Aim

To develop a machine learning model for accurately classifying breast cancer tumors as malignant or benign using a publicly available dataset.

Objective

1. To preprocess the breast cancer dataset by handling missing values and encoding categorical data.
2. To evaluate the performance of a Support Vector Machine (SVM) model using cross-validation.
3. To implement a pipeline that scales features and trains the SVM model to ensure consistent preprocessing.

Summary

This project aims to classify breast cancer tumors into malignant or benign categories using machine learning techniques. The dataset is preprocessed by handling missing values and encoding categorical features. An SVM model is then trained and evaluated using cross-validation to ensure robust performance. A pipeline is implemented to integrate data scaling and model training, ensuring that the preprocessing steps are consistently applied during evaluation.

Tools and Libraries Used

- Google Colab
- Python
- Pandas
- Seaborn
- Scikit-learn

Procedure

1. **Mount Google Drive and Import Libraries**

CODE:

```
from google.colab import drive

drive.mount('/content/drive')

import sys

import sklearn

import pandas as pd

import seaborn as sns

from sklearn.impute import SimpleImputer
```

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import KFold, cross_val_score
from sklearn import svm
from sklearn.pipeline import Pipeline
```

2. **Load and Explore Dataset**

CODE:

```
df = pd.read_csv("/content/drive/MyDrive/brest cancer tumer size/data.csv")
print(df.describe().T)
print(df.isnull().sum())
```

3. **Data Preprocessing**
    - Drop columns with all NaN values
    - Encode 'diagnosis' column to numeric values
    - Impute missing values with the median of each column

CODE:

```
df = df.drop(columns=["Unnamed: 32"])
label_encoder = LabelEncoder()
df['diagnosis'] = label_encoder.fit_transform(df['diagnosis'])
df = df.rename(columns={'diagnosis':'Label'})
imputer = SimpleImputer(strategy='median')
df[df.columns] = imputer.fit_transform(df[df.columns])
```

4. **Visualize Class Distribution**

CODE:

```
sns.countplot(x="Label", data=df)
```

5. **Prepare Data for Modeling**
    - Define dependent (y) and independent (X) variables
    - Convert DataFrame to NumPy array

CODE:

```
y = df["Label"].values
X = df.drop(labels=["Label", "id"], axis=1).to_numpy()
```

6. **Cross-Validation and Model Training**
    - Define cross-validation procedure
    - Create and evaluate SVM model
    - Scale data before evaluation

CODE:

```
cv = KFold(n_splits=7, random_state=42, shuffle=True)
model = svm.SVC(kernel='linear', C=1, random_state=42)
scaler = MinMaxScaler()
X_array = scaler.fit_transform(X_array)
scores = cross_val_score(model, X_array, y, scoring='accuracy', cv=cv, n_jobs=-1)
for score in scores:
   print("Accuracy for this fold is: ", scores)
print('Mean accuracy over all folds is: ', np.mean(scores))
```

7. **Pipeline Implementation**
    - Define a pipeline to include scaling and model training
    - Evaluate the pipeline using cross-validation

CODE:

```
steps = [('scaler', MinMaxScaler()), ('model', svm.SVC(kernel='linear', C=1, random_state=42))]
pipeline = Pipeline(steps=steps)
scores2 = cross_val_score(pipeline, X_array, y, scoring='accuracy', cv=cv, n_jobs=-1)
for score in scores2:
   print("Accuracy for this fold is: ", scores2)
print('Mean accuracy over all folds is: ', np.mean(scores2))
```

Highlights

- **Cross-Validation:** The use of KFold cross-validation ensures that the model's performance is evaluated on different subsets of the data, providing a more reliable estimate of its accuracy.
- **Pipeline Implementation:** Creating a pipeline that includes both data scaling and model training ensures that all preprocessing steps are applied consistently, preventing data leakage and improving model robustness.

- **Handling Missing Values:** Using median imputation for missing values ensures that the dataset remains complete without introducing bias.

Conclusion

The project successfully demonstrates the process of classifying breast cancer tumors using an SVM model. The use of cross-validation and a pipeline ensures robust and consistent model performance. The model achieves a mean accuracy that indicates its potential effectiveness in real-world applications for breast cancer diagnosis.