TITLE : To perform Exploratory Data Analysis (EDA), enhance classification accuracy using ensemble

methods, and evaluate model performance using appropriate validation techniques in Python.

NAME : Shinde Shubham Dnyandev

ROLL NO. : 23107121

CLASS : TY-B

BATCH : B

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, Voti
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
cancer = load_breast_cancer()
df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
df['target'] = cancer.target

print("\nDataset shape:", df.shape)
print("\nFirst 5 rows:\n", df.head())
print("\nClass distribution:\n", df['target'].value_counts())
```

```
Dataset shape: (569, 31)

First 5 rows:
    mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                 0.07871  ...          17.33           184.60      2019.0
1                 0.05667  ...          23.41           158.80      1956.0
2                 0.05999  ...          25.53           152.50      1709.0
3                 0.09744  ...          26.50            98.87       567.7
4                 0.05883  ...          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
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

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]

Class distribution:
 target
1    357
0    212
Name: count, dtype: int64
```

In [6]:
```
X = df.drop('target', axis=1)
y = df['target']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, rando
```

In [10]:
```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
gb = GradientBoostingClassifier(n_estimators=100, random_state=42)
ensemble = VotingClassifier(estimators=[('rf', rf), ('gb', gb)], voting='soft')

models = {'Random Forest': rf, 'Gradient Boosting': gb, 'Voting Ensemble': ensemble}
```

In [12]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print(f"\n=== {name} ===")
```

```
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("ROC-AUC:", roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
=== Random Forest ===
Accuracy: 0.956140350877193
ROC-AUC: 0.9937169312169312
Confusion Matrix:
 [[39  3]
 [ 2 70]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        42
           1       0.96      0.97      0.97        72

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


=== Gradient Boosting ===
Accuracy: 0.956140350877193
ROC-AUC: 0.9907407407407407
Confusion Matrix:
 [[38  4]
 [ 1 71]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.90      0.94        42
           1       0.95      0.99      0.97        72

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


=== Voting Ensemble ===
Accuracy: 0.956140350877193
ROC-AUC: 0.9927248677248678
Confusion Matrix:
 [[38  4]
 [ 1 71]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.90      0.94        42
           1       0.95      0.99      0.97        72

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

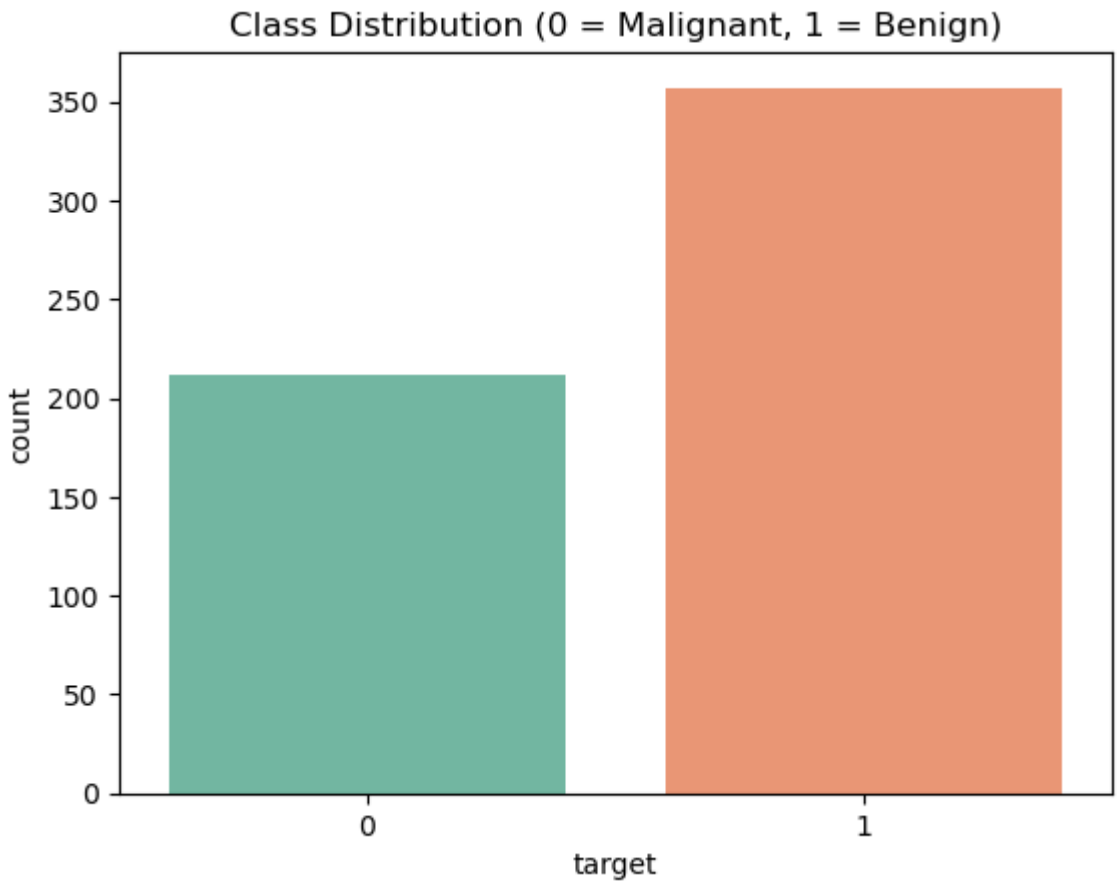
In [13]:
```
    # Cross-validation accuracy
    cv_score = cross_val_score(model, X_scaled, y, cv=5, scoring='accuracy').mean()
    print("Cross-validation Accuracy:", cv_score)
```

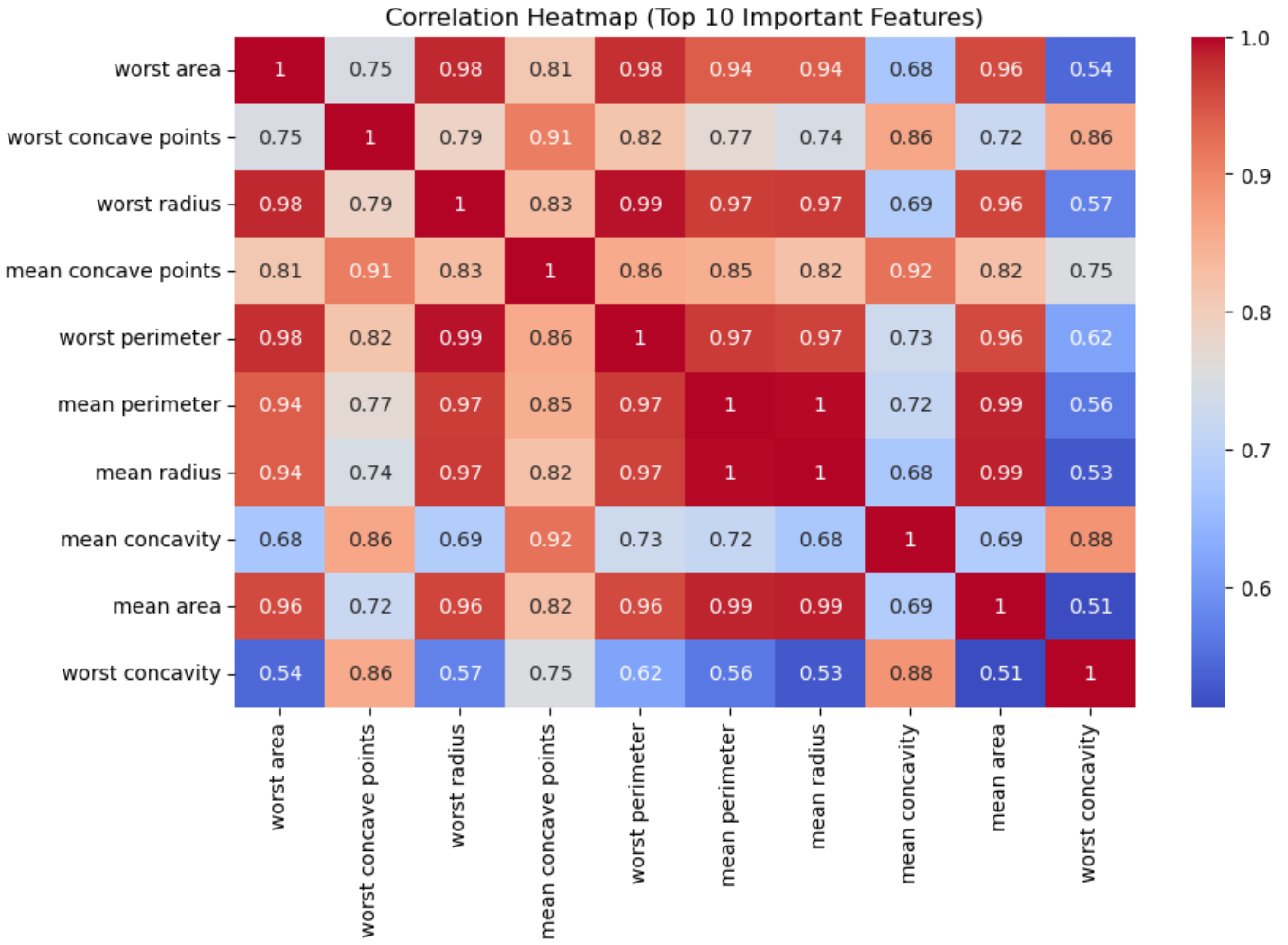
```
Cross-validation Accuracy: 0.9613569321533924
```

In [15]:
```
feat_importances = pd.Series(rf.feature_importances_, index=cancer.feature_names)
top_features = feat_importances.nlargest(10).index

sns.countplot(x='target', data=df, palette='Set2')
```

```
plt.title("Class Distribution (0 = Malignant, 1 = Benign)")
plt.show()

plt.figure(figsize=(10, 6))
sns.heatmap(df[top_features].corr(), annot=True, cmap="coolwarm", cbar=True)
plt.title("Correlation Heatmap (Top 10 Important Features)")
plt.show()
```

```
/tmp/ipykernel_3111/1012119997.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='target', data=df, palette='Set2')
```




Correlation Heatmap (Top 10 Important Features)
worst area
worst concave points
worst radius
mean concave points
worst perimeter
mean perimeter
mean radius
mean concavity
mean area
worst concavity
1 0.75 0.98 0.81 0.98 0.94 0.94 0.68 0.96 0.54
0.75 1 0.79 0.91 0.82 0.77 0.74 0.86 0.72 0.86
0.98 0.79 1 0.83 0.99 0.97 0.97 0.69 0.96 0.57
0.81 0.91 0.83 1 0.86 0.85 0.82 0.92 0.82 0.75
0.98 0.82 0.99 0.86 1 0.97 0.97 0.73 0.96 0.62
0.94 0.77 0.97 0.85 0.97 1 1 0.72 0.99 0.56
0.94 0.74 0.97 0.82 0.97 1 1 0.68 0.99 0.53
0.68 0.86 0.69 0.92 0.73 0.72 0.68 1 0.69 0.88
0.96 0.72 0.96 0.82 0.96 0.99 0.99 0.69 1 0.51
0.54 0.86 0.57 0.75 0.62 0.56 0.53 0.88 0.51 1
1.0
0.9
0.8
0.7
0.6

In [ ]: