***Human Stress Level Detection Model Training***

This document provides a detailed explanation of the steps followed in the source code to train a machine learning model for human stress level detection.

**1. Import Required Libraries**

The necessary libraries for data manipulation, visualization, and model training are imported.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

import plotly.express as px

**2. Load the Dataset**

The dataset is loaded into a pandas Data Frame.

data = pd.read\_csv(“Stress-Lysis.csv”)

**3. Data Pre-Processing & Feature Selection**

**Data Cleaning**

The missing values in the dataset are identified and filled using the forward fill method.

print("Missing values before cleaning:\\n", data.isnull().sum())

data.fillna(method='ffill', inplace=True)

print("Missing values after cleaning:\\n", data.isnull().sum())

**Noisy Data & Removal of Outliers**

Outliers are removed using the z-score method.

from scipy import stats

z\_scores = np.abs(stats.zscore(data.select\_dtypes(include=[np.number])))

data = data[(z\_scores < 3).all(axis=1)]

**Transforming Categorical Variables into Numerical Variables**

Categorical variables are transformed into numerical variables using Label Encoding.

label\_encoders = {}

for column in data.select\_dtypes(include=[object]).columns:

le = LabelEncoder()

data[column] = le.fit\_transform(data[column])

label\_encoders[column] = le

**4. Data Visualization**

Several visualizations are created to understand the data better.

**Bar Chart**

A bar chart is plotted for the stress level variable.

plt.subplot(3, 2, 1)

data['Stress Level'].value\_counts().plot(kind='bar')

plt.title('Bar Chart of Stress Level')

**Heat Map**

A heat map is created to show the correlation between features.

plt.subplot(3, 2, 2)

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

plt.title('Heat Map of Features')

**Histogram**

Histograms of all features are plotted.

plt.subplot(3, 2, 3)

data.hist(figsize=(12, 8))

plt.title('Histograms of Features')

**Pie Chart**

A pie chart is plotted for the stress level variable.

plt.subplot(3, 2, 4)

data['Stress Level'].value\_counts().plot(kind='pie', autopct='%1.1f%%')

plt.title('Pie Chart of Stress Level')

**Treemap**

A treemap is created for the stress level variable.

plt.subplot(3, 2, 5)

fig = px.treemap(data, path=[px.Constant("all"), 'Stress Level'])

fig.update\_traces(root\_color="lightgrey")

fig.show()

**5. Splitting and Training the Data**

The data is split into training (80%) and testing (20%) sets.

X = data.drop('Stress Level', axis=1)

y = data['Stress Level']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**6. Load the Model & Fit the Training Data**

A RandomForestClassifier model is loaded and trained with the training data.

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

**7. Evaluating the Model**

The model's performance is evaluated, and its accuracy is checked.

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy \* 100:.2f}%')

if accuracy < 75:

print('Accuracy is below 75%. Consider changing the algorithm or tuning parameters.')

# Additional Evaluation Metrics

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**8. Build the Predictive Model**

The model is built and trained during the previous steps.

**9. Deploy the Model**

The trained model is saved for deployment.

import joblib

joblib.dump(model, 'Stress Level\_model.pkl')

print('Model has been saved and is ready for deployment.')

**Conclusion**

The provided code and documentation outline a comprehensive approach to training a machine learning model for human stress level detection. By following a systematic process of data pre-processing, visualization, model training, and evaluation, we ensure the model's robustness and accuracy. The use of a RandomForestClassifier allows for effective handling of the dataset's complexities, and the final model, achieving satisfactory accuracy, is saved for deployment. This workflow not only highlights key data science techniques but also offers a practical solution to predicting stress levels, paving the way for potential real-world applications in health monitoring and stress management. The accompanying documentation ensures clarity and reproducibility, making it a valuable resource for future enhancements and deployments.