Machine Learning Project

1) *Introduction:*

First, I **imported** all the libraries that I will need in the project (sklearn, seaborn, matplotlib, numpy, pandas, tensorflow, random, keras, collections).

After that, I read the data and **dropped** the column named "subject" because its data type is a string.

The data consists of one million rows, so **I took a sample** from the data because its size is very large. There are 13 classes ranging from 0 to 12. There are 900,000 samples from class number 0, and the remaining classes are distributed among the remaining rows, this indicates that the **data is imbalanced**.

To solve this problem, I took **8000 rows** from each class, leaving us with 104,000 rows.

After that, I **shuffled** the data and cleaned it by **dropping** the duplicates and null values. After that I visualized the data with a heat map to show the correlations between features.

2) *Splitting the data:*

I divided the data into features and target, placing the **features** in a variable named 'x' and the **target** in a variable named 'y'. Then performed scaling on the features and divided the data into training and testing sets.

3) *Models and there Evaluation metrics:*

Note: Before each model i made **Hyperparameter** tuning to get the best parameters and best Accuracy except for the **Linear Regression** model

A) KNN:

First Before I train the model I have **Normalized** the data to make the training better.

- **Cross Validation:** 0.9655
- **Accuracy:** 0.9651
- **Precision:** 0.9658
- **Recall:** 0.9651
- **F1 Score:** 0.9651
- **Mean Squared Error (MSE):** 0.3508

- These numbers are relatively **high**, and that's a good thing. The KNN model classifies data points into different classes based on the majority vote of their K nearest neighbors. I have chosen the K nearest neighbor number = 3 based on the **hypertuning.**
  From the **Strengths** of KNN that it is a **simple** model and doesn't take time in Training.
  And from its **Weakness** that it is sensitive to **outliers** and if there is outlier in the data the model will make very poor performance.

B)Linear Regression:

- **Root mean square error (RMSE):** 3.2936

- This is a **large** number for error, which is normal because the linear regression model doesn't work for **classification** but rather for continuous values and regression and predictions. Therefore, this data is **not suitable** for this model.
  Form the **Strengths** of Linear Regression can be trained quickly, even on **large datasets**. The computational complexity is relatively low compared to other algorithms, making it suitable for large-scale applications.
  And the **Weakness** is that linear regression models are prone to overfitting, especially when the number of features is large relative to the number of observations. Regularization techniques like Ridge regression or Lasso

regression can help mitigate this issue.

C)SVM:

- **Cross Validation:** 0.9741
- **Accuracy:** 0.9746
- **Precision:** 0.9753
- **Recall:** 0.9746
- **F1 Score:** 0.9746
- **Mean Squared Error (MSE):** 0.2431

- These numbers and accuracy are the **Good** obtained from the SVM model I used C = 20 and gamma = 5. SVM can **handle overfitting**, especially in high-dimensional spaces, due to its ability to maximize the margin between classes. This margin maximization helps in generalizing well to unseen data. And can efficiently handle **nonlinear** relationships between features using the **kernel** trick. By mapping the input features into a higher-dimensional space, SVM can find complex decision boundaries that separate classes effectively.
  But SVM takes so much time in Training and Processing.

D) Neural Network (NN):

Notes:

- **Input layer:** 12 Neurons because we have 12 features. ● **Hidden layers:** we have 4 hidden layers, the first hidden layer has 32 Neurons and the other three layers have 128 Neurons. And all hidden layers using the **Relu** activation function.
- **Output layer:** 13 Neurons Because we have 13 classes and we used **Sigmoid** activation function.
- **Optimizer:** Adam with 0.001 Learning rate
- **Epochs:** 100

- **Accuracy:** 0.9776
- **Precision:** 0.8209
- **Recall:** 0.8425
- **F1 Score:** 0.8209
- **Mean Squared Error (MSE):** 2.6145

- This accuracy is very high, which is a good thing. This is because the neural network is suitable for this data, and most of the time it yields high accuracy. Neural networks can learn complex patterns and relationships in data, making them highly flexible and adaptable to a wide range of tasks, including classification and regression. But it often requires large amounts of labeled data to train effectively.

E) Logistic Regression:

- **Cross Validation:** 0.6833
- **Accuracy:** 0.6855
- **Precision:** 0.6740
- **Recall:** 0.6855
- **F1 Score:** 0.6720
- **Mean Squared Error (MSE):** 9.2185

- Logistic regression produces interpretable results, as the coefficients represent the relationship between the independent variables and the log odds of the dependent variable. This makes it easy to understand the impact of each feature on the outcome.
  But it works well on small datasets only while in the huge dataset like MHEALTH its performance will be very poor

4) Conclusion:

In the end, we discovered that the Neural Network and SVM are the best models with the highest accuracy. This happened because they are the most suitable model for this data. Because they can learn complex patterns and relationships in data, making them highly flexible and adaptable to a wide range of tasks, including classification and regression.