Importing Libraries And Datasets

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
```

```
df = pd.read_csv("/content/breast-cancer.csv")
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1

5 rows × 32 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

```
df.describe()
```

```
df.shape
```

```
(569, 32)
```

Preparing For Binary Classification

```
df = df.drop('id', axis=1)
```

```
le = LabelEncoder()
df['diagnosis'] = le.fit_transform(df['diagnosis'])
```

```
X = df.drop('diagnosis', axis=1)
y = df['diagnosis']
```

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

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
print(f"Features (X) shape: {X.shape}")
print(f"Target (y) shape: {y.shape}")
```

```
Features (X) shape: (569, 30)
Target (y) shape: (569,)
```

Training An SVM

```
svm_linear = SVC(kernel='linear', C=1, random_state=42)
svm_linear.fit(X_train_scaled, y_train)

y_pred_linear = svm_linear.predict(X_test_scaled)
print(f"Linear SVM Accuracy: {accuracy_score(y_test, y_pred_linear):.5f}")
print("Classification Report (Linear):\n", classification_report(y_test, y_pred_linear))

svm_rbf = SVC(kernel='rbf', random_state=42)
svm_rbf.fit(X_train_scaled, y_train)

y_pred_rbf = svm_rbf.predict(X_test_scaled)
print(f"\nRBF SVM Accuracy: {accuracy_score(y_test, y_pred_rbf):.5f}")
print("Classification Report (RBF):\n", classification_report(y_test, y_pred_rbf))
```

```
Linear SVM Accuracy: 0.9649
Classification Report (Linear):
              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
```

```
RBF SVM Accuracy: 0.9737
Classification Report (RBF Default):
              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
```

Tuning Hyperparameters And Cross Validation

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

grid_search = GridSearchCV(
    estimator=SVC(random_state=42),
    param_grid=param_grid,
    scoring='accuracy',
    cv=5,
    verbose=1,
    n_jobs=-1
)

grid_search.fit(X_train_scaled, y_train)

print("\nGrid Search Results :-\n")
print(f"Best Hyperparameters: {grid_search.best_params_}")
print(f"Best Cross-Validation Score: {grid_search.best_score_:.4f}")

best_svm = grid_search.best_estimator_
y_pred_best = best_svm.predict(X_test_scaled)

print("\nFinal Best Model Performance (on Test Set) :-\n")
print(f"Final Tuned RBF SVM Test Accuracy: {accuracy_score(y_test, y_pred_best):.4f}")
print("Classification Report (Tuned RBF):\n", classification_report(y_test, y_pred_best))
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits

Grid Search Results :-

Best Hyperparameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Best Cross-Validation Score: 0.9758

Final Best Model Performance (on Test Set) :-

Final Tuned RBF SVM Test Accuracy: 0.9737
Classification Report (Tuned RBF):
              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
```