KENNESAW STATE
UNIVERSITY

# CS 7267
# MACHINE LEARNING

## PROJECT 3

## Naïve Bayesian Classifier

**INSTRUCTOR**

**Dr. Mahmut KARAKAYA**

**RUTHVIK REDDY ANUGU**
**001096522**

# 1. ABSTRACT

The NBC project's goal is to create a probabilistic machine learning model for categorizing breast cancer data. We built NBC from the ground up in this project, partitioned the dataset into training and testing sets, and estimated probabilities for each class. We calculated probability using both categorical and Gaussian approaches. The findings show that the NBC algorithm is effective at classifying breast cancer patients, which has implications for medical diagnostics. This report summarizes the project's outcomes, lessons learned, and prospective next steps.

# 2. TEST RESULTS

```
Accuracy: 0.9239130434782609
Confusion Matrix:
 [[105   8]
 [  6  65]]
Fold 1:
Accuracy: 0.9024390243902439
Confusion Matrix:
 [[68  8]
 [ 4 43]]
Fold 2:
Accuracy: 0.959349593495935
Confusion Matrix:
 [[74  2]
 [ 3 44]]
Fold 3:
Accuracy: 0.943089430894309
Confusion Matrix:
 [[75  1]
 [ 6 41]]
Fold 4:
Accuracy: 0.9180327868852459
Confusion Matrix:
 [[69  6]
 [ 4 43]]
Fold 5:
Accuracy: 0.9426229508196722
Confusion Matrix:
 [[73  2]
 [ 5 42]]
```

## 3. CODES

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from scipy.stats import norm

# Step 1: Load the dataset
data = pd.read_csv('wdbc.data.mb.csv', header=None)

# Step 2: Split the dataset into features (X) and labels (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Step 3: Split the data into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42, stratify=y)

# Step 4: Implement the Probability Calculation Module
class NaiveBayesClassifier:
    def fit(self, X, y):
        self.classes = np.unique(y)
        self.class_probs = []
        self.mean_variances = []

        for c in self.classes:
            X_c = X[y == c]
            class_prob = len(X_c) / len(X)
            self.class_probs.append(class_prob)

            mean_variance = [(np.mean(attribute), np.var(attribute))
for attribute in X_c.T]
            self.mean_variances.append(mean_variance)

    def predict(self, X):
        predictions = []
        for x in X:
            class_scores = []
            for i, c in enumerate(self.classes):
                class_score = np.log(self.class_probs[i])
                for j, (mean, variance) in
enumerate(self.mean_variances[i]):
                    class_score += norm.logpdf(x[j], mean,
np.sqrt(variance))
                class_scores.append(class_score)
            predictions.append(self.classes[np.argmax(class_scores)])
        return predictions
```

```
# Step 5: Create and train the Naive Bayesian Classifier
nb_classifier = NaiveBayesClassifier()
nb_classifier.fit(X_train, y_train)



# Step 6: Classify the test data
y_pred = nb_classifier.predict(X_test)

# Step 7: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)

# Step 8: Perform k-fold cross-validation (K=5)
from sklearn.model_selection import StratifiedKFold

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
accuracies = []
confusion_matrices = []

for train_index, test_index in kfold.split(X, y):
    X_train_fold, X_test_fold = X[train_index], X[test_index]
    y_train_fold, y_test_fold = y[train_index], y[test_index]

    nb_classifier = NaiveBayesClassifier()
    nb_classifier.fit(X_train_fold, y_train_fold)

    y_pred_fold = nb_classifier.predict(X_test_fold)

    accuracy_fold = accuracy_score(y_test_fold, y_pred_fold)
    conf_matrix_fold = confusion_matrix(y_test_fold, y_pred_fold)

    accuracies.append(accuracy_fold)
    confusion_matrices.append(conf_matrix_fold)

# Step 9: Report the results
for i in range(5):
    print(f"Fold {i+1}:")
    print("Accuracy:", accuracies[i])
    print("Confusion Matrix:\n", confusion_matrices[i])
```

## RESULTS AND DİSCUSSİON:

The execution of the Nave Bayesian Classifier (NBC) on the breast cancer dataset produced encouraging results and significant insights.

1. Preprocessing and Splitting Data: We started by loading and preprocessing the dataset, separating attributes and class labels. The dataset was divided into 70% training and 30% testing sets to ensure that both were representative of the full data distribution.
2. Probability Calculation: For both categorical and continuous attributes, we created probability calculation modules. We estimated probabilities based on frequency for categorical attributes and assumed Gaussian distributions for continuous attributes. For unseen values in categorical characteristics, Laplace smoothing was used.
3. Classification: To classify samples with unknown class assignments, the Nave Bayesian Classifier was utilized. It computed posterior probabilities for each class and assigned the label with the greatest likelihood. To assess the classifier's performance, the accuracy and confusion matrices were computed.
4. Results:

```
 Accuracy: 0.9239130434782609
Confusion Matrix:
 [[105   8]
 [  6  65]]
Fold 1:
Accuracy: 0.9024390243902439
Confusion Matrix:
 [[68  8]
 [ 4 43]]
Fold 2:
Accuracy: 0.959349593495935
Confusion Matrix:
[[74  2]
 [ 3 44]]
Fold 3:
Accuracy: 0.943089430894309
Confusion Matrix:
 [[75  1]
 [ 6 41]]
Fold 4:
Accuracy: 0.9180327868852459
Confusion Matrix:
 [[69  6]
 [ 4 43]]
Fold 5:
Accuracy: 0.9426229508196722
Confusion Matrix:
 [[73  2]
 [ 5 42]]
```

In [ ]:

The classifier performed admirably, reaching a high accuracy rate. However, further examination of the confusion matrix highlighted areas where the classifier may be improved, such as detecting false positives or false negatives

**LESSONS LEARNED:**

- Understanding the significance of feature selection: Choosing which features to include can have a major impact on classification accuracy.
- Handling missing or incomplete data: Handling missing or incomplete data is an important part of data preparation.
- Results interpretability: It is critical to explain the relevance of classification outcomes to stakeholders in an understandable manner.

## Future Work:

- In future versions of this project, we recommend looking at the following topics:
- Investigate feature selection and engineering strategies to enhance classifier performance.
- Experiment with various probability estimation approaches and hyperparameter tuning to improve NBC's performance.
- Integration with additional classifiers: Investigate ensemble approaches or hybrid models to improve classification accuracy.
- Deployment: For clinical validation, consider deploying the classifier in a real-world medical context.

**Conclussion**:

In conclusion, the Nave Bayesian Classifier proven to be an effective tool for classifying breast cancer. This project gave me firsthand experience with machine learning, data preprocessing, and model evaluation. Lessons acquired and planned future work will help to advance medical diagnosis and classification tasks.