```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import pprint

from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
import warnings # current version of seaborn generates a bunch of
warnings that we'll ignore
warnings.filterwarnings("ignore")

pd.read_csv("breast-cancer.csv")
```

```
           id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
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
..        ...       ...          ...           ...             ...
...
564    926424         M        21.56         22.39          142.00
1479.0
565    926682         M        20.13         28.25          131.20
1261.0
566    926954         M        16.60         28.08          108.30
858.1
567    927241         M        20.60         29.33          140.10
1265.0
568     92751         B         7.76         24.54           47.92
181.0

     smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0            0.11840           0.27760         0.30010
0.14710
1            0.08474           0.07864         0.08690
```

```
0.07017
2          0.10960          0.15990         0.19740
0.12790
3          0.14250          0.28390         0.24140
0.10520
4          0.10030          0.13280         0.19800
0.10430
..             ...              ...             ...
...
564        0.11100          0.11590         0.24390
0.13890
565        0.09780          0.10340         0.14400
0.09791
566        0.08455          0.10230         0.09251
0.05302
567        0.11780          0.27700         0.35140
0.15200
568        0.05263          0.04362         0.00000
0.00000

     ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0    ...        25.380          17.33           184.60      2019.0
1    ...        24.990          23.41           158.80      1956.0
2    ...        23.570          25.53           152.50      1709.0
3    ...        14.910          26.50            98.87       567.7
4    ...        22.540          16.67           152.20      1575.0
..   ...           ...            ...              ...         ...
564  ...        25.450          26.40           166.10      2027.0
565  ...        23.690          38.25           155.00      1731.0
566  ...        18.980          34.12           126.70      1124.0
567  ...        25.740          39.42           184.60      1821.0
568  ...         9.456          30.37            59.16       268.6

     smoothness_worst  compactness_worst  concavity_worst  \
0             0.16220            0.66560           0.7119
1             0.12380            0.18660           0.2416
2             0.14440            0.42450           0.4504
3             0.20980            0.86630           0.6869
4             0.13740            0.20500           0.4000
..                ...                ...              ...
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     concave points_worst  symmetry_worst  fractal_dimension_worst
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
2                  0.2430          0.3613                  0.08758
```

```
3                     0.2575          0.6638                   0.17300
4                     0.1625          0.2364                   0.07678
..                       ...             ...                       ...
564                   0.2216          0.2060                   0.07115
565                   0.1628          0.2572                   0.06637
566                   0.1418          0.2218                   0.07820
567                   0.2650          0.4087                   0.12400
568                   0.0000          0.2871                   0.07039

[569 rows x 32 columns]
```

```
df = pd.read_csv("breast-cancer.csv")
print("Dataset Shape:", df.shape)
print(df.head())
```

```
Dataset Shape: (569, 32)
         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
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

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  ...         25.38          17.33           184.60      2019.0
1  ...         24.99          23.41           158.80      1956.0
2  ...         23.57          25.53           152.50      1709.0
3  ...         14.91          26.50            98.87       567.7
4  ...         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave
```

```
points_worst  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

```
print("\nSummary Statistics:\n", df.describe())
```

```

Summary Statistics:
                  id  radius_mean  texture_mean  perimeter_mean
area_mean  \
count  5.690000e+02   569.000000    569.000000      569.000000
569.000000
mean   3.037183e+07    14.127292     19.289649       91.969033
654.889104
std    1.250206e+08     3.524049      4.301036       24.298981
351.914129
min    8.670000e+03     6.981000      9.710000       43.790000
143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000
420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000
551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000
782.700000
max    9.113205e+08    28.110000     39.280000      188.500000
2501.000000

       smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
count       569.000000        569.000000      569.000000
569.000000
mean          0.096360          0.104341        0.088799
0.048919
```

```
std           0.014064          0.052813        0.079720
0.038803
min           0.052630          0.019380        0.000000
0.000000
25%           0.086370          0.064920        0.029560
0.020310
50%           0.095870          0.092630        0.061540
0.033500
75%           0.105300          0.130400        0.130700
0.074000
max           0.163400          0.345400        0.426800
0.201200

       symmetry_mean  ...  radius_worst  texture_worst
perimeter_worst  \
count     569.000000  ...    569.000000     569.000000
569.000000
mean        0.181162  ...     16.269190      25.677223
107.261213
std         0.027414  ...      4.833242       6.146258
33.602542
min         0.106000  ...      7.930000      12.020000
50.410000
25%         0.161900  ...     13.010000      21.080000
84.110000
50%         0.179200  ...     14.970000      25.410000
97.660000
75%         0.195700  ...     18.790000      29.720000
125.400000
max         0.304000  ...     36.040000      49.540000
251.200000

        area_worst  smoothness_worst  compactness_worst
concavity_worst  \
count   569.000000        569.000000         569.000000
569.000000
mean    880.583128          0.132369           0.254265
0.272188
std     569.356993          0.022832           0.157336
0.208624
min     185.200000          0.071170           0.027290
0.000000
25%     515.300000          0.116600           0.147200
0.114500
50%     686.500000          0.131300           0.211900
0.226700
75%    1084.000000          0.146000           0.339100
0.382900
max    4254.000000          0.222600           1.058000
```

```
1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

[8 rows x 31 columns]
```

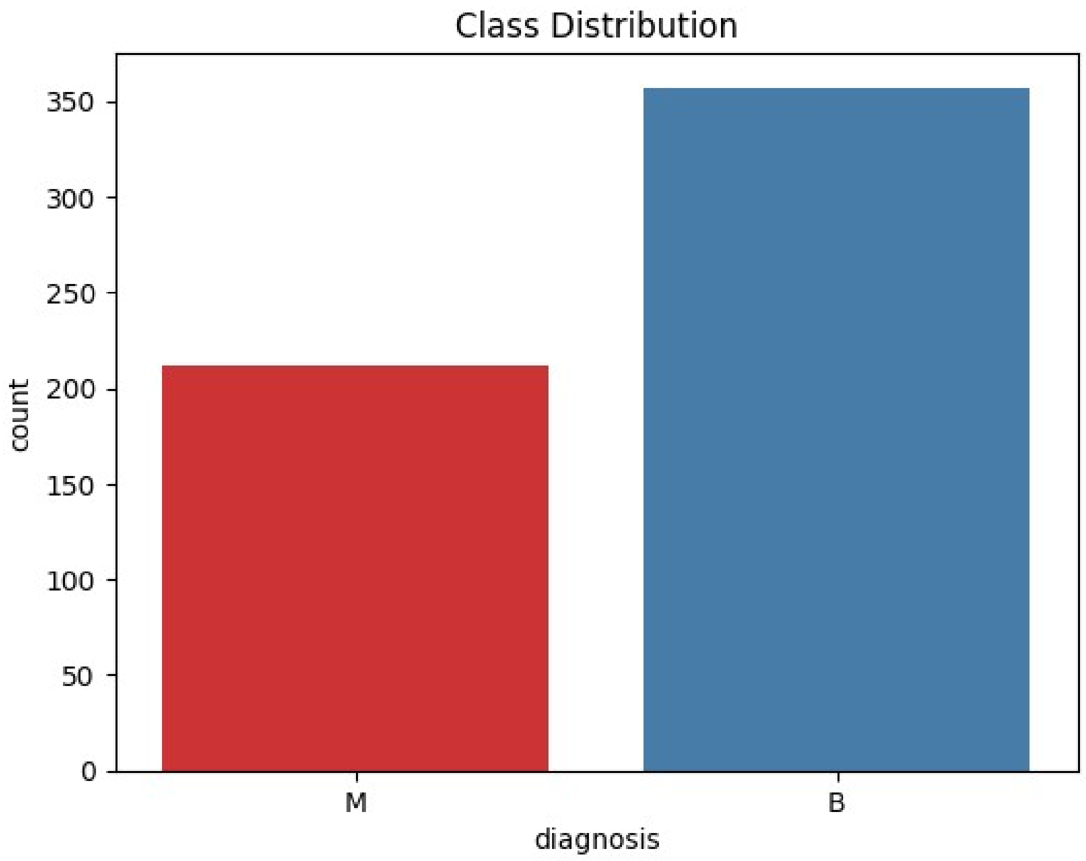
```
print("\nMissing Values:\n", df.isnull().sum())
```

```
Missing Values:
 id                           0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
```

```
fractal_dimension_worst    0
dtype: int64
```

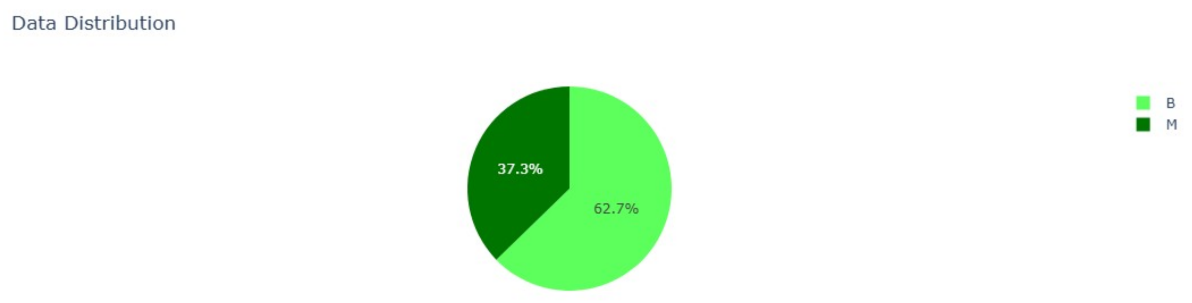
Visualization

```
sns.countplot(x='diagnosis', data=df, palette="Set1")
plt.title("Class Distribution")
plt.show()

print(df['diagnosis'].value_counts(normalize=True))
```

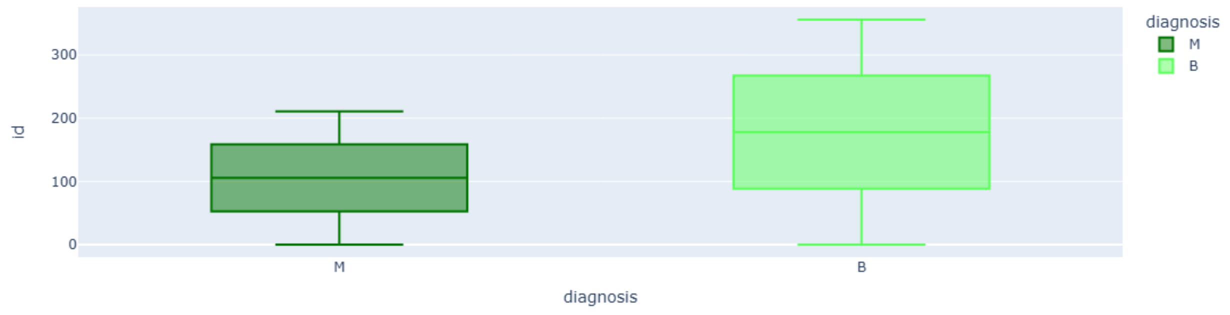


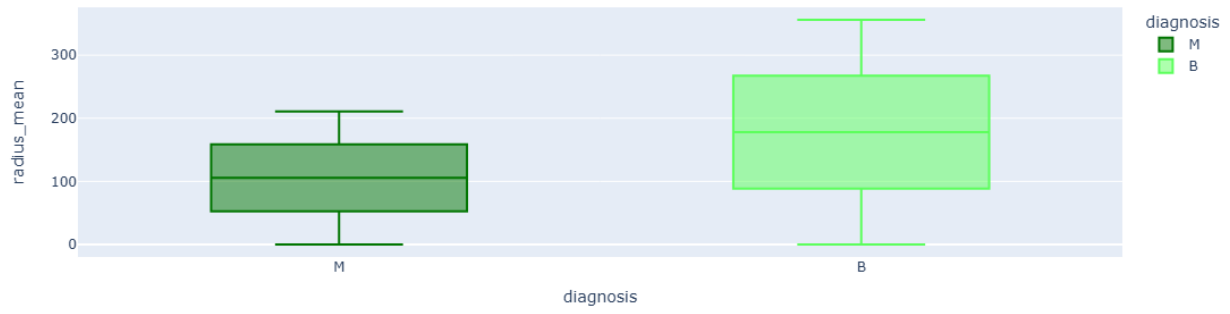
```
diagnosis
B    0.627417
M    0.372583
Name: proportion, dtype: float64
```

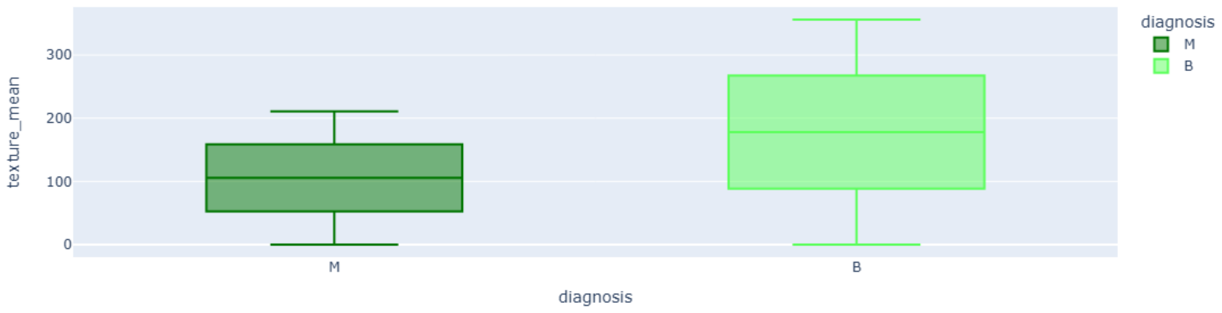
```
px.pie(df, 'diagnosis',
color='diagnosis',color_discrete_sequence=['#007500','#5CFF5C'],title=
'Data Distribution')
```

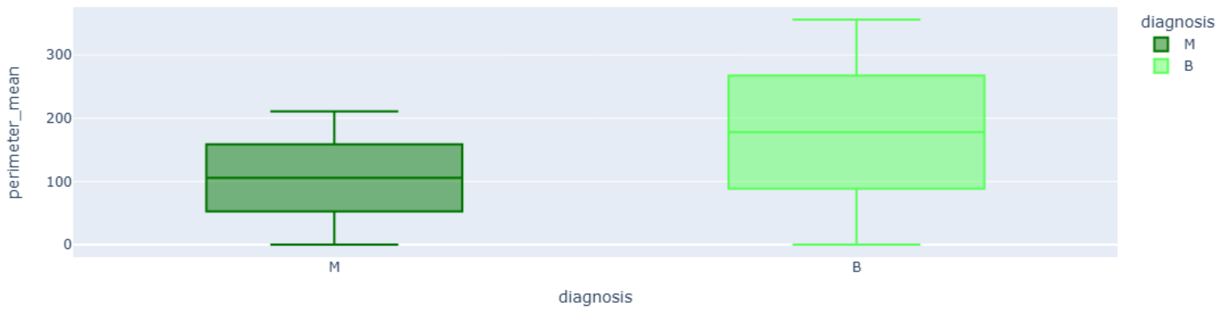


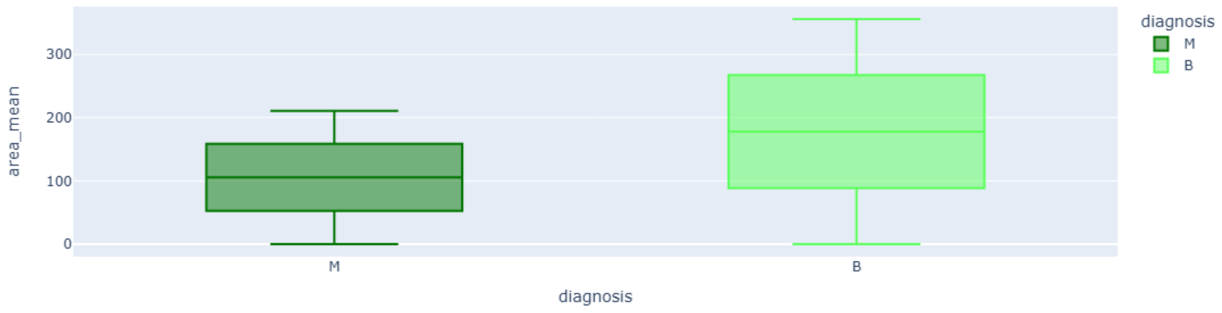
```
for column in  df.drop('diagnosis',axis=1).columns[:5]:
    fig = 
px.box(data_frame=df,x='diagnosis',color='diagnosis',y=column,color_di
screte_sequence=['#007500','#5CFF5C'],orientation='v')
    fig.show()
```



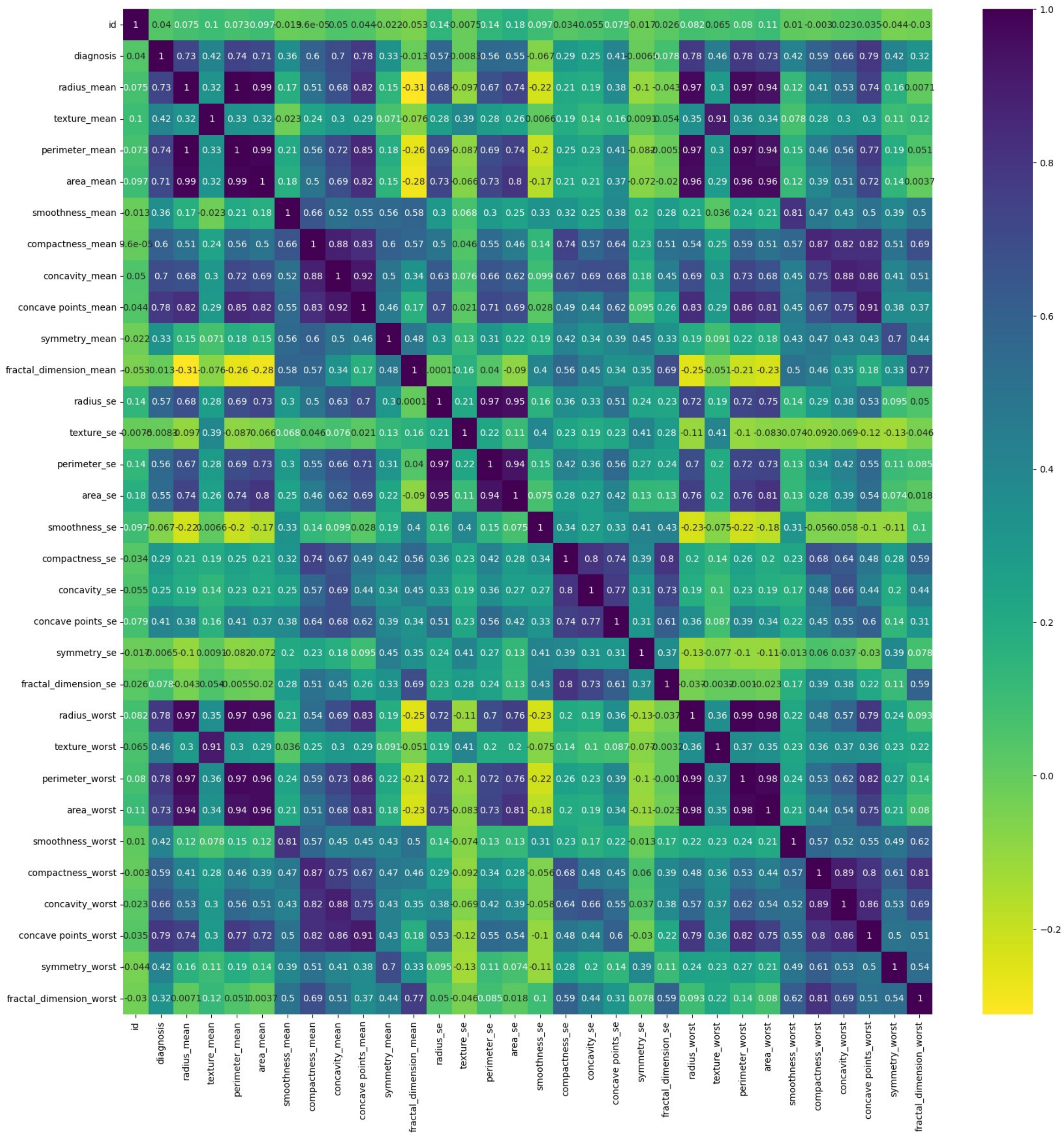




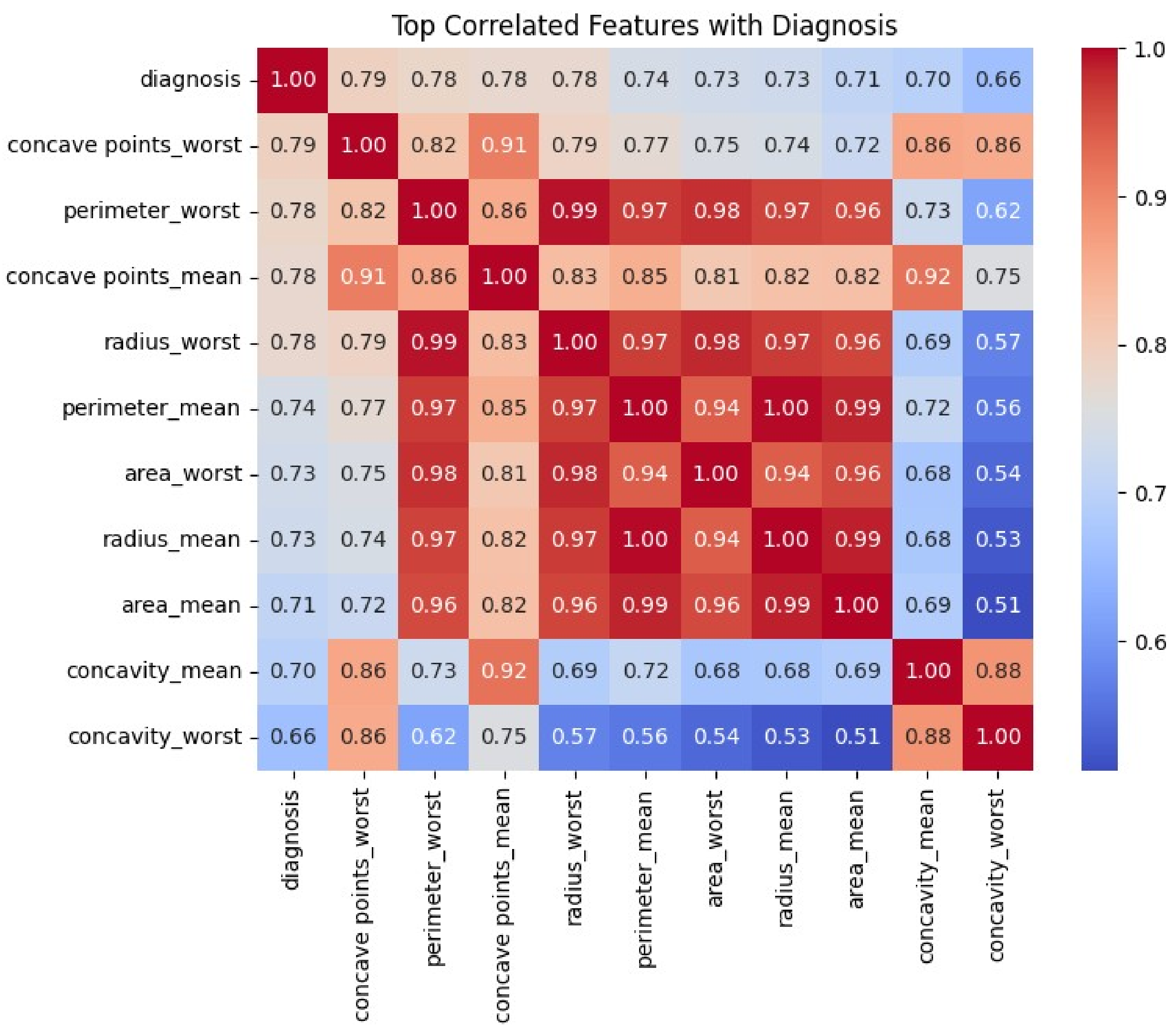




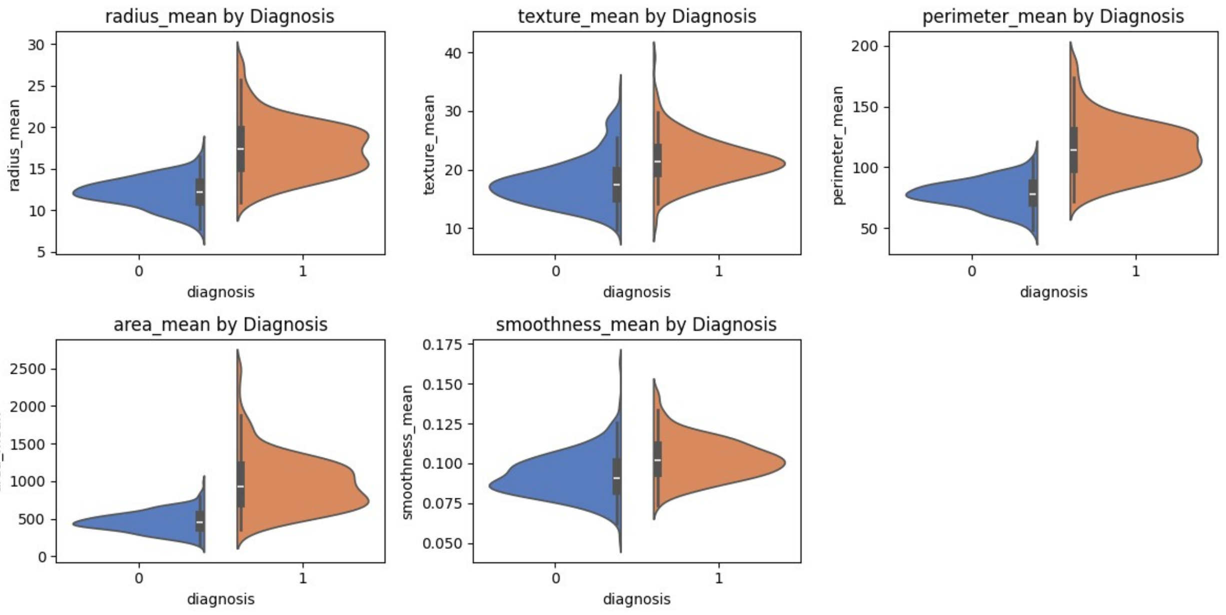
```
df['diagnosis'] = (df['diagnosis'] == 'M').astype(int) #encode the
label into 1/0
corr = df.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr, cmap='viridis_r',annot=True)
plt.show()
```



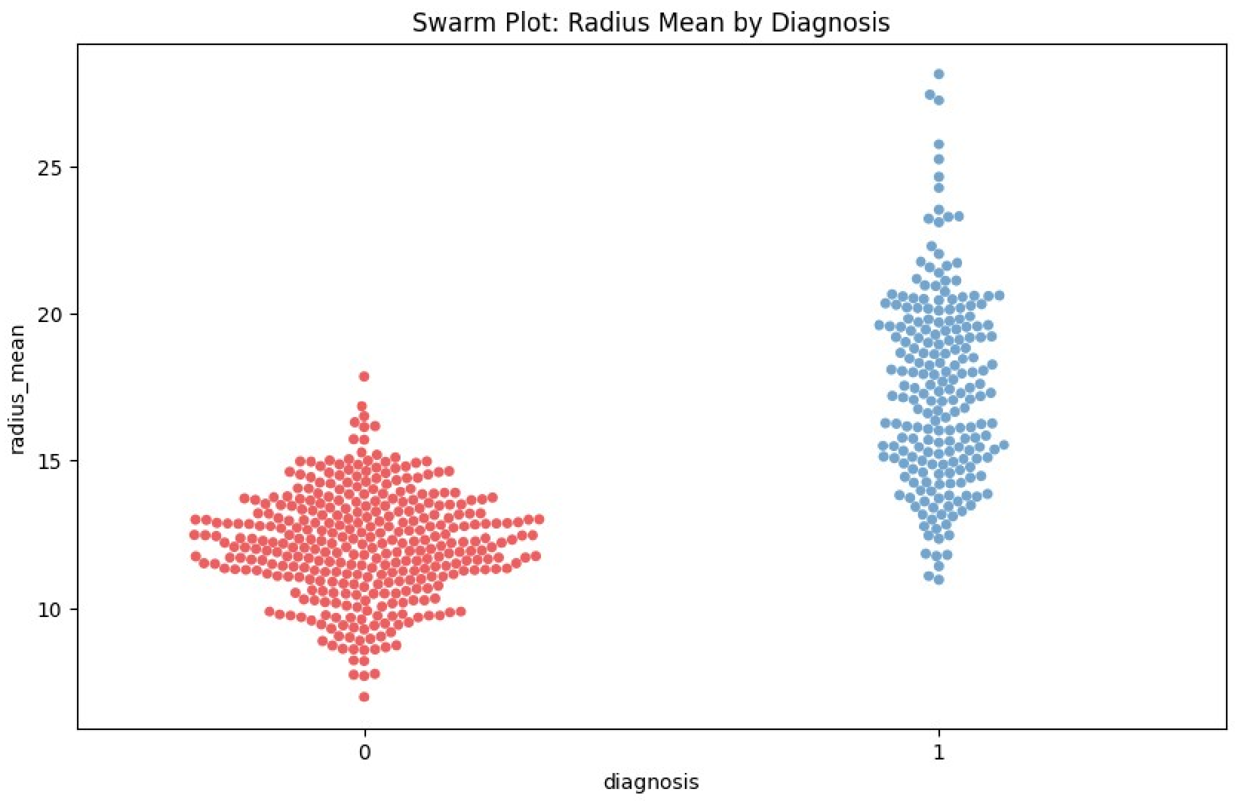
```
# Correlation with diagnosis
corr = df.corr()['diagnosis'].sort_values(ascending=False).head(11)  #
top 10 + diagnosis
plt.figure(figsize=(8,6))
sns.heatmap(df[corr.index].corr(), annot=True, cmap="coolwarm",
fmt=".2f")
plt.title("Top Correlated Features with Diagnosis")
plt.show()
```



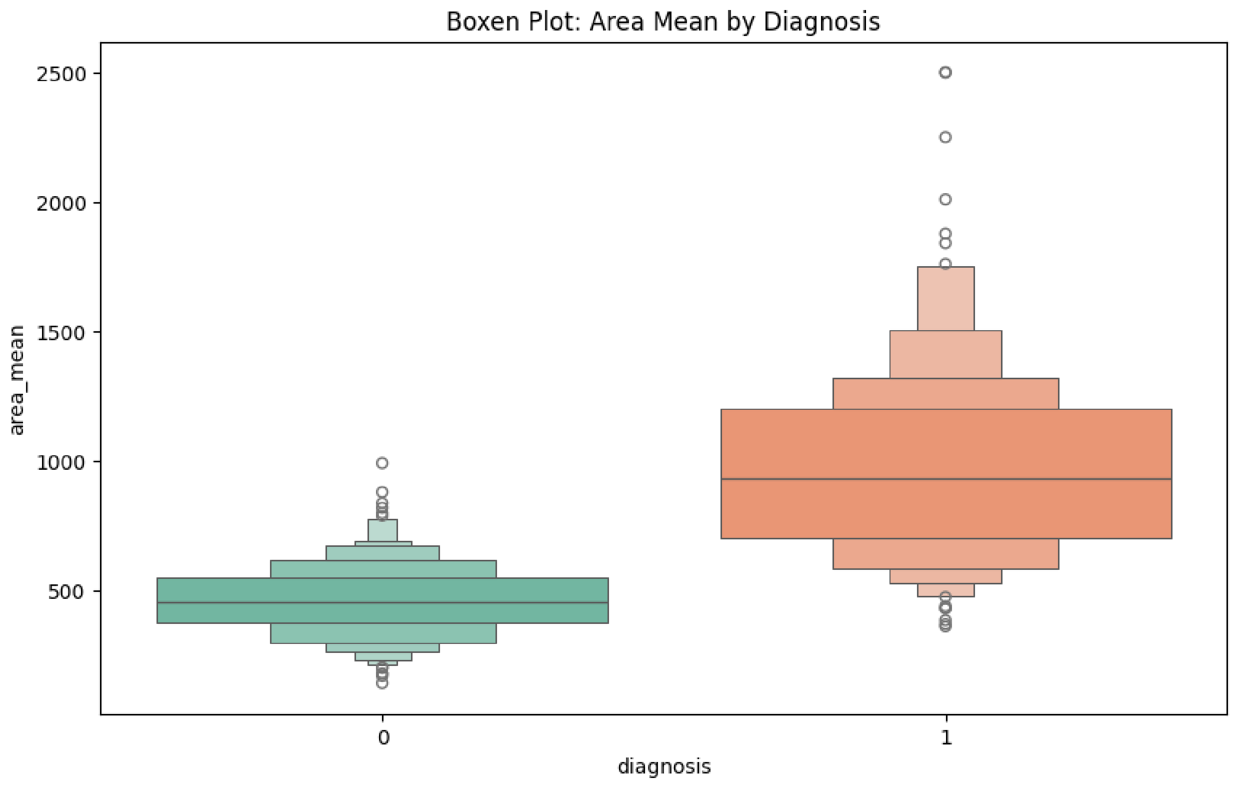
```
features = ['radius_mean', 'texture_mean', 'perimeter_mean',
'area_mean', 'smoothness_mean']
plt.figure(figsize=(12,6))
for i, feat in enumerate(features, 1):
    plt.subplot(2, 3, i)
    sns.violinplot(x="diagnosis", y=feat, data=df, palette="muted",
split=True)
    plt.title(f"{feat} by Diagnosis")
plt.tight_layout()
plt.show()
```



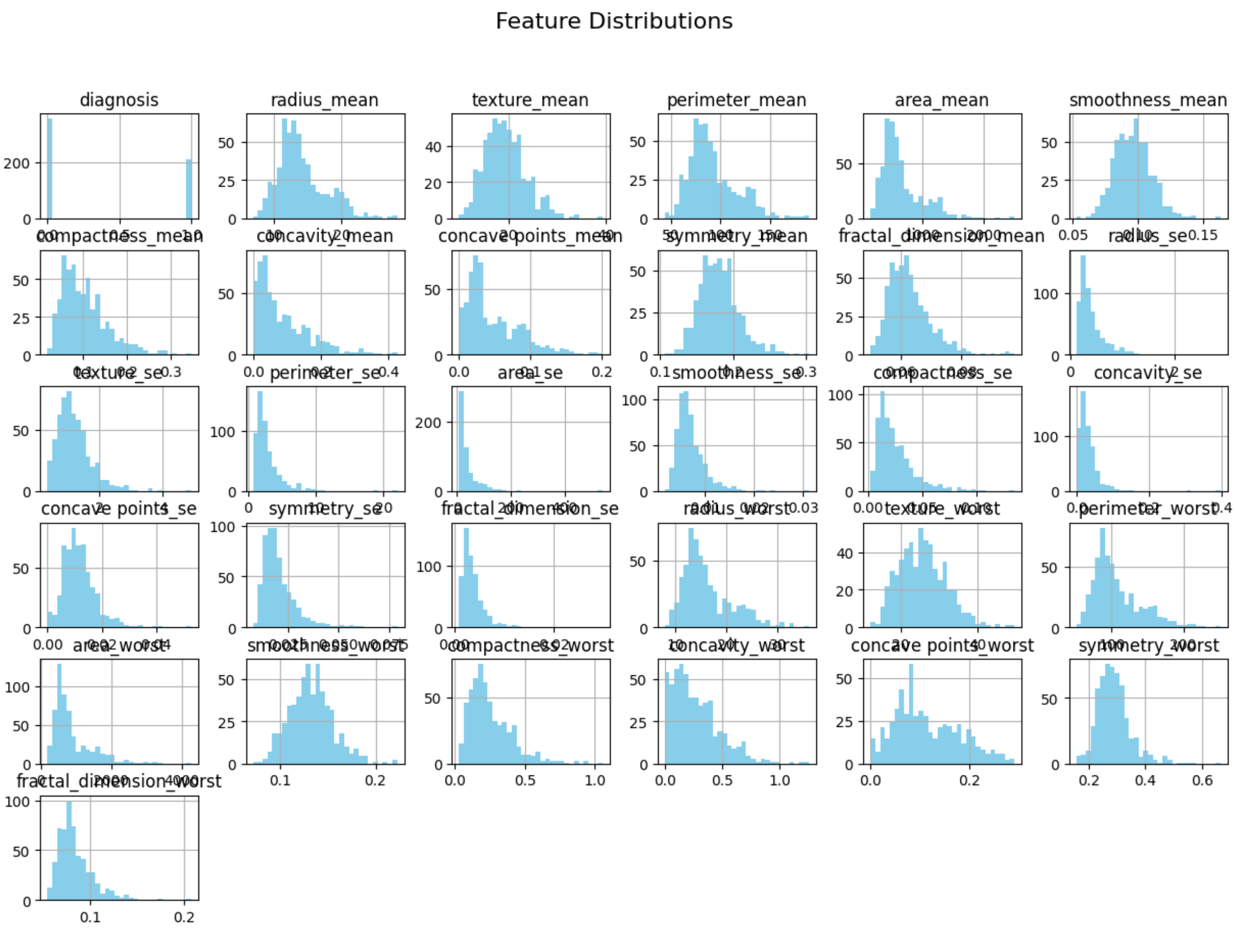
```
plt.figure(figsize=(10,6))
sns.swarmplot(x="diagnosis", y="radius_mean", data=df, palette="Set1",
alpha=0.7)
plt.title("Swarm Plot: Radius Mean by Diagnosis")
plt.show()
```



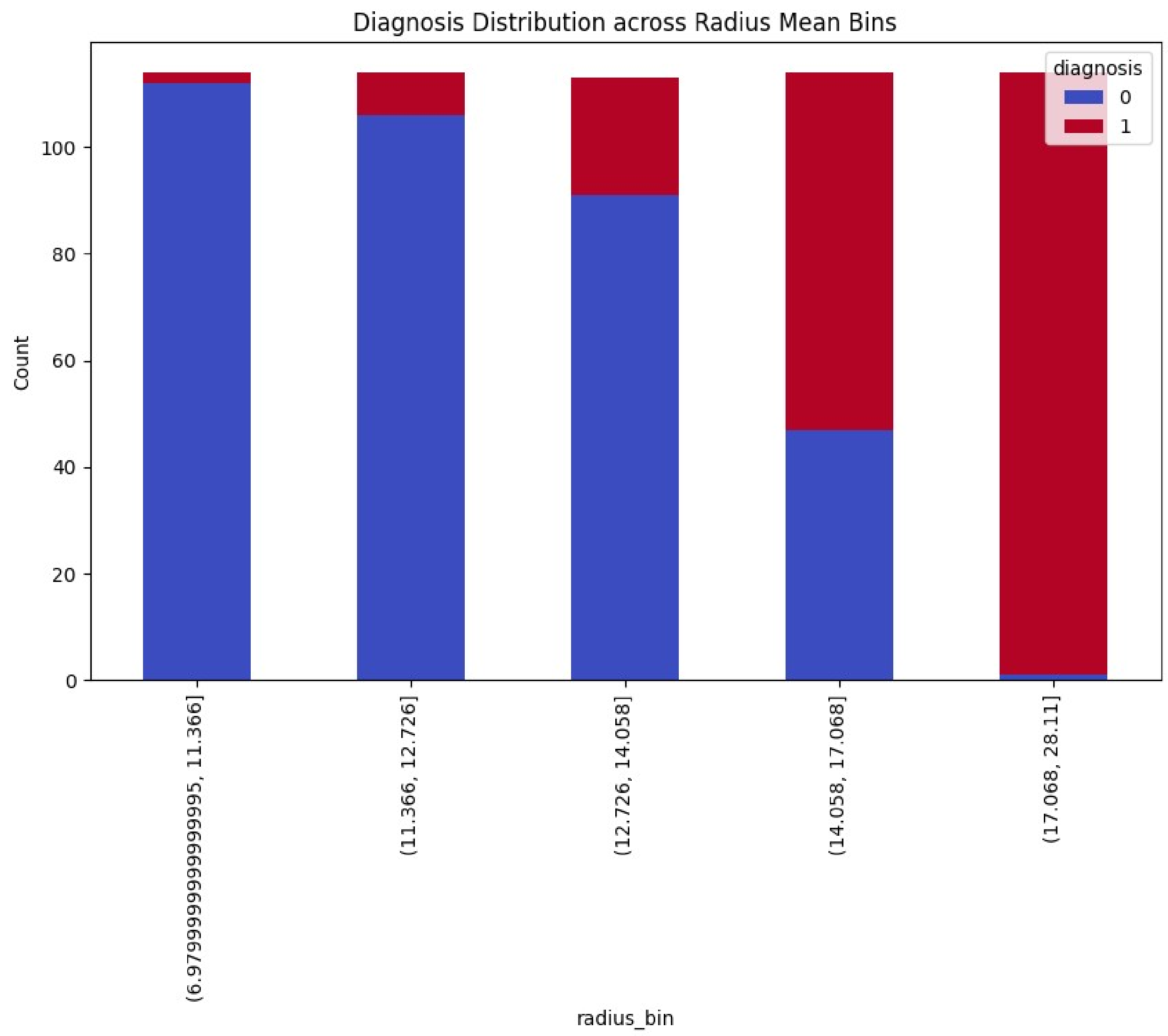
```
plt.figure(figsize=(10,6))
sns.boxenplot(x="diagnosis", y="area_mean", data=df, palette="Set2")
plt.title("Boxen Plot: Area Mean by Diagnosis")
plt.show()
```



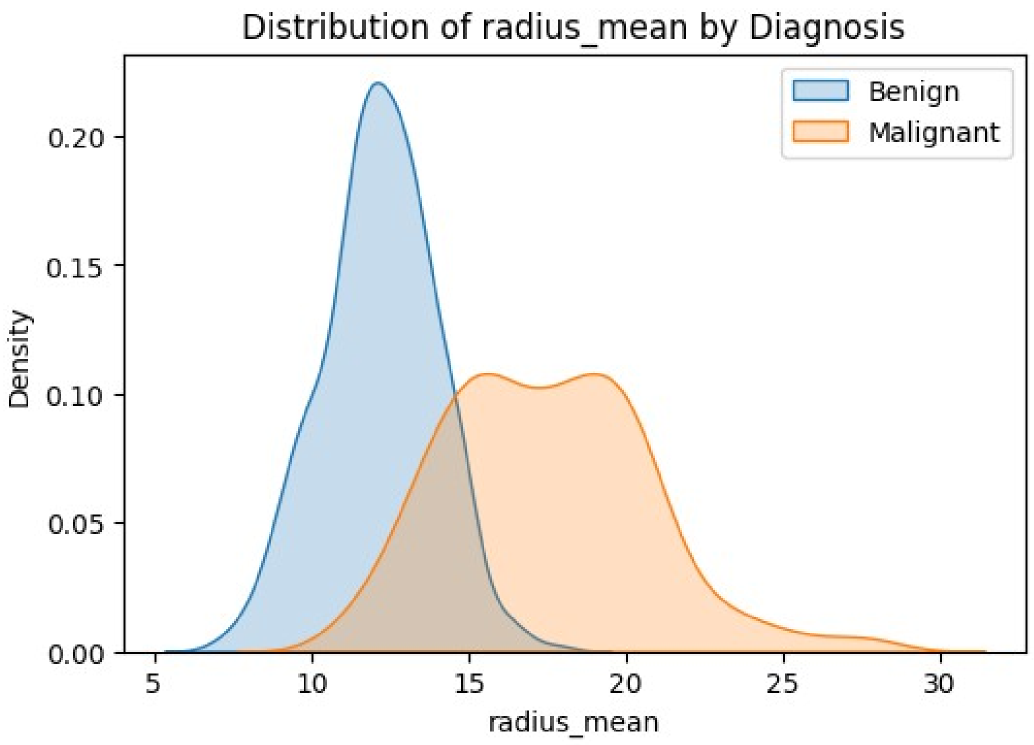
```
df.drop(['id'], axis=1, errors='ignore').hist(bins=30,
figsize=(15,10), color="skyblue")
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()
```

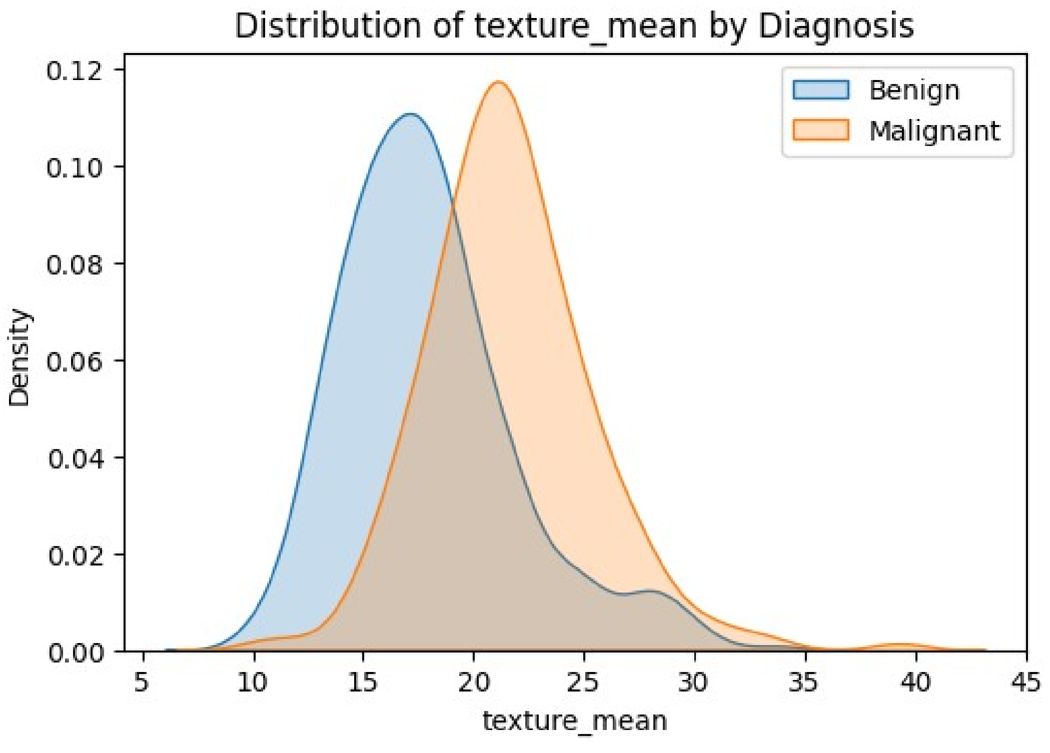


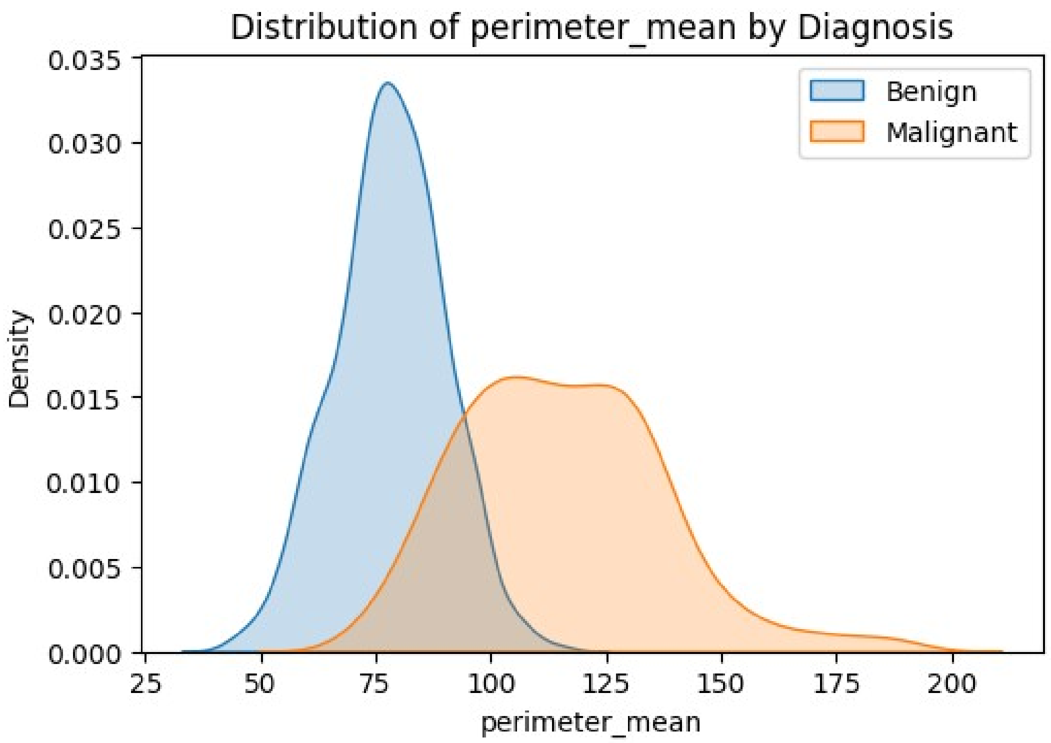
```
df['radius_bin'] = pd.qcut(df['radius_mean'], q=5)  # divide into 5
bins
ct = pd.crosstab(df['radius_bin'], df['diagnosis'])

ct.plot(kind="bar", stacked=True, figsize=(10,6), colormap="coolwarm")
plt.title("Diagnosis Distribution across Radius Mean Bins")
plt.ylabel("Count")
plt.show()
```

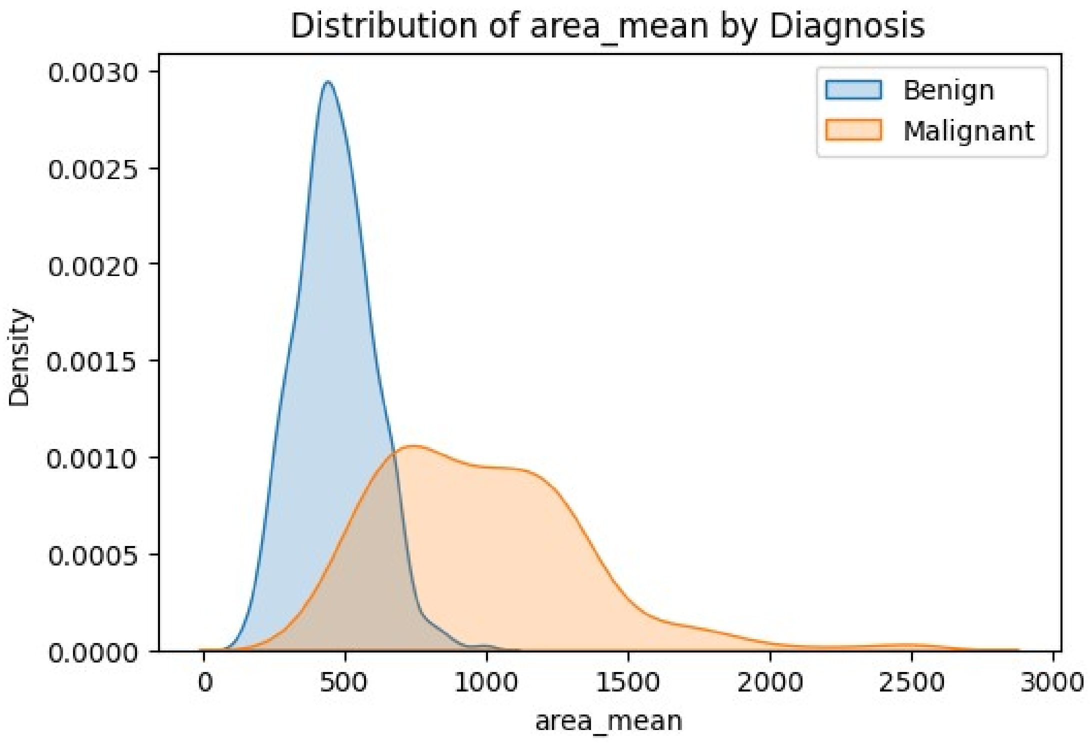


```
features = ['radius_mean', 'texture_mean', 'perimeter_mean',
'area_mean']
for feat in features:
    plt.figure(figsize=(6,4))
    sns.kdeplot(df[df['diagnosis']==0][feat], label="Benign",
shade=True)
    sns.kdeplot(df[df['diagnosis']==1][feat], label="Malignant",
shade=True)
    plt.title(f"Distribution of {feat} by Diagnosis")
    plt.legend()
    plt.show()
```


Distribution of radius_mean by Diagnosis
Benign
Malignant
Density
radius_mean
0.00
0.05
0.10
0.15
0.20
5
10
15
20
25
30

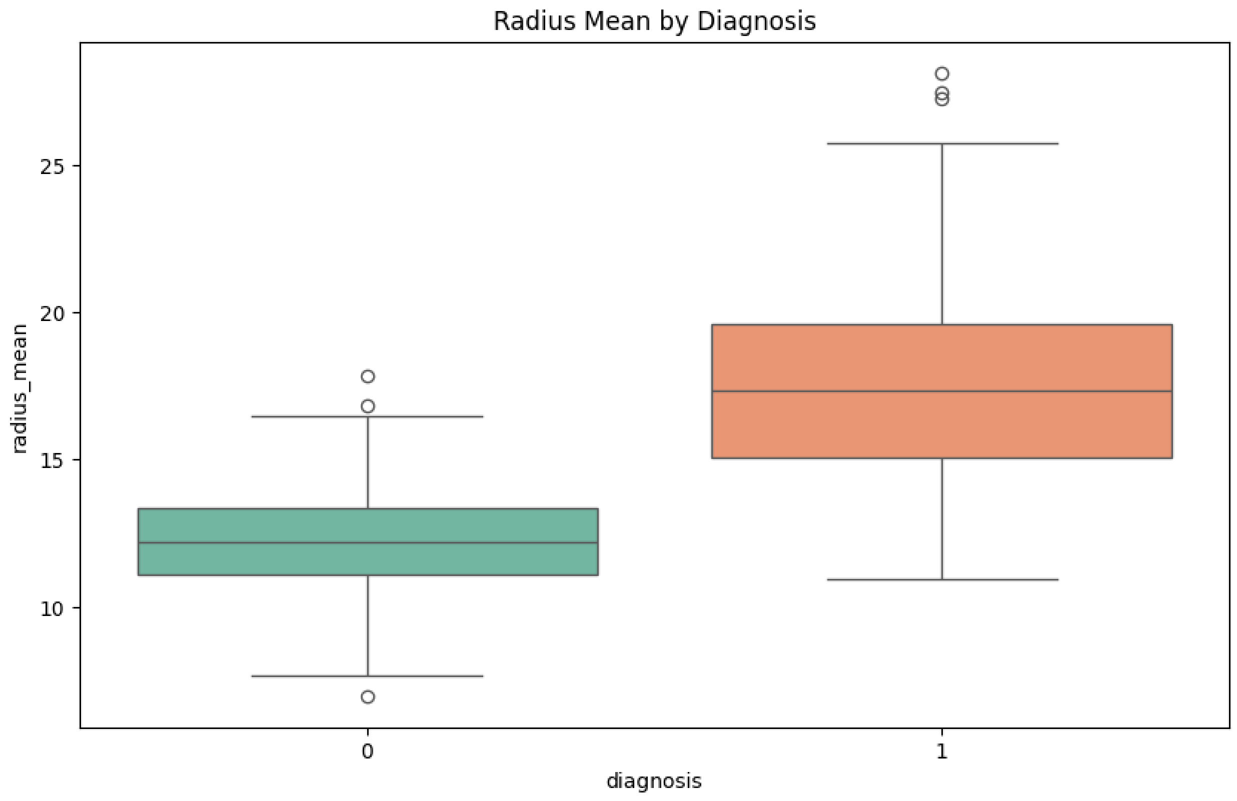

Distribution of texture_mean by Diagnosis
Benign
Malignant
Density
texture_mean
0.00
0.02
0.04
0.06
0.08
0.10
0.12
5
10
15
20
25
30
35
40
45



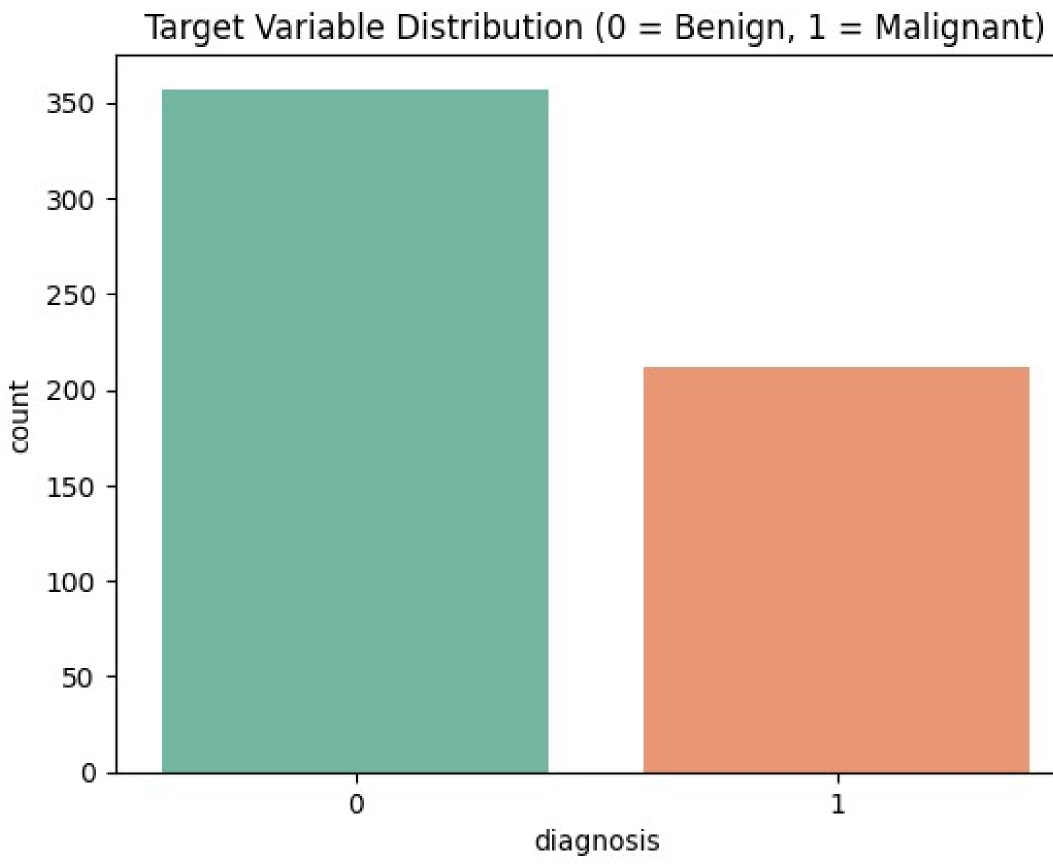


```
plt.figure(figsize=(10,6))
sns.boxplot(x="diagnosis", y="radius_mean", data=df, palette="Set2")
```

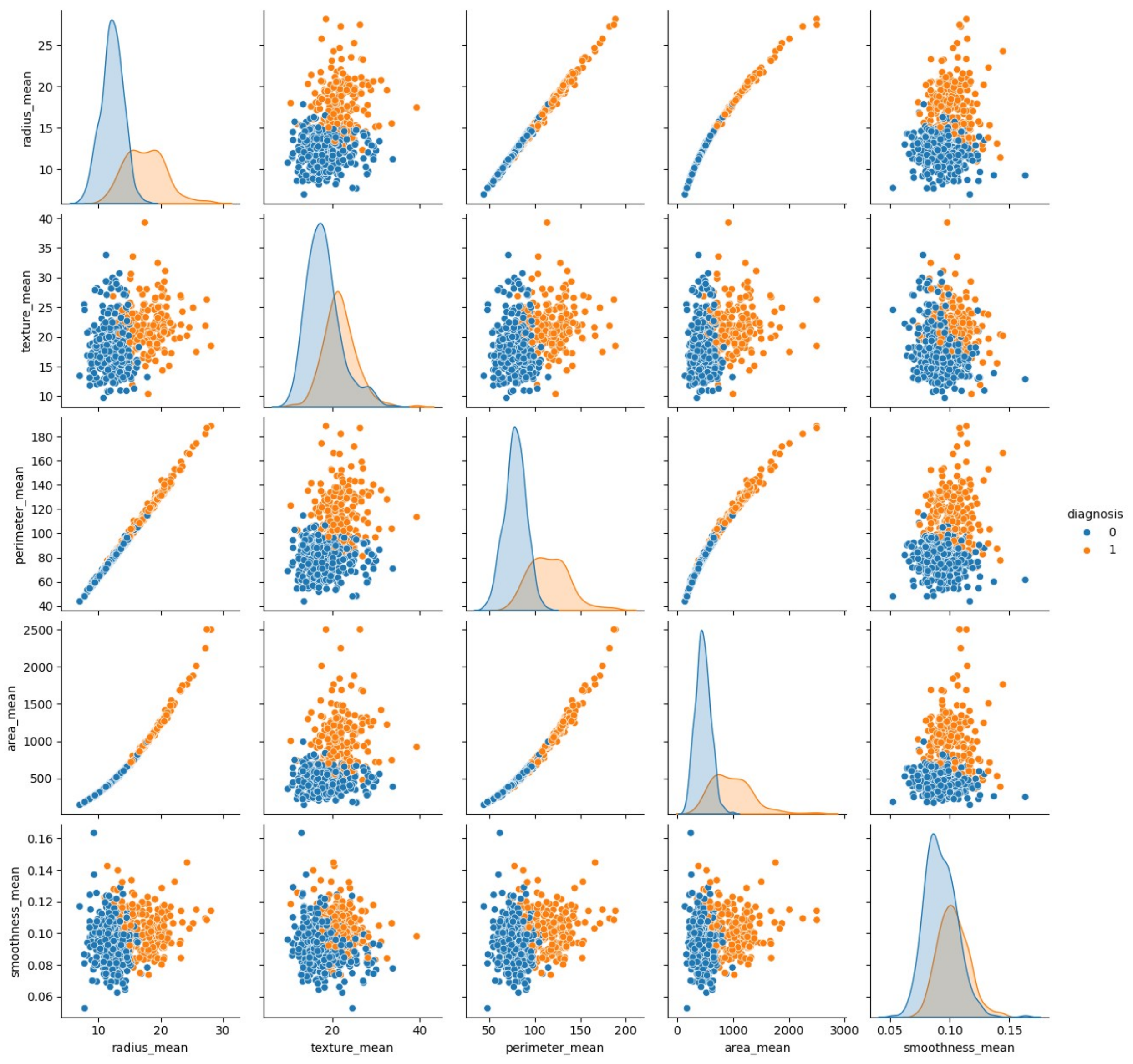
```
plt.title("Radius Mean by Diagnosis")
plt.show()
```



```
# Distribution of classes
sns.countplot(x='diagnosis', data=df, palette='Set2')
plt.title("Target Variable Distribution (0 = Benign, 1 = Malignant)")
plt.show()
```

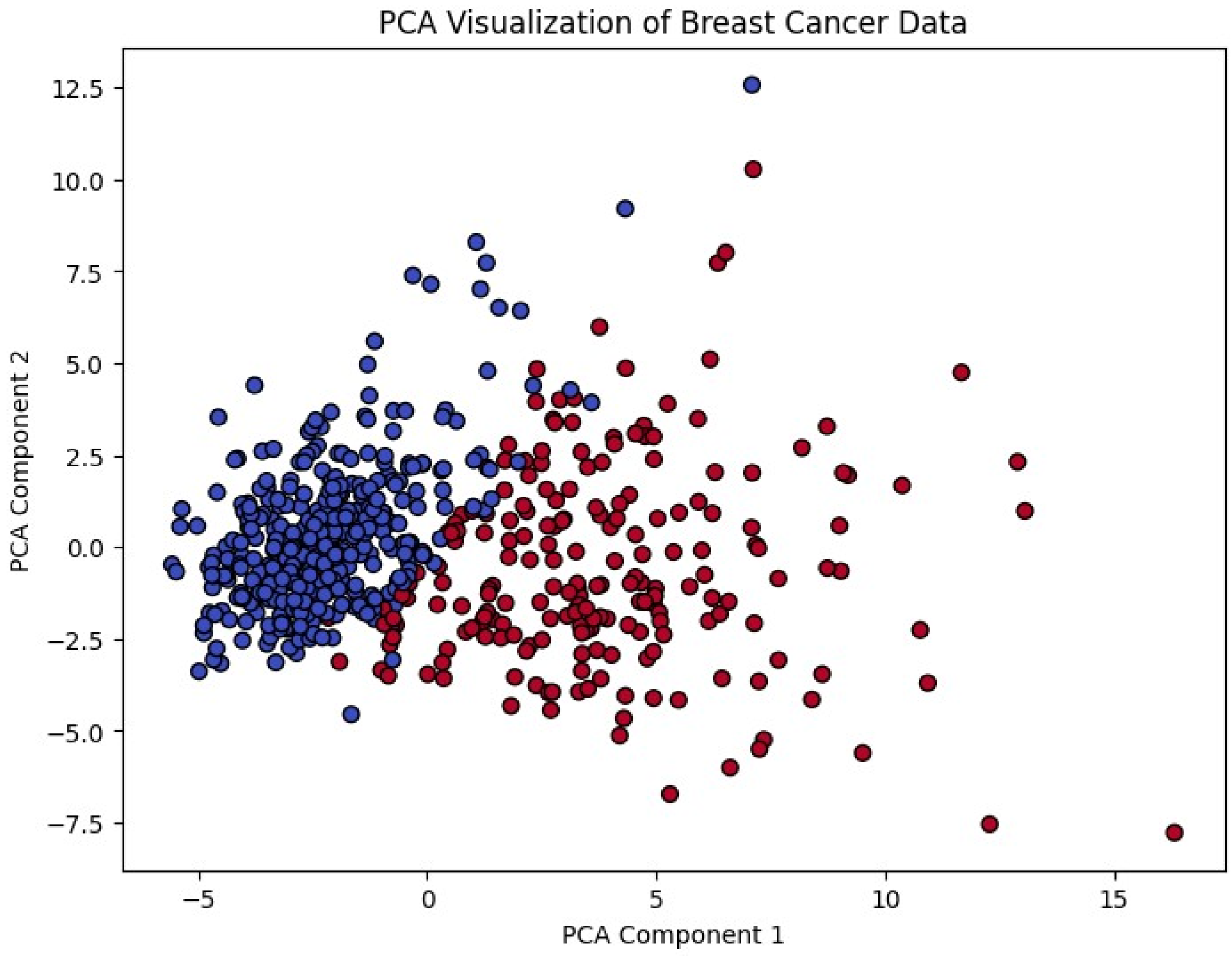


```
# Select only first 5 numerical features + target column
sns.pairplot(df[['radius_mean', 'texture_mean', 'perimeter_mean',
'area_mean', 'smoothness_mean', 'diagnosis']],
             hue="diagnosis", diag_kind="kde")
plt.show()
```



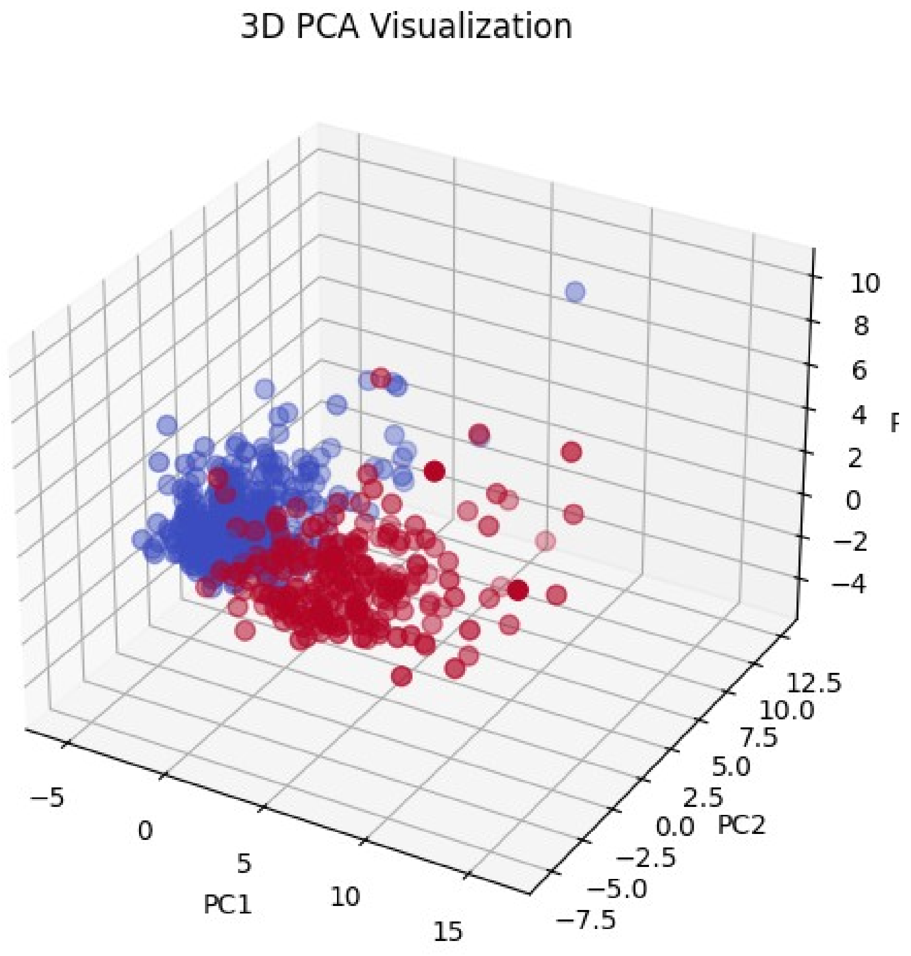
```
#  Preprocessing

X = df.drop(['id','diagnosis'], axis=1, errors='ignore')  # drop id if
present
y = df['diagnosis']

X = X.select_dtypes(include=[np.number])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
```

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=y, cmap="coolwarm",
edgecolor="k", s=40)
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("PCA Visualization of Breast Cancer Data")
plt.show()
```



```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)

fig = plt.figure(figsize=(8,6))
```

```
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_pca[:,0], X_pca[:,1], X_pca[:,2], c=y, cmap="coolwarm",
s=50)
ax.set_xlabel("PC1")
ax.set_ylabel("PC2")
ax.set_zlabel("PC3")
plt.title("3D PCA Visualization")
plt.show()
```



```
# Train SVM (Linear & RBF)

svm_linear = SVC(kernel="linear")
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)

print("\n--- Linear SVM Performance ---")
print(classification_report(y_test, y_pred_linear))
```

```
--- Linear SVM Performance ---
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        72
           1       1.00      0.90      0.95        42

    accuracy                           0.96       114
   macro avg       0.97      0.95      0.96       114
weighted avg       0.97      0.96      0.96       114
```

```
svm_rbf = SVC(kernel="rbf")
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)

print("\n--- RBF SVM Performance ---")
print(classification_report(y_test, y_pred_rbf))
```

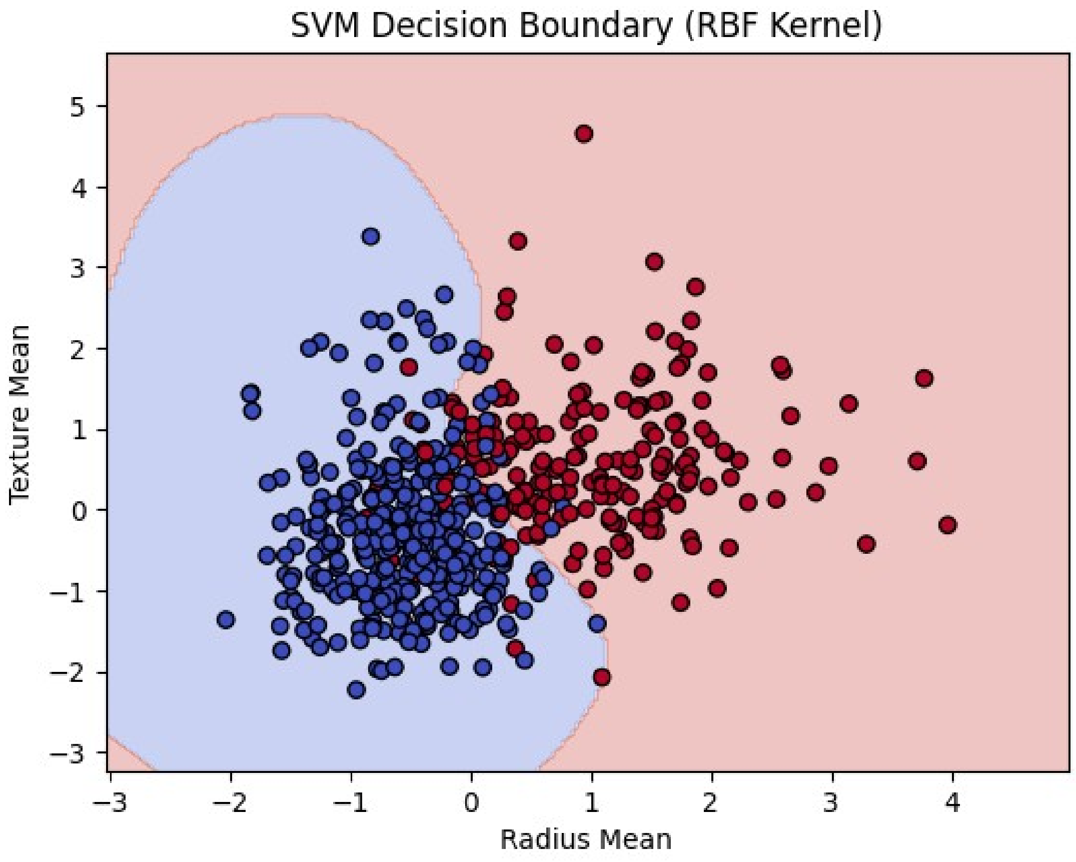
```
--- RBF SVM Performance ---
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

```
# Decision Boundary (2D Visualization)

# Use only two features (e.g., radius_mean, texture_mean)
X2 = df[['radius_mean', 'texture_mean']]
y2 = df['diagnosis']
X2_scaled = scaler.fit_transform(X2)

X2_train, X2_test, y2_train, y2_test = train_test_split(X2_scaled, y2,
test_size=0.2, random_state=42)

clf2 = SVC(kernel='rbf', C=1, gamma=0.5)
clf2.fit(X2_train, y2_train)
```

```
SVC(C=1, gamma=0.5)
```

```
# Meshgrid for decision boundary
x_min, x_max = X2_scaled[:,0].min() - 1, X2_scaled[:,0].max() + 1
y_min, y_max = X2_scaled[:,1].min() - 1, X2_scaled[:,1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
                     np.linspace(y_min, y_max, 200))

Z = clf2.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.coolwarm)
plt.scatter(X2_scaled[:,0], X2_scaled[:,1], c=y2, edgecolors='k',
cmap=plt.cm.coolwarm)
plt.xlabel("Radius Mean")
plt.ylabel("Texture Mean")
plt.title("SVM Decision Boundary (RBF Kernel)")
plt.show()
```



```
#  Hyperparameter Tuning

param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf']
}

grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=1, cv=5)
grid.fit(X_train, y_train)

print("\nBest Hyperparameters:", grid.best_params_)
print("Best Cross-Validation Score:", grid.best_score_)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits

Best Hyperparameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
Best Cross-Validation Score: 0.9758241758241759
```

```
# Evaluate best model
y_pred_best = grid.best_estimator_.predict(X_test)
print("\n--- Tuned RBF SVM Performance ---")
print(classification_report(y_test, y_pred_best))
```

```
--- Tuned RBF SVM Performance ---
              precision    recall  f1-score   support

           0       0.96      0.97      0.97        72
           1       0.95      0.93      0.94        42

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

```
#  Cross-validation with Linear SVM

cv_scores = cross_val_score(svm_linear, X_scaled, y, cv=5)
print("\nCross-validation Accuracy (Linear SVM):", cv_scores.mean())
```

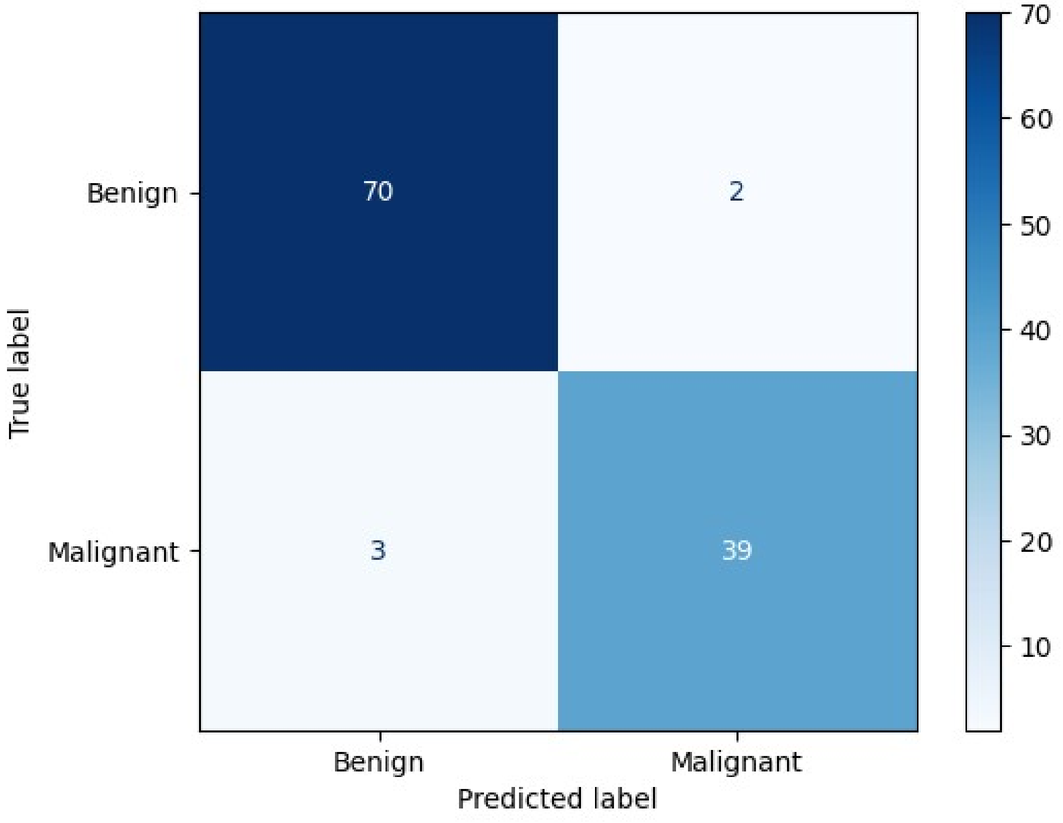
```
Cross-validation Accuracy (Linear SVM): 0.9701443875174661
```

```
from sklearn.feature_selection import SelectKBest, f_classif

selector = SelectKBest(score_func=f_classif, k=10)
fit = selector.fit(X_scaled, y)
feature_scores = pd.DataFrame({
    'Feature': X.columns,
    'Score': fit.scores_
}).sort_values(by="Score", ascending=False)

print(feature_scores.head(10))
```

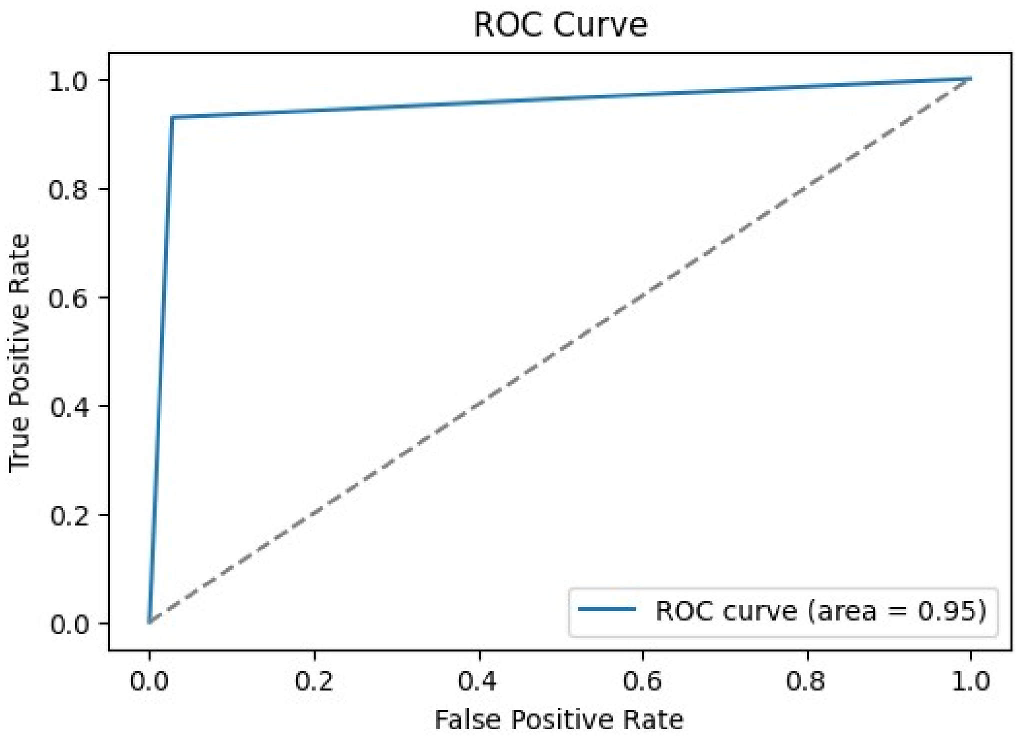
```
                 Feature       Score
27  concave points_worst  964.385393
22       perimeter_worst  897.944219
7    concave points_mean  861.676020
20          radius_worst  860.781707
2         perimeter_mean  697.235272
23            area_worst  661.600206
0            radius_mean  646.981021
3              area_mean  573.060747
```

```
6         concavity_mean  533.793126
26       concavity_worst  436.691939
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
roc_curve, auc

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_best)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=["Benign", "Malignant"])
disp.plot(cmap="Blues")
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_best)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"ROC curve (area = {roc_auc:.2f})")
plt.plot([0,1],[0,1],'--', color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

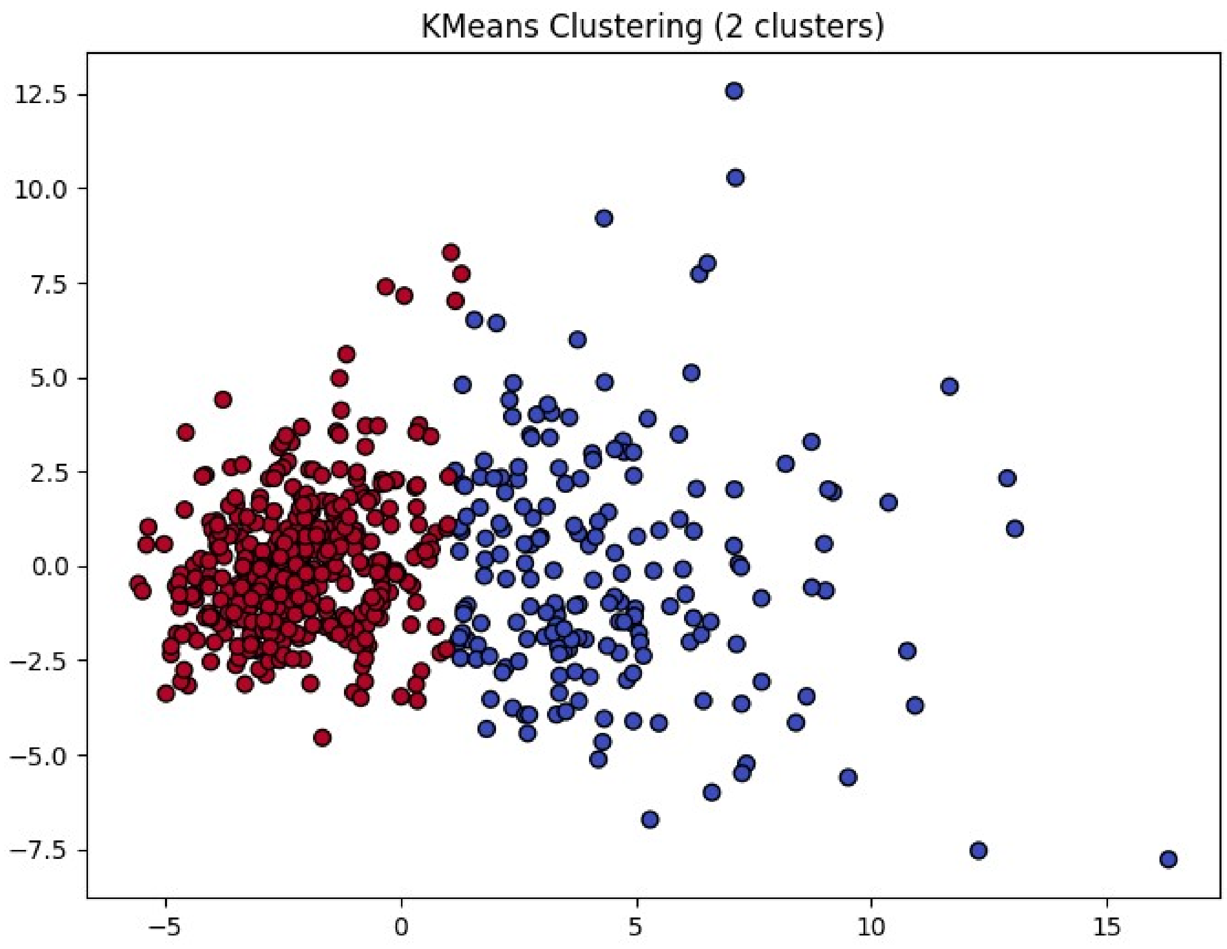


True label
Benign
Malignant
70
2
3
39
Predicted label
Benign
Malignant
70
60
50
40
30
20
10



ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.95)

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=clusters, cmap="coolwarm", s=40,
edgecolor="k")
plt.title("KMeans Clustering (2 clusters)")
plt.show()
```



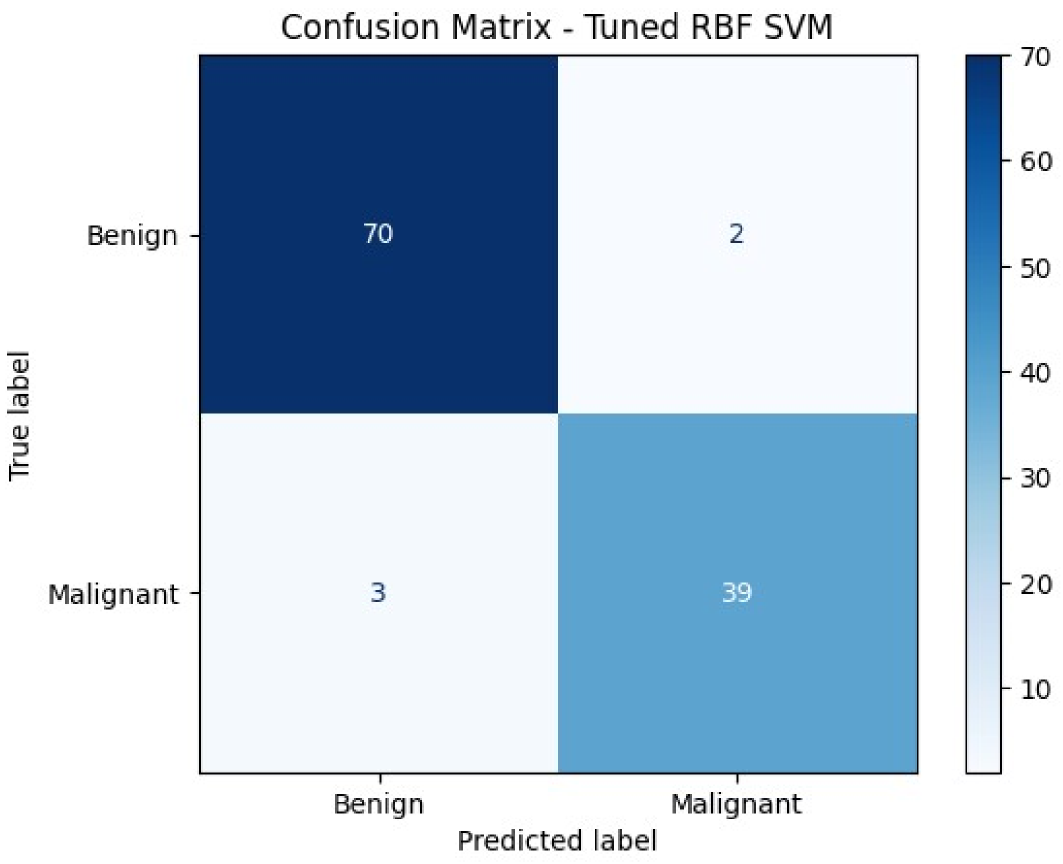
```
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# ==============================
# Hyperparameter Tuning (RBF SVM)
# ==============================
param_grid = {
    'C': [0.1, 1, 10, 100],        # Regularization parameter
    'gamma': [1, 0.1, 0.01, 0.001], # Kernel coefficient
    'kernel': ['rbf']
}

grid = GridSearchCV(SVC(probability=True), param_grid, refit=True,
verbose=1, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

print("□ Best Parameters:", grid.best_params_)
print("□ Best CV Score:", grid.best_score_)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
□ Best Parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
□ Best CV Score: 0.9758241758241759
```

```
# Evaluate on test set
y_pred_best = grid.best_estimator_.predict(X_test)

print("\n--- Tuned RBF SVM Performance ---")
print("Accuracy:", accuracy_score(y_test, y_pred_best))
print(classification_report(y_test, y_pred_best))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_best))
```

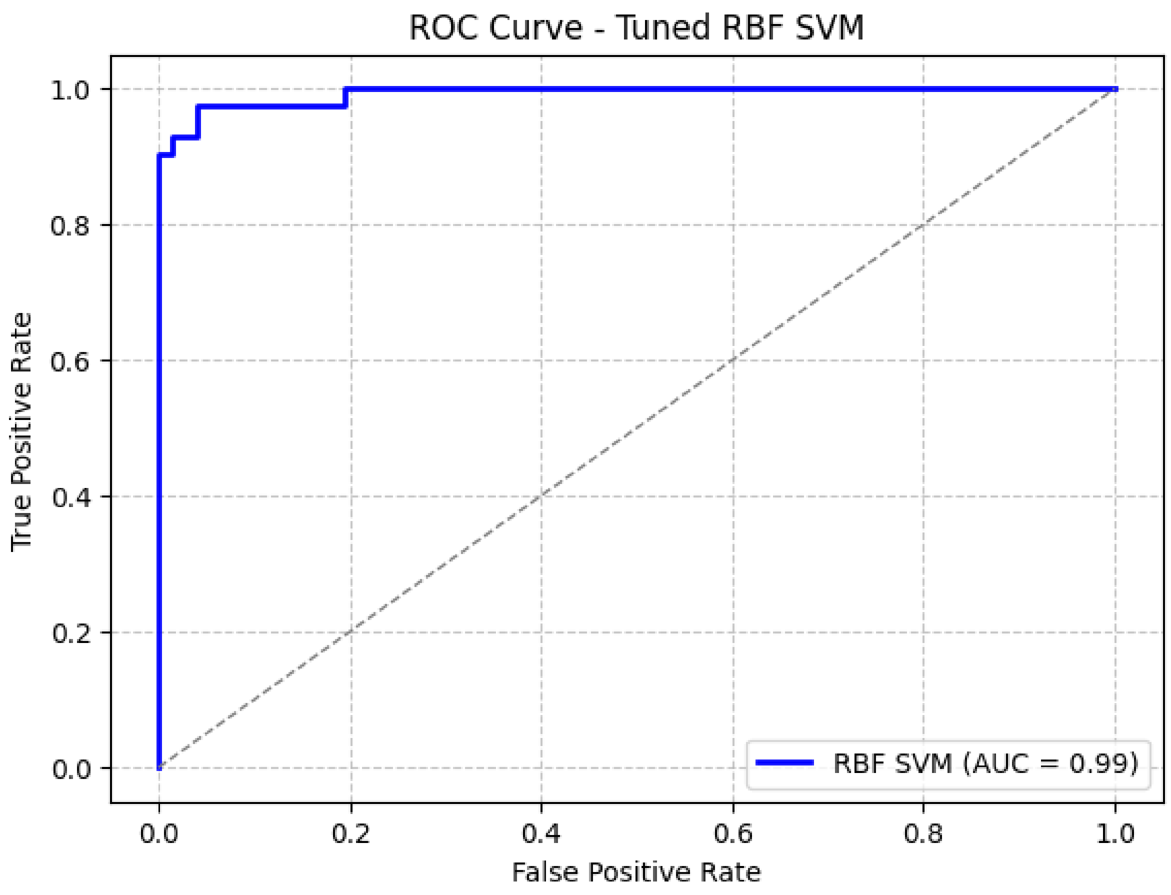
```

--- Tuned RBF SVM Performance ---
Accuracy: 0.956140350877193
              precision    recall  f1-score   support

           0       0.96      0.97      0.97        72
           1       0.95      0.93      0.94        42

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114

Confusion Matrix:
 [[70  2]
 [ 3 39]]
```

```
# Plot Confusion Matrix
disp =
ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test,
y_pred_best),
                              display_labels=["Benign","Malignant"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix - Tuned RBF SVM")
plt.show()
```



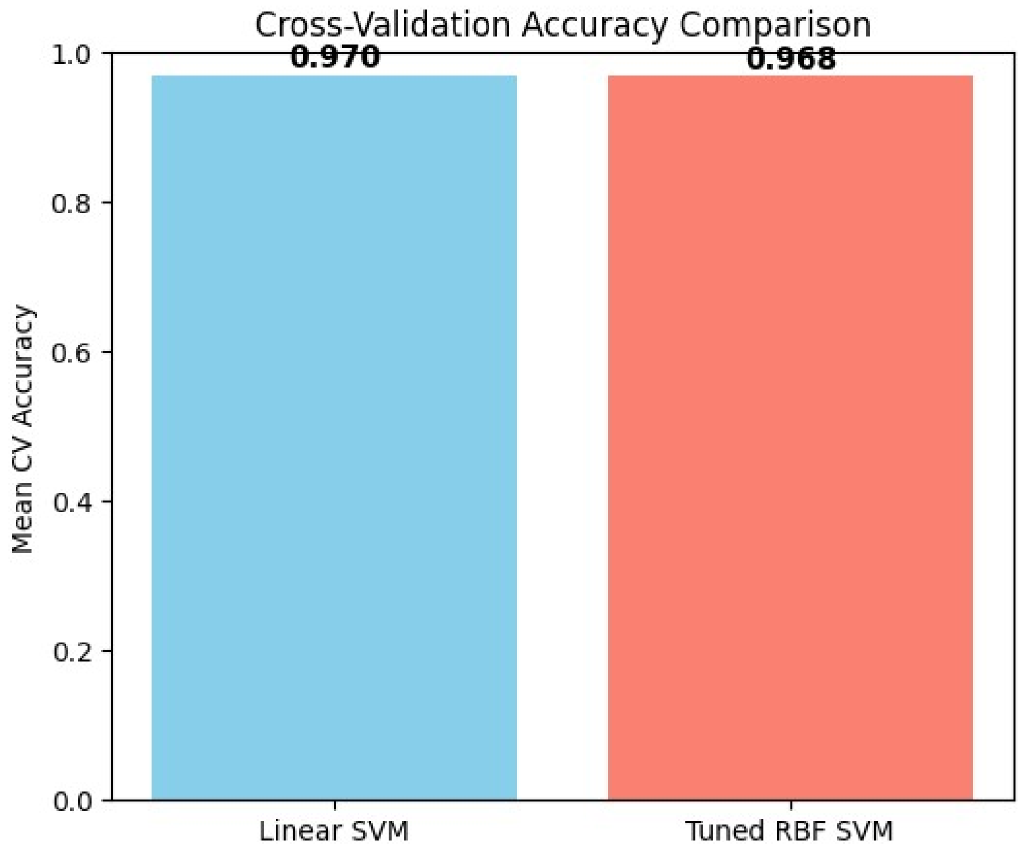
```
# Cross-validation (Linear SVM)

# Linear SVM
svm_linear = SVC(kernel="linear", C=1)
cv_scores_linear = cross_val_score(svm_linear, X_scaled, y, cv=5,
scoring='accuracy')
print("\nCross-validation Accuracy (Linear SVM):",
cv_scores_linear.mean())


Cross-validation Accuracy (Linear SVM): 0.9701443875174661

# Best RBF SVM (from GridSearch)
best_rbf = grid.best_estimator_
cv_scores_rbf = cross_val_score(best_rbf, X_scaled, y, cv=5,
scoring='accuracy')
print("Cross-validation Accuracy (Best RBF SVM):",
cv_scores_rbf.mean())

Cross-validation Accuracy (Best RBF SVM): 0.9683744760130415

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
```

```
# Predict probabilities with best RBF SVM
y_prob = grid.best_estimator_.predict_proba(X_test)[:,1]

# Compute ROC curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

# Plot ROC Curve
plt.figure(figsize=(7,5))
plt.plot(fpr, tpr, color="blue", lw=2, label=f"RBF SVM (AUC =
{roc_auc:.2f})")
plt.plot([0,1], [0,1], color="gray", linestyle="--", lw=1)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Tuned RBF SVM")
plt.legend(loc="lower right")
plt.grid(True, linestyle="--", alpha=0.7)
plt.show()
```



```
# Cross-validation accuracy results
linear_acc = cv_scores_linear.mean()
```

```
rbf_acc = cv_scores_rbf.mean()

# Bar chart
models = ['Linear SVM', 'Tuned RBF SVM']
scores = [linear_acc, rbf_acc]

plt.figure(figsize=(6,5))
plt.bar(models, scores, color=['skyblue','salmon'])
plt.ylabel("Mean CV Accuracy")
plt.ylim(0,1)
for i, score in enumerate(scores):
    plt.text(i, score+0.01, f"{score:.3f}", ha='center', fontsize=11,
fontweight='bold')
plt.title("Cross-Validation Accuracy Comparison")
plt.show()
```


Cross-Validation Accuracy Comparison
Mean CV Accuracy
Linear SVM
Tuned RBF SVM