

**Tayaba Mahmood**

# **Report for KNN, Gaussian Naive Bayes, MLP, and Simple Neural Network Classifiers**

## ****Introduction****

This report explores three popular machine learning classifiers: K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GNB), and a Neural Network-based Multi-Layer Perceptron (MLP). Each model was implemented to classify data from the given dataset (/SVMtrain.csv), focusing on their accuracy and performance under various configurations. Additionally, a custom neural network implementation is included to provide deeper insights into how hyperparameters, such as the number of layers and neurons, affect model performance.

## ****Dataset Overview****

The dataset consists of the following columns:  
PassengerId, Survived, Pclass, Sex, Age, SibSp, Parch, Fare, and Embarked.

**Preprocessing Steps:**

1. **Encoding Categorical Data**: Using LabelEncoder, all categorical features were converted into numerical values to ensure compatibility with the models.
2. **Feature-Target Split**: Features (X) were separated from the target variable (y), which is the Embarked column.
3. **Train-Test Split**: The dataset was divided into training and testing sets (80% training, 20% testing) for model evaluation.
4. **Standardization**: Feature scaling was performed using StandardScaler to normalize the data for models that are sensitive to feature magnitudes (e.g., MLP).

## ****Classifiers and Results****

### 1. ****K-Nearest Neighbors (KNN)****

**Implementation Details**:

* **Hyperparameters**: Number of neighbors k=5 (default).
* Distance metric: Euclidean distance.
* Predicts the class of a sample based on the majority class of its nearest neighbors.

**Results**:

* Accuracy: X% (replace with the actual result).

**Analysis**:

* KNN performs well with simpler datasets but can struggle with large feature spaces or imbalanced data.
* Increasing k reduces sensitivity to noise but might lead to underfitting.

### 2. ****Gaussian Naive Bayes (GNB)****

**Implementation Details**:

* Assumes features are normally distributed (Gaussian distribution).
* Probabilistic model, calculates the posterior probability for each class.

**Results**:

* Accuracy: Y% (replace with the actual result).

**Analysis**:

* GNB is computationally efficient and works well with categorical or numerical data.
* Performs well on small datasets but can be inaccurate if the normality assumption is violated.

### 3. ****Multi-Layer Perceptron (MLP)****

**Implementation Details**:

* **Architecture**: Three hidden layers with 5, 3, and 2 neurons, respectively.
* **Activation Function**: ReLU for hidden layers, softmax for the output layer.
* **Optimization**: Backpropagation with stochastic gradient descent.
* **Feature Scaling**: StandardScaler applied.

**Results**:

* Accuracy: Z% (replace with the actual result).

**Analysis**:

* MLP can handle complex non-linear relationships between features.
* Performance depends heavily on the architecture (number of layers, neurons) and hyperparameters (learning rate, epochs, etc.).

### 4. ****Custom Neural Network****

**Implementation Details**:

* A fully connected neural network implemented from scratch.
* One hidden layer with 10 neurons.
* Uses sigmoid activation for hidden layers and softmax for the output.

**Results**:

* Accuracy: W% (replace with the actual result).

**Analysis**:

* Custom implementation demonstrates the mechanics of forward propagation, backpropagation, and weight updates.
* Visualizations of training loss show how the model learns over time.

## ****Comparison of Classifiers****

| **Classifier** | **Accuracy** | **Strengths** | **Weaknesses** |
| --- | --- | --- | --- |
| K-Nearest Neighbors (KNN) | X% | Simple to understand and implement, non-parametric. | Sensitive to feature scaling, performance decreases with high dimensions. |
| Gaussian Naive Bayes (GNB) | Y% | Computationally efficient, handles categorical data well. | Assumes feature independence, sensitive to incorrect normality assumptions. |
| Multi-Layer Perceptron (MLP) | Z% | Handles non-linear relationships, customizable architecture. | Requires significant computational resources and hyperparameter tuning. |
| Custom Neural Network | W% | Demonstrates inner workings of NNs, flexibility in design. | Less optimized compared to library implementations. |

## ****Impact of Hyperparameters on MLP****

### ****Number of Layers****:

* Adding more layers increases the model's capacity to learn complex patterns but can lead to overfitting if the dataset is small or lacks diversity.
* Decreasing layers simplifies the model, making it faster but potentially underpowered for non-linear relationships.

### ****Number of Neurons****:

* Increasing neurons per layer can improve accuracy up to a point, after which performance plateaus or decreases due to overfitting.
* Too few neurons may limit the model's ability to capture patterns in the data.

### ****Activation Function****:

* ReLU is preferred for hidden layers due to its simplicity and ability to mitigate the vanishing gradient problem.
* Softmax is ideal for multi-class classification in the output layer.

## ****Visualizations****

### ****Training Loss****

The custom neural network shows how the loss decreases over epochs, indicating successful learning.  
Plot included in the code above.

### ****True vs. Predicted Labels****

A scatter plot provides a comparison of the true and predicted labels, highlighting the model's strengths and weaknesses in classification.  
Plot included in the code above.

## ****Conclusion****

* **Best Performing Model**: [Insert best-performing model based on results].
* The choice of classifier depends on the problem at hand, dataset size, and computational resources.
* Fine-tuning hyperparameters, such as the number of layers, neurons, and learning rate, can significantly improve MLP performance.

This report showcases a comprehensive comparison of classifiers, emphasizing their strengths, weaknesses, and suitability for various types of data. Further experimentation with advanced techniques like regularization and dropout could enhance model robustness.



