Machine Learning Tutorial

Student Name : Rahul Yadav Suresh
Student ID : 23072902

Introduction

- Overview of Machine Learning: Machine learning is a field of artificial intelligence that enables computers to learn from data and make predictions or decisions without being explicitly programmed. It has applications in various domains, including healthcare, finance, and technology.

- Objective: The goal of this tutorial is to build a machine learning model to classify breast cancer using Support Vector Machines (SVM). We will use the breast cancer dataset and cover various steps including data preprocessing, feature scaling, PCA for visualization, model training with hyperparameter tuning using Research, and model evaluation.

---

1. Theoretical Background

What is a Support Vector Machine?(SVM)

Support Vector Machines (SVM) are supervised machine learning algorithms used for classification and regression tasks. They are particularly well-known for their effectiveness in high-dimensional spaces and their ability to handle both linear and non-linear data.

How SVM Works

The core idea of SVM is to find a hyperplane that best separates the classes in the feature space. The hyperplane is chosen to maximize the margin between the classes, which is the distance between the hyperplane and the nearest data points from each class. These nearest data points are called support vectors.

- **Hyperplane**: A decision boundary that separates different classes.
- **Support Vectors**: Data points closest to the hyperplane that influence its position and orientation.
- **Margin**: The distance between the hyperplane and the support vectors. A larger margin is preferred as it indicates better separation between classes

Types of SVM

1. **Linear SVM**

    - **Description**: Linear SVM is used when the data is linearly separable, meaning a single straight line (or hyperplane in higher dimensions) can separate the classes.
    - **Use Case**: Ideal for datasets where classes can be separated with a straight line.
    - **Example**: Classifying emails as spam or not spam based on word frequency.
    -

2. **Non-Linear SVM**

   Description: Non-Linear SVM is used when the data is not linearly separable. It uses kernel functions to transform the data into a higher-dimensional space where a hyperplane can be used to separate the classes.

Kernel Functions:

1. **Linear Kernel**: it computes the similarity between two data points using a simple dot product.
2. **Polynomial Kernel**: Transforms the data into a higher-dimensional space using polynomial functions.
3. **Radial Basis Function (RBF) Kernel**: Also known as the Gaussian kernel, it maps the data into an infinite-dimensional space.
4. **Sigmoid Kernel**: Uses the sigmoid function to map the data into a higher-dimensional space

2. Experiment Setup

SVM in Breast Cancer Classification

Model Overview

In the breast cancer classification model, SVM is used to classify tumors as either malignant (cancerous) or benign (non-cancerous) based on various features extracted from digitized images of fine needle aspirate (FNA) of breast mass.

Dataset and Preprocessing

Data set link**: (**[Kaggle]) click here

The **Wisconsin Breast Cancer dataset** contains features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, describing characteristics of cell nuclei.

The Breast Cancer dataset consists of **diagnostic measurements** collected from tumor cells. The target variable is **`diagnosis`:**

- **M (Malignant) = 1** (Cancerous)
- **B (Benign) = 0** (Non-cancerous)

Features in the Dataset

- **ID Column**: Dropped as it's not useful for classification.
- **30 Numerical Features**: Representing tumor cell characteristics (radius, texture, smoothness, compactness, etc.).
- **Target Variable (`diagnosis`)**: Indicates whether cancer is **Malignant (1)** or **Benign (0)**.

Key Preprocessing Steps :

Code Snippet:

```
# Load the dataset
```

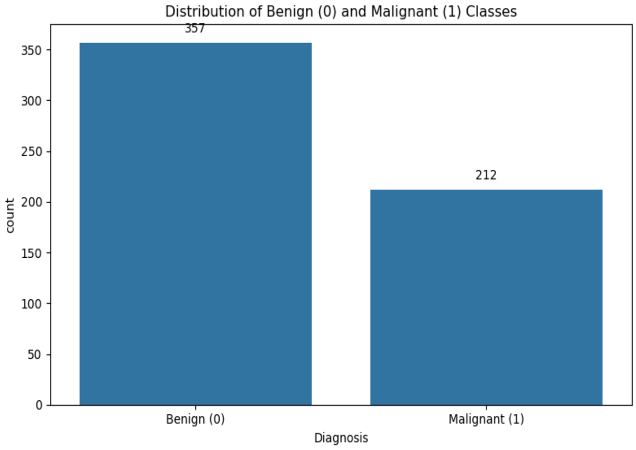
```
data = pd.read_csv ("/Users/rahulyadav/Desktop/breast-cancer.csv")

# Drop ID column and convert labels
data.drop(columns=["id"], inplace=True)
data["diagnosis"] = data["diagnosis"].map({"M": 1, "B": 0})

# Separate features and target variable
X = data.iloc[:, 1:]  # Features
y = data.iloc[:, 0]   # Target variable (Diagnosis)

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

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

3. Class Distribution

Here's the class distribution in the dataset:

- **Benign (0)**: 357 samples
- **Malignant (1)**: 212 samples

Visualizing the Class Distribution:*To better understand the class distribution, we can visualize it using a bar chart:*



- **Impact on Model:** This imbalance may negatively affect model performance, especially when evaluating metrics like recall and F1-score for the "Malignant" class. The model might struggle to correctly identify Malignant cases, which are crucial in medical diagnosis.

- **SMOTE Implementation**: To address this imbalance, techniques like **SMOTE** (Synthetic Minority Over-sampling Technique) can be used to generate synthetic samples for the minority class, thereby improving the model's ability to detect Malignant cases.

Code snippet :

```
# Apply SMOTE to balance the dataset
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)
```

4. Kernel Comparison and Hyperparameter Tuning

To identify the best kernel and hyperparameters for the Breast Cancer Classification model, I perform a **grid search** for each kernel type. The goal is to tune the model's hyperparameters to achieve the best performance on the dataset.

Key Hyperparameters:

- **C (Regularization Parameter)**: This parameter controls the trade-off between minimizing training error and minimizing testing error. A higher value of **C** prioritizes minimizing training error but may lead to overfitting, while a lower **C** allows for better generalization by smoothing the decision boundary.
- **Gamma**: Defines the influence of a single training example. A higher **gamma** value creates a complex decision boundary, potentially overfitting, while a lower **gamma** makes the model more generalized.

Grid Search with Cross-Validation:

To determine the best combination of hyperparameters, A grid search was performed. This method performs an exhaustive search over the parameter grid (C and gamma values) for each kernel type. It uses **5-fold cross-validation** to evaluate model performance, ensuring a robust selection of hyperparameters that generalize well.

Code snippet:

```
# Define the SVM model and parameters
parameters = {'C': [0.1, 1, 10], 'gamma': [0.01, 0.1, 1, 10]}
kernels = ['linear', 'poly', 'rbf', 'sigmoid']

# Initialize a dictionary to store results
best_models = {}

# Perform GridSearchCV for each kernel:
model = SVC(kernel=kernel)
grid = GridSearchCV(model, parameters, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)
best_models[kernel] = {'params': grid.best_params_, 'score':
grid.best_score_}
```

---

5. Evaluation and Visualization

After training our SVM model with the best hyperparameters, it’s time to assess its performance and visualize its decision-making.

Performance Metrics

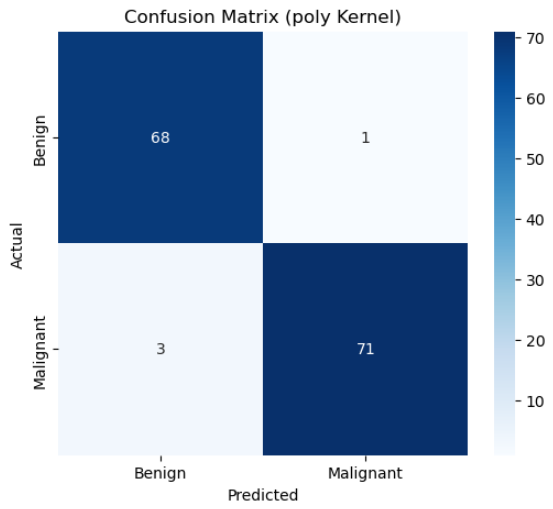
- **Accuracy:** Measures the overall correctness of the model.

- **Confusion Matrix:** Shows how many benign/malignant cases were correctly or incorrectly classified.
- **Classification Report:** Provides precision, recall, and F1-score for each class.

Confusion Matrix : A confusion matrix provides a **visual snapshot** of the model's performance. *It helps us identify:* How many benign cases were misclassified as malignant (**false positives**) and How many malignant cases were missed (**false negatives**).

- **Top-left (True Negatives, TN):** Correctly classified benign cases.
- **Top-right (False Positives, FP):** Benign cases misclassified as malignant (bad for patients!).
- **Bottom-left (False Negatives, FN):** Malignant cases misclassified as benign (very dangerous!).
- **Bottom-right (True Positives, TP):** Correctly classified malignant cases.

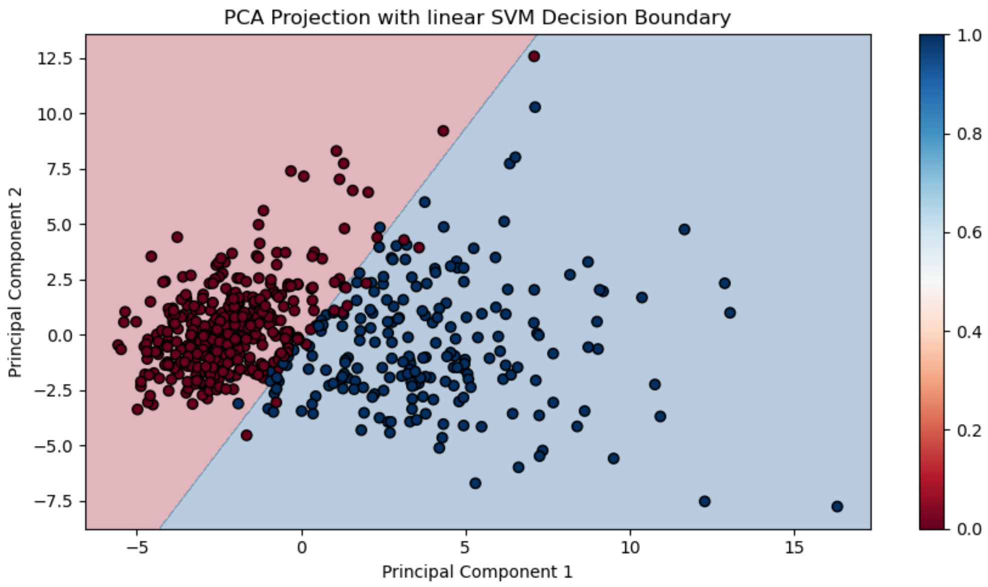
A **low number of FN (False Negatives)** is **critical in cancer detection**, as missing a malignant case can have serious consequences.



PCA Visualization with Decision Boundaries:

We use **Principal Component Analysis (PCA)** to reduce the features to **2 dimensions** and plot the **decision boundary** of the SVM. **Why PCA?**

- Reduces dimensionality while retaining important patterns.
- Helps visualize how SVM draws a boundary between benign and malignant cases.



Why is the Best Kernel Linear for PCA? The **Linear Kernel** performed best after PCA because PCA preserves the dataset's linearity—if the data was already separable with a straight line, reducing it to 2D wouldn't change that. Non-linear kernels like RBF and Poly work better for curved decision boundaries, but PCA's transformation may have simplified the data, removing complex relationships. Additionally, PCA can cause some information loss, meaning any non-linearity present before transformation might have been oversimplified.

---

6. Practical Considerations

Choosing the Right Kernel: Selecting the right kernel is key to SVM performance:

- **Linear Kernel** → Ideal when data is linearly separable.
- **Polynomial Kernel** → Captures structured, non-linear patterns.
- **RBF Kernel** → Best for highly complex, non-linear relationships.
- **Sigmoid Kernel** → Occasionally effective in NLP applications.

Handling Class Imbalance: Medical datasets often have imbalanced classes, which can bias the model. To counter this*:*

- **Use class_weight='balanced'** in SVM to adjust for uneven class distribution.
- **Apply resampling techniques** like SMOTE to synthetically balance the dataset.
- **Leverage ensemble methods**, integrating SVM as a base learner for robustness.

Model Justification

Because SVM can handle high-dimensional data and identify the best hyperplane, it is selected. On huge datasets, nevertheless, it can be computationally costly. Although they are computationally quicker, alternative models like logistic regression could have trouble with data that is not linearly separable.

---

7. Conclusion:

In this tutorial, I've walked through the process of using Support Vector Machines (SVM) for breast cancer classification. By testing different kernels and tuning hyperparameters, I found that the Linear Kernel performed the best after PCA, as it naturally aligned with the data's linearity. Through visualizations and model evaluation, I gained a deeper understanding of how SVM classifies tumors as malignant or benign. This project not only sharpened my SVM skills but also helped me explore practical considerations like kernel selection and handling class imbalance, which are crucial in medical datasets. I'm excited to apply these learnings to real-world scenarios and continue improving my machine learning expertise.

GitHub Repository

Git hub link: (here)

The complete code, visualizations, and additional resources for this tutorial are available in the GitHub repository. The repository includes a detailed README file with instructions for reproducing the results, as well as an MIT license to facilitate reuse and modification.

Accessibility Considerations

This tutorial has been designed with accessibility in mind:

- **Colour-Blind Friendly Palettes:** All visualizations use color schemes that are accessible to users with color vision deficiencies.
- **Alternative Text for Figures:** Each figure includes descriptive alternative text to ensure that visually impaired users can understand the content.
- **Screen Reader Compatibility:** The text structure is optimized for screen readers, making it easier to navigate and comprehend.
- **Well-Documented Code:** The provided code snippets include clear comments explaining each step, improving readability and ease of reuse for all users

Reference

- **UCI Machine Learning Repository – Breast Cancer Dataset** – *https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)*
- **Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks."** ***Machine Learning, 20*****(3), 273–297. Access the paper here.**
- **Pattern Recognition and Machine Learning –** *Christopher M. Bishop* **(Access here),**
  A great introduction to **SVM** and **Contains visual explanations and practical applications.**
- **SMOTE Implementation Guide –** ***https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html***
  Covers how to handle class imbalance using SMOTE.