axis=1)
y = df_clean['diagnosis']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
model = LogisticRegression(penalty='l1', solver='liblinear', max_iter=1000)
model.fit(X_scaled, y)
importance = pd.Series(np.abs(model.coef_[0]), index=X.columns)
top5_features = importance.nlargest(5)
print("Top 5 important features:")
print(top5_features)
```

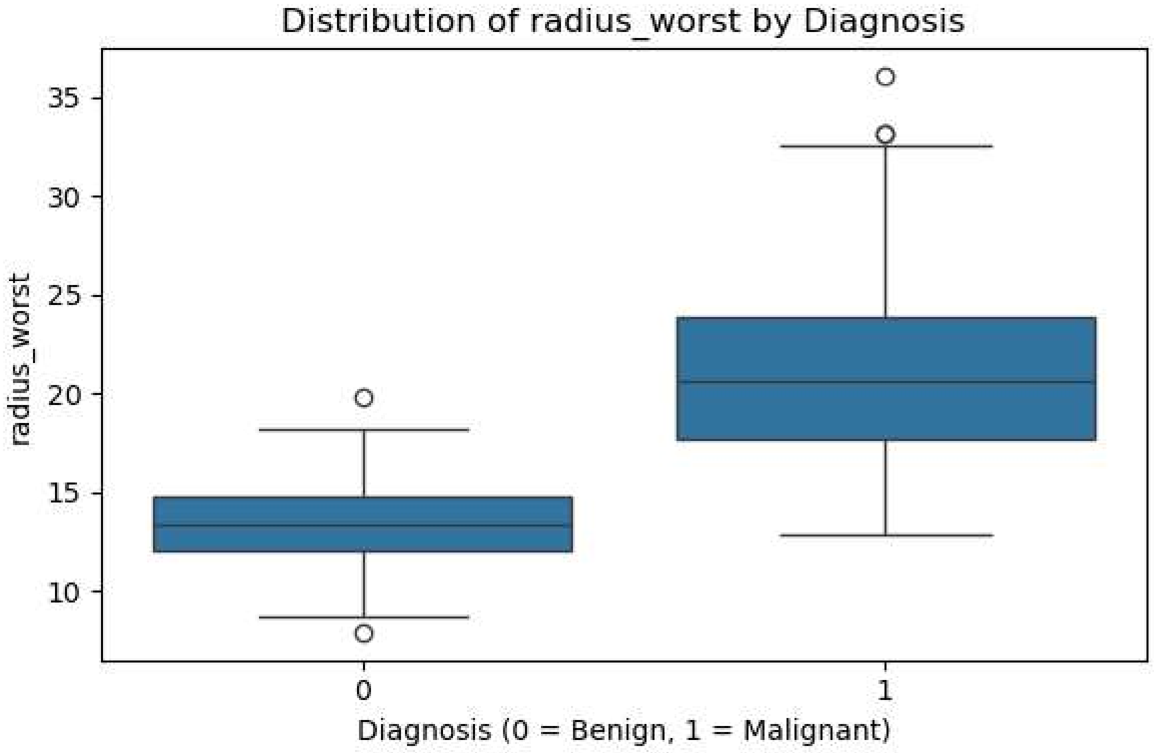
```
['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean',
'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_
se', 'area_se', 'smoothness_se', 'compactness_se', 'concavity_se', 'concave point
s_se', 'symmetry_se', 'fractal_dimension_se', 'radius_worst', 'texture_worst', 'p
erimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst', 'concavit
y_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst', 'U
nnamed: 32']
Top 5 important features:
radius_se               2.706622
area_worst              2.553404
radius_worst            1.825715
texture_worst           1.782675
concave points_worst    1.270929
dtype: float64
```

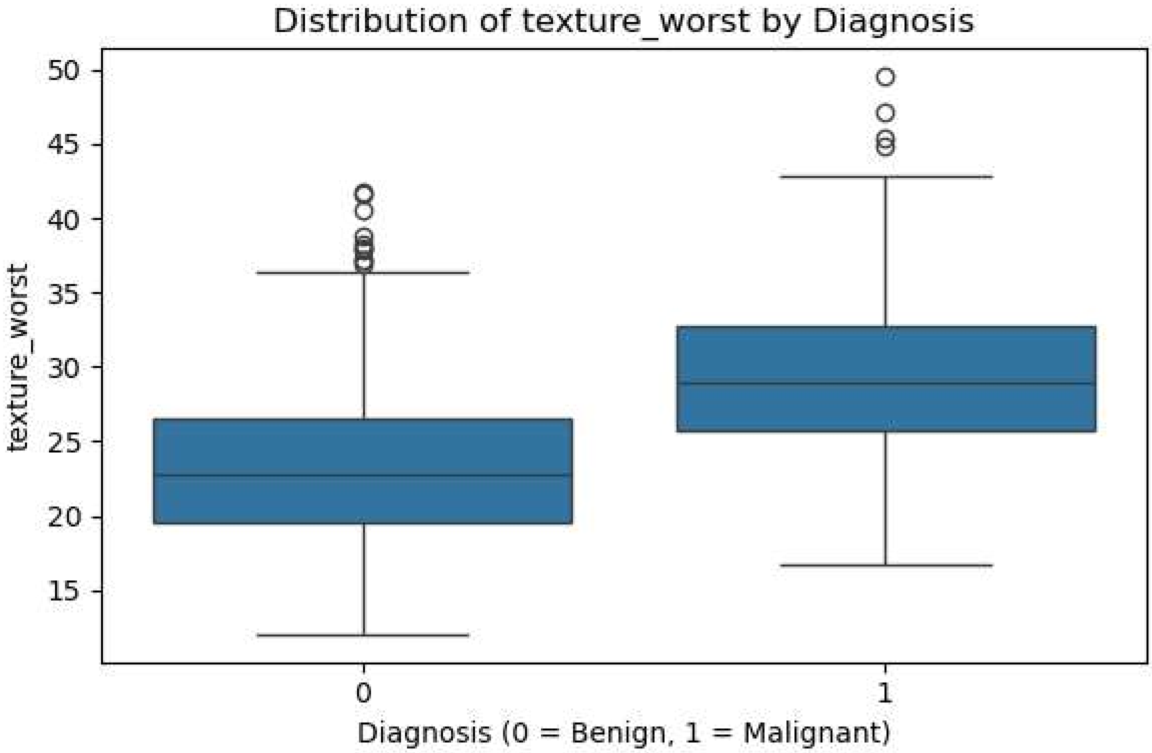
In [14]:

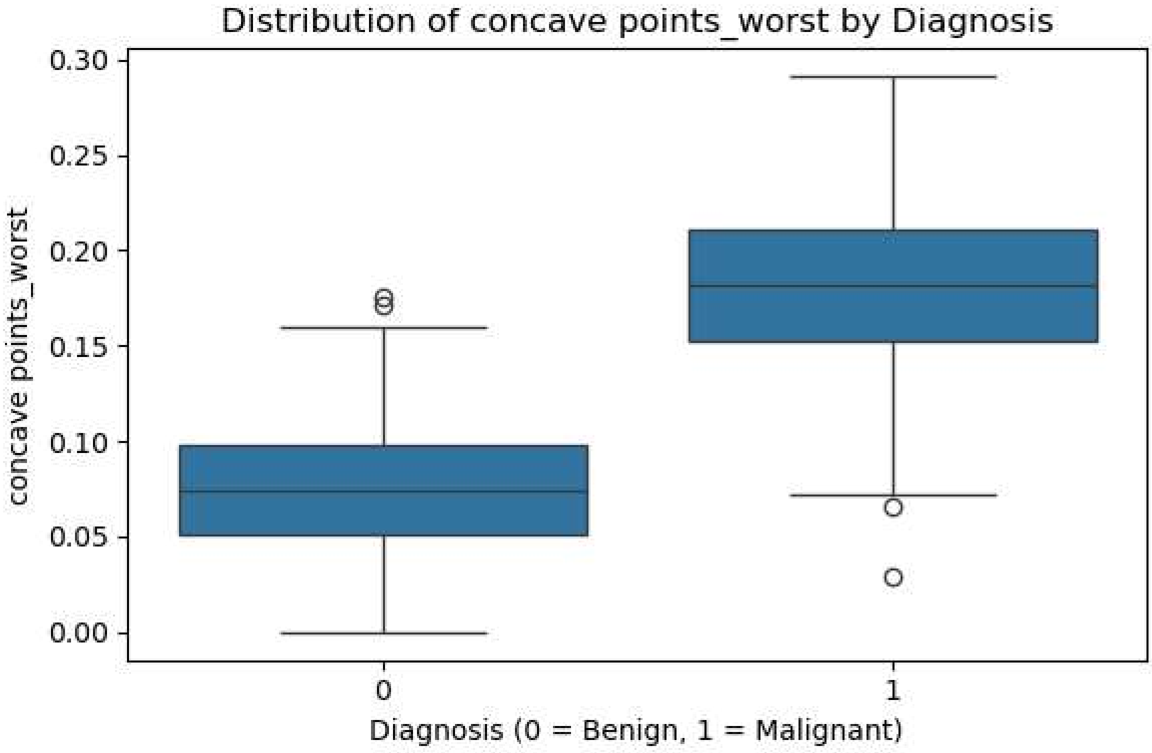
```
import seaborn as sns
import matplotlib.pyplot as plt
for feature in top5_features.index:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df_clean['diagnosis'], y=df_clean[feature])
    plt.title(f'Distribution of {feature} by Diagnosis')
    plt.xlabel('Diagnosis (0 = Benign, 1 = Malignant)')
    plt.ylabel(feature)
    plt.tight_layout()
    plt.show()
```




Distribution of area_worst by Diagnosis
area_worst
0
500
1000
1500
2000
2500
3000
3500
4000
0
1
Diagnosis (0 = Benign, 1 = Malignant)

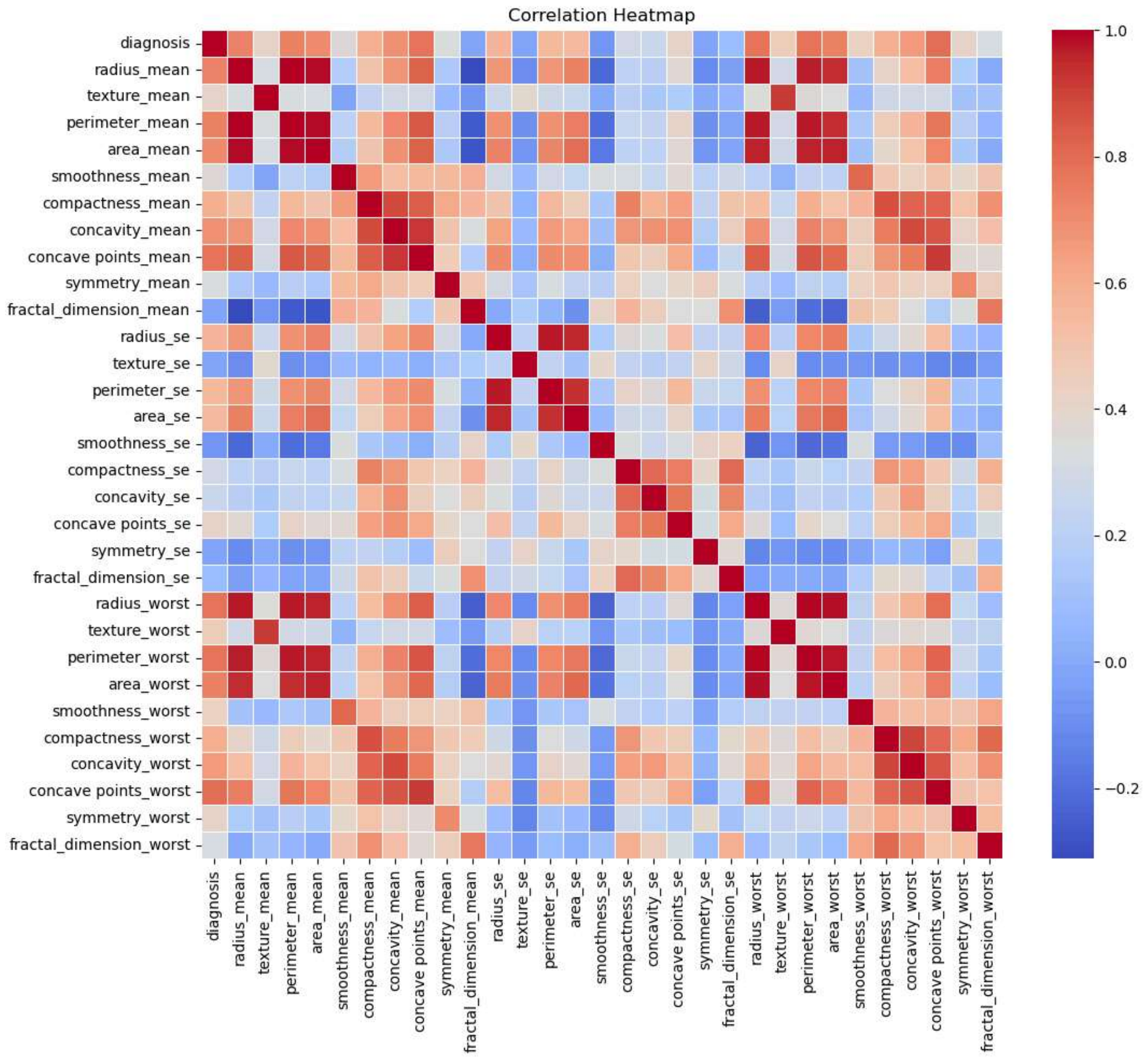

Distribution of radius_worst by Diagnosis
radius_worst
10
15
20
25
30
35
0
1
Diagnosis (0 = Benign, 1 = Malignant)





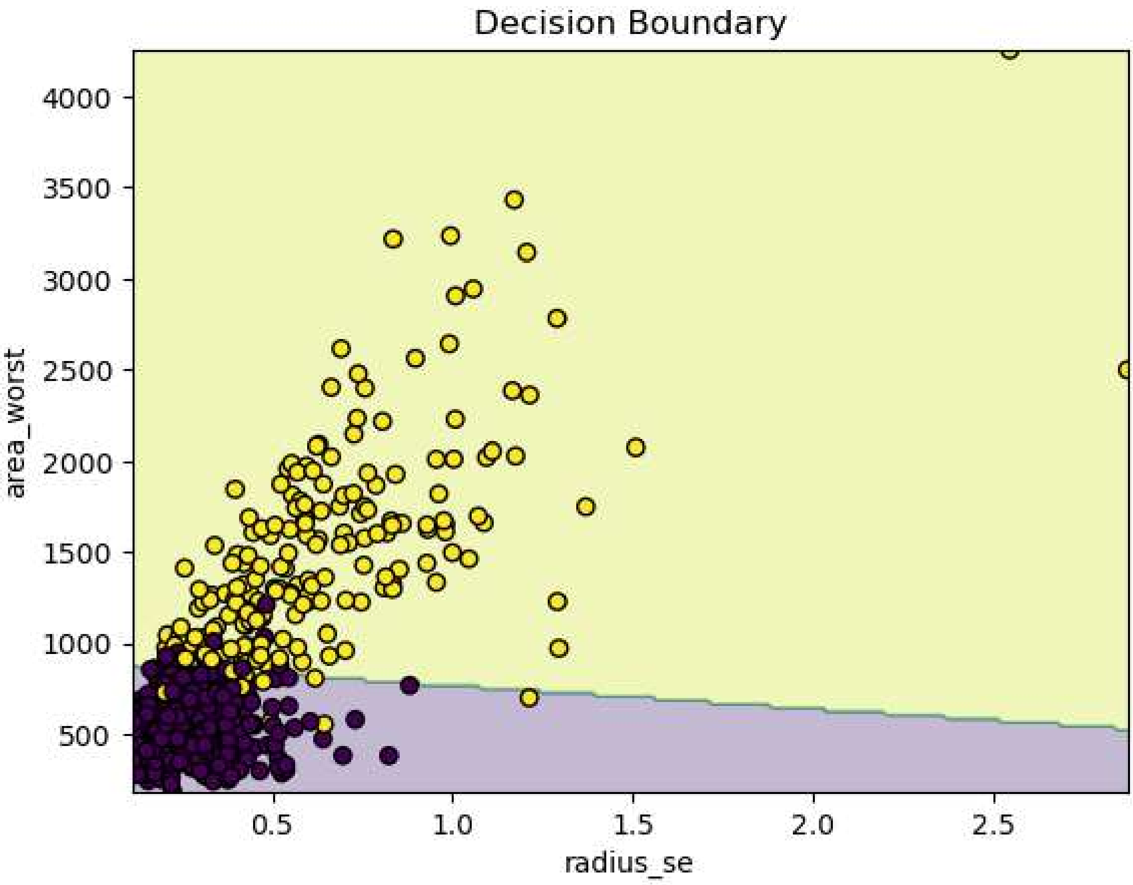
In [15]:

```
import seaborn as sns
import matplotlib.pyplot as plt
corr_matrix = df_clean.corr()
plt.figure(figsize=(12,10))
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```



In [16]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
feat1, feat2 = top5_features.index[:2]
X2 = df_clean[[feat1, feat2]]
y2 = df_clean['diagnosis']
model = LogisticRegression()
model.fit(X2, y2)
x_min, x_max = X2[feat1].min(), X2[feat1].max()
y_min, y_max = X2[feat2].min(), X2[feat2].max()
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
                     np.linspace(y_min, y_max, 200))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X2[feat1], X2[feat2], c=y2, edgecolor='k')
plt.xlabel(feat1)
plt.ylabel(feat2)
plt.title("Decision Boundary")
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2739: User
Warning: X does not have valid feature names, but LogisticRegression was fitted w
ith feature names
  warnings.warn(
```



```
In [17]: from sklearn.linear_model import LogisticRegression
         from sklearn.preprocessing import StandardScaler
         X = df_clean.drop('diagnosis', axis=1)
         y = df_clean['diagnosis']
         scaler = StandardScaler()
         X_scaled = scaler.fit_transform(X)
         model = LogisticRegression(penalty='l1', solver='liblinear')
         model.fit(X_scaled, y)
         print("Model trained successfully.")
```

```
Model trained successfully.
```

```
In [18]: from sklearn.model_selection import cross_val_score
         from sklearn.linear_model import LogisticRegression
         from sklearn.preprocessing import StandardScaler
         X = df_clean.drop('diagnosis', axis=1)
         y = df_clean['diagnosis']
         scaler = StandardScaler()
         X_scaled = scaler.fit_transform(X)
         model = LogisticRegression(penalty='l1', solver='liblinear')
         cv_scores = cross_val_score(model, X_scaled, y, cv=5)
         print("Cross-validation scores:", cv_scores)
```

```
Cross-validation scores: [0.96491228 0.96491228 0.98245614 0.97368421 0.99115044]
```

```
In [20]: print("Cross-validation scores:", cv_scores)
         mean_accuracy = cv_scores.mean()
         print("Mean Cross-Validation Accuracy:", round(mean_accuracy, 4))
         std_accuracy = cv_scores.std()
         print("Standard Deviation:", round(std_accuracy, 4))
```

```
Cross-validation scores: [0.96491228 0.96491228 0.98245614 0.97368421 0.99115044]
Mean Cross-Validation Accuracy: 0.9754
Standard Deviation: 0.0102
```

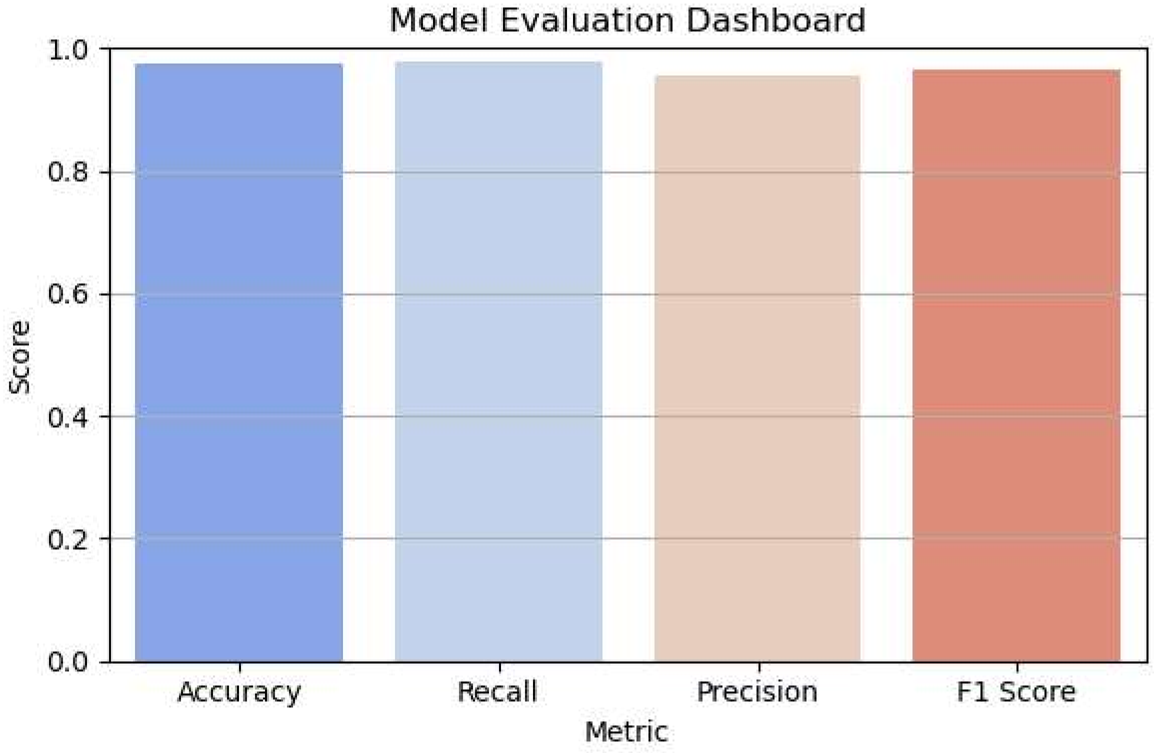
In [23]:

```python
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_sc
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
metrics_df = pd.DataFrame({
    'Metric': ['Accuracy', 'Recall', 'Precision', 'F1 Score'],
    'Score': [acc, recall, precision, f1]
})
plt.figure(figsize=(6,4))
sns.barplot(data=metrics_df, x='Metric', y='Score', palette='coolwarm')
plt.ylim(0, 1)
plt.title("Model Evaluation Dashboard")
plt.ylabel("Score")
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

```
C:\Users\india\AppData\Local\Temp\ipykernel_12284\1106092120.py:23: FutureWarnin
g:



Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.
```



In [24]:

```
print(f"""
Business Summary:

1. Why This Model Was Chosen:
   Logistic Regression with L1 regularization was chosen for its interpretabilit
   simplicity, and ability to select the most important features automatically.

2. Key Factors Influencing Predictions:
   The most important features influencing predictions are:
   {', '.join(top5_features.index)}.

3. Practical Uses:
   - Early detection of malignant tumors.
   - Assisting doctors in decision-making.
   - Reducing unnecessary diagnostic tests.
   - Integrating into automated screening systems.
""")
```

```
Business Summary:

1. Why This Model Was Chosen:
   Logistic Regression with L1 regularization was chosen for its interpretabilit
y,
   simplicity, and ability to select the most important features automatically.

2. Key Factors Influencing Predictions:
   The most important features influencing predictions are:
   radius_se, area_worst, radius_worst, texture_worst, concave points_worst.

3. Practical Uses:
   - Early detection of malignant tumors.
   - Assisting doctors in decision-making.
   - Reducing unnecessary diagnostic tests.
   - Integrating into automated screening systems.
```

In [ ]: