**Joint Tech Internship Community Program Organizer**

**Generative AI Consortium (MSME)**

**SystimaNX IT Solutions PVt Ltd.**

**TASK-2: Predicting Drug Effectiveness for Medical Conditions**

**Team Name**: Matrix Masters

**Team Member 1**: Vishnu Priya R

**GitLink:**

**Team Member 2:** Suganth A

GitLink:

**Team Member 3:** Shafeer Ahamed M

GitLink:

**Drug Prediction Process**

**1. Data Exploration**

- Objective: Get familiar with the dataset and understand its structure for predicting drug effectiveness.

- Steps:

- Load the dataset and examine its structure:

```python

import pandas as pd

df = pd.read\_csv('/content/drug\_prediction\_dataset.csv')

print(df.head())

print(df.info())

```

- Check for missing values:

```python

print(df.isnull().any())

```

**2. Data Preprocessing**

- Objective: Prepare the dataset for machine learning models.

- Steps:

- Remove irrelevant columns, such as patient IDs:

```python

df.drop('PatientID', axis=1, inplace=True)

df.drop('HospitalID', axis=1, inplace=True)

```

- Encode categorical features (e.g., drug types, conditions):

```python

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['DrugType'] = le.fit\_transform(df['DrugType'])

df['Condition'] = le.fit\_transform(df['Condition'])

df['Gender'] = le.fit\_transform(df['Gender'])

df['SideEffects'] = le.fit\_transform(df['SideEffects'])

```

- Normalize numerical features to ensure they are on a similar scale:

```python

from sklearn.preprocessing import MinMaxScaler

features\_to\_scale = ['Age', 'Dosage', 'Effectiveness']

scaler = MinMaxScaler()

df[features\_to\_scale] = scaler.fit\_transform(df[features\_to\_scale])

```

**3. Model Selection**

- Objective: Choose appropriate models to predict drug effectiveness.

- Steps:

- Define feature set and target variable:

```python

X = df.drop('Effectiveness', axis=1)

y = df['Effectiveness']

```

- Split data into training and testing sets:

```python

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

```

**4. Model Evaluation**

- Objective: Evaluate model performance based on accuracy.

- Steps:

- Implement K-Nearest Neighbors (KNN):

```python

from sklearn.metrics import accuracy\_score

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=7)

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

y\_pred\_knn = knn.predict(X\_test)

acc\_knn = round(accuracy\_score(y\_pred\_knn, y\_test) \* 100, 2)

print(f"KNN Accuracy: {acc\_knn}%")

```

- Try XGBoost for better performance:

```python

from xgboost import XGBClassifier

xgb = XGBClassifier()

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

y\_pred\_xgb = xgb.predict(X\_test)

acc\_xgb = round(accuracy\_score(y\_test, y\_pred\_xgb) \* 100, 2)

print(f"XGBoost Accuracy: {acc\_xgb}%")

```

**5. Visualizations**

- Understanding Data: Visualize key features (e.g., age distribution):

```python

import seaborn as sns

import matplotlib.pyplot as plt

sns.histplot(df['Age'], bins=10, kde=True)

plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.show()

```

- **Model Comparison:** Compare model performance with a bar chart:

```python

model\_names = ['KNN', 'XGBoost']

accuracies = [acc\_knn, acc\_xgb]

sns.barplot(x=model\_names, y=accuracies)

plt.title('Model Accuracy Comparison')

plt.ylabel('Accuracy (%)')

plt.ylim(0, 100)

plt.show()

```

**Conclusion**

This documentation explains the steps for data exploration, preprocessing, model selection, and evaluation in predicting drug effectiveness. Visualizations help to better understand the data and compare model performance effectively.