```
import pandas as pd
import random

# Lists to generate synthetic Indian data
names = ["Aarav", "Vihaan", "Aditya", "Sai", "Rohan", "Kabir",
"Arjun", "Ishaan", "Vivaan", "Reyansh",
         "Diya", "Saanvi", "Ananya", "Aadhya", "Kiara", "Pari",
"Riya", "Anika", "Myra", "Meera"]
surnames = ["Sharma", "Patel", "Verma", "Gupta", "Singh", "Kumar",
"Mehta", "Reddy", "Nair", "Iyer",
            "Joshi", "Kapoor", "Khan", "Das", "Chopra", "Malhotra",
"Bhat", "Saxena", "Rao", "Gowda"]
locations = ["Bangalore", "Mumbai", "Delhi", "Hyderabad", "Chennai",
"Pune", "Kolkata", "Ahmedabad", "Noida", "Gurgaon"]
degrees = ["B.Tech CS", "B.E. ECE", "MCA", "MBA", "B.Sc Stats",
"B.Com", "M.Tech", "B.Des", "BA English", "BCA"]
skills_pool = ["Python, SQL", "Java, Spring", "React, Node.js", "AWS,
Docker", "Excel, Tableau",
               "Sales, CRM", "Figma, Adobe XD", "HR, Recruiting", "C+
+, IoT", "Accounting, Tally"]

# Function to determine personality based on highest score
def get_personality(o, c, e, a, n):
    scores = {'Openness': o, 'Conscientiousness': c, 'Extraversion':
e, 'Agreeableness': a, 'Neuroticism': n}
    # Logic: Just taking the highest trait as the label for this
synthetic dataset
    max_trait = max(scores, key=scores.get)

    # Mapping traits to typical "Personality" labels found in CV
projects
    if max_trait == 'Conscientiousness': return 'Responsible'
    if max_trait == 'Extraversion': return 'Extroverted'
    if max_trait == 'Openness': return 'Lively'
    if max_trait == 'Agreeableness': return 'Dependable'
    if max_trait == 'Neuroticism': return 'Serious'
    return 'Serious'

data = []

for i in range(1, 501):
    # Generate Random Attributes
    fname = random.choice(names)
    lname = random.choice(surnames)
    full_name = f"{fname} {lname}"
    age = random.randint(21, 35)
    gender = "Female" if fname in ["Diya", "Saanvi", "Ananya",
"Aadhya", "Kiara", "Pari", "Riya", "Anika", "Myra", "Meera"] else
"Male"
    loc = random.choice(locations)
```

```
    deg = random.choice(degrees)
    exp = random.randint(0, 12)
    skill = random.choice(skills_pool)

    # Generate Big Five Scores (1-10 Scale)
    o = random.randint(1, 10) # Openness
    c = random.randint(1, 10) # Conscientiousness
    e = random.randint(1, 10) # Extraversion
    a = random.randint(1, 10) # Agreeableness
    n = random.randint(1, 10) # Neuroticism

    personality = get_personality(o, c, e, a, n)

    data.append([i, full_name, age, gender, loc, deg, exp, skill, o,
c, e, a, n, personality])

# Create DataFrame
df = pd.DataFrame(data, columns=["ID", "Name", "Age", "Gender",
"Location", "Degree", "Experience_Years",
                                 "Skills", "Openness",
"Conscientiousness", "Extraversion", "Agreeableness",
                                 "Neuroticism", "Personality_Label"])

# Save to Excel
file_name = "Indian_CV_Personality_Dataset.xlsx"
df.to_excel(file_name, index=False)

print(f"Successfully created {file_name} with 500 records.")
```

```
Successfully created Indian_CV_Personality_Dataset.xlsx with 500
records.
```

```
from google.colab import files

files.download('Indian_CV_Personality_Dataset.xlsx')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

Task

Load the `Indian_CV_Personality_Dataset.xlsx` file into a pandas DataFrame, then perform exploratory data analysis, preprocess the data by encoding categorical features and the target variable, split the data into training and testing sets, train a multi-class classification model, and finally evaluate the model's performance using appropriate metrics.

Load Dataset

Subtask:

Load the generated `Indian_CV_Personality_Dataset.xlsx` file into a pandas DataFrame.

```
import pandas as pd
import random

# Lists to generate synthetic Indian data
names = ["Aarav", "Vihaan", "Aditya", "Sai", "Rohan", "Kabir",
"Arjun", "Ishaan", "Vivaan", "Reyansh",
         "Diya", "Saanvi", "Ananya", "Aadhya", "Kiara", "Pari",
"Riya", "Anika", "Myra", "Meera"]
surnames = ["Sharma", "Patel", "Verma", "Gupta", "Singh", "Kumar",
"Mehta", "Reddy", "Nair", "Iyer",
            "Joshi", "Kapoor", "Khan", "Das", "Chopra", "Malhotra",
"Bhat", "Saxena", "Rao", "Gowda"]
locations = ["Bangalore", "Mumbai", "Delhi", "Hyderabad", "Chennai",
"Pune", "Kolkata", "Ahmedabad", "Noida", "Gurgaon"]
degrees = ["B.Tech CS", "B.E. ECE", "MCA", "MBA", "B.Sc Stats",
"B.Com", "M.Tech", "B.Des", "BA English", "BCA"]
skills_pool = ["Python, SQL", "Java, Spring", "React, Node.js", "AWS,
Docker", "Excel, Tableau",
               "Sales, CRM", "Figma, Adobe XD", "HR, Recruiting", "C+
+, IoT", "Accounting, Tally"]

# Function to determine personality based on highest score
def get_personality(o, c, e, a, n):
    scores = {'Openness': o, 'Conscientiousness': c, 'Extraversion':
e, 'Agreeableness': a, 'Neuroticism': n}
    # Logic: Just taking the highest trait as the label for this
synthetic dataset
    max_trait = max(scores, key=scores.get)

    # Mapping traits to typical "Personality" labels found in CV
projects
    if max_trait == 'Conscientiousness': return 'Responsible'
    if max_trait == 'Extraversion': return 'Extroverted'
    if max_trait == 'Openness': return 'Lively'
    if max_trait == 'Agreeableness': return 'Dependable'
    if max_trait == 'Neuroticism': return 'Serious'
    return 'Serious'

data = []

for i in range(1, 501):
    # Generate Random Attributes
    fname = random.choice(names)
```

```
    lname = random.choice(surnames)
    full_name = f"{fname} {lname}"
    age = random.randint(21, 35)
    gender = "Female" if fname in ["Diya", "Saanvi", "Ananya",
"Aadhya", "Kiara", "Pari", "Riya", "Anika", "Myra", "Meera"] else
"Male"
    loc = random.choice(locations)
    deg = random.choice(degrees)
    exp = random.randint(0, 12)
    skill = random.choice(skills_pool)

    # Generate Big Five Scores (1-10 Scale)
    o = random.randint(1, 10) # Openness
    c = random.randint(1, 10) # Conscientiousness
    e = random.randint(1, 10) # Extraversion
    a = random.randint(1, 10) # Agreeableness
    n = random.randint(1, 10) # Neuroticism

    personality = get_personality(o, c, e, a, n)

    data.append([i, full_name, age, gender, loc, deg, exp, skill, o,
c, e, a, n, personality])

# Create DataFrame
df_generation = pd.DataFrame(data, columns=["ID", "Name", "Age",
"Gender", "Location", "Degree", "Experience_Years",
                                "Skills", "Openness",
"Conscientiousness", "Extraversion", "Agreeableness",
                                "Neuroticism", "Personality_Label"])

# Save to Excel
file_name = "Indian_CV_Personality_Dataset.xlsx"
df_generation.to_excel(file_name, index=False)

print(f"Successfully created {file_name} with 500 records.")

# Load the dataset
df = pd.read_excel(file_name)
df.head()
```

```
Successfully created Indian_CV_Personality_Dataset.xlsx with 500
records.
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 500,\n  \"fields\": [\
n    {\n      \"column\": \"ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 144,\n        \"min\": 1,\n
\"max\": 500,\n        \"num_unique_values\": 500,\n
\"samples\": [\n          362,\n          74,\n          375\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 280,\n        \"samples\": [\n          \"Kiara
Joshi\",\n          \"Rohan Verma\",\n          \"Aarav Joshi\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 4,\n        \"min\":
21,\n        \"max\": 35,\n        \"num_unique_values\": 15,\n
\"samples\": [\n          28,\n          21,\n          31\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Male\",\n          \"Female\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Location\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
10,\n        \"samples\": [\n          \"Pune\",\n          \"Noida\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Degree\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"B.E. ECE\",\n          \"B.Sc Stats\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Experience_Years\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 12,\n
\"num_unique_values\": 13,\n        \"samples\": [\n          2,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Skills\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"AWS, Docker\",\n          \"Figma, Adobe
XD\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Openness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 10,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          2,\n          5\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Conscientiousness\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 10,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          8,\n          6\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Extraversion\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 10,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          4,\n          7\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
```

\"column\": \"Agreeableness\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 2,\n   \"min\": 1,\n
\"max\": 10,\n   \"num_unique_values\": 10,\n   \"samples\":
[\n   9,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Neuroticism\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 2,\n   \"min\": 1,\n
\"max\": 10,\n   \"num_unique_values\": 10,\n   \"samples\":
[\n   3,\n   7\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Personality_Label\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 5,\n
\"samples\": [\n   \"Serious\",\n   \"Dependable\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

Explore Data

Subtask:

Perform initial exploratory data analysis (EDA) to understand the dataset's structure, distributions, and potential issues like missing values. This includes looking at column types, unique values for categorical features, and the distribution of the target variable ('Personality_Label').

Reasoning: To begin the exploratory data analysis, I will display general information about the DataFrame, including data types and non-null counts, check for any missing values, show descriptive statistics for numerical columns, and then examine unique values and their counts for the specified categorical columns to understand their distribution.

```
print("DataFrame Info:")
df.info()

print("\nMissing Values:")
print(df.isnull().sum())

print("\nDescriptive Statistics for Numerical Columns:")
print(df.describe())

categorical_cols = ['Gender', 'Location', 'Degree', 'Skills',
'Personality_Label']
print("\nUnique Values and Counts for Categorical Columns:")
for col in categorical_cols:
    print(f"\n--- {col} ---")
    print("Unique Values:", df[col].unique())
    print("Value Counts:\n", df[col].value_counts())
```

```
DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID                 500 non-null    int64
 1   Name               500 non-null    object
 2   Age                500 non-null    int64
 3   Gender             500 non-null    object
 4   Location           500 non-null    object
 5   Degree             500 non-null    object
 6   Experience_Years   500 non-null    int64
 7   Skills             500 non-null    object
 8   Openness           500 non-null    int64
 9   Conscientiousness  500 non-null    int64
 10  Extraversion       500 non-null    int64
 11  Agreeableness      500 non-null    int64
 12  Neuroticism        500 non-null    int64
 13  Personality_Label  500 non-null    object
dtypes: int64(8), object(6)
memory usage: 54.8+ KB

Missing Values:
ID                   0
Name                 0
Age                  0
Gender               0
Location             0
Degree               0
Experience_Years     0
Skills               0
Openness             0
Conscientiousness    0
Extraversion         0
Agreeableness        0
Neuroticism          0
Personality_Label    0
dtype: int64

Descriptive Statistics for Numerical Columns:
                ID         Age  Experience_Years    Openness  \
count  500.000000  500.000000        500.000000  500.000000
mean   250.500000   27.954000          6.034000    5.712000
std    144.481833    4.232223          3.817065    2.902051
min      1.000000   21.000000          0.000000    1.000000
25%    125.750000   24.000000          3.000000    3.000000
50%    250.500000   28.000000          6.000000    6.000000
75%    375.250000   31.000000          9.000000    8.000000
max    500.000000   35.000000         12.000000   10.000000

       Conscientiousness  Extraversion  Agreeableness  Neuroticism
```

```
count         500.000000     500.00000      500.00000   500.000000
mean            5.618000       5.48800        5.42800     5.622000
std             2.856641       2.84993        2.82928     2.864879
min             1.000000       1.00000        1.00000     1.000000
25%             3.000000       3.00000        3.00000     3.000000
50%             6.000000       6.00000        5.00000     6.000000
75%             8.000000       8.00000        8.00000     8.000000
max            10.000000      10.00000       10.00000    10.000000

Unique Values and Counts for Categorical Columns:

--- Gender ---
Unique Values: ['Female' 'Male']
Value Counts:
 Gender
Male      288
Female    212
Name: count, dtype: int64

--- Location ---
Unique Values: ['Ahmedabad' 'Noida' 'Bangalore' 'Chennai' 'Kolkata'
'Mumbai' 'Delhi'
 'Gurgaon' 'Pune' 'Hyderabad']
Value Counts:
 Location
Delhi        59
Kolkata      59
Bangalore    58
Gurgaon      56
Noida        53
Ahmedabad    48
Pune         43
Mumbai       42
Chennai      41
Hyderabad    41
Name: count, dtype: int64

--- Degree ---
Unique Values: ['BCA' 'B.Sc Stats' 'MCA' 'M.Tech' 'BA English' 'B.Com'
'MBA' 'B.Des'
 'B.E. ECE' 'B.Tech CS']
Value Counts:
 Degree
M.Tech        64
MCA           54
B.Tech CS     53
MBA           52
B.Des         52
B.Sc Stats    49
B.Com         48
```

```
BA English    47
BCA           44
B.E. ECE      37
Name: count, dtype: int64

--- Skills ---
Unique Values: ['Java, Spring' 'Figma, Adobe XD' 'Excel, Tableau'
'Sales, CRM' 'C++, IoT'
 'Accounting, Tally' 'React, Node.js' 'HR, Recruiting' 'AWS, Docker'
 'Python, SQL']
Value Counts:
 Skills
Java, Spring         64
AWS, Docker          59
C++, IoT             55
Python, SQL          54
Sales, CRM           53
HR, Recruiting       48
Excel, Tableau       46
React, Node.js       44
Accounting, Tally    43
Figma, Adobe XD      34
Name: count, dtype: int64

--- Personality_Label ---
Unique Values: ['Lively' 'Serious' 'Responsible' 'Extroverted'
'Dependable']
Value Counts:
 Personality_Label
Lively         134
Responsible    106
Extroverted     95
Dependable      90
Serious         75
Name: count, dtype: int64
```

Preprocess Data

Subtask:

Prepare the data for machine learning. This will involve encoding categorical features such as 'Gender', 'Location', 'Degree', and 'Skills' into numerical representations, and also encoding the 'Personality_Label' for model training.

Reasoning: To preprocess the target variable 'Personality_Label' for model training, I will use `LabelEncoder` from `sklearn.preprocessing` to convert its categorical values into numerical representations. I will then perform one-hot encoding on the specified categorical feature columns ('Gender', 'Location', 'Degree', 'Skills') using `pd.get_dummies()`, and finally, combine all numerical features (including the encoded target and the original numerical

columns) into a single DataFrame, dropping irrelevant columns ('ID', 'Name') and the original categorical columns.

```
from sklearn.preprocessing import LabelEncoder

# 1. Label Encode 'Personality_Label'
le = LabelEncoder()
df['Personality_Label_Encoded'] =
le.fit_transform(df['Personality_Label'])

# Display the mapping of original labels to encoded integers
print("Mapping of Personality_Label to Encoded values:")
for i, label in enumerate(le.classes_):
    print(f"{label}: {i}")

# 2. Identify categorical feature columns for one-hot encoding
categorical_features = ['Gender', 'Location', 'Degree', 'Skills']

# 3. Apply one-hot encoding
df_encoded = pd.get_dummies(df, columns=categorical_features,
drop_first=True)

# 4. Drop original 'Personality_Label', 'ID', and 'Name' columns
df_processed = df_encoded.drop(columns=['ID', 'Name',
'Personality_Label'])


print("\nFirst 5 rows of the processed DataFrame:")
print(df_processed.head())
print("\nShape of the processed DataFrame:", df_processed.shape)
```

```
Mapping of Personality_Label to Encoded values:
Dependable: 0
Extroverted: 1
Lively: 2
Responsible: 3
Serious: 4

First 5 rows of the processed DataFrame:
   Age  Experience_Years  Openness  Conscientiousness  Extraversion  \
0   31                 0         9                  5             2
1   22                 0         5                  6             7
2   34                12         7                  1             6
3   35                 7         1                 10             6
4   31                 5         5                  5             1

   Agreeableness  Neuroticism  Personality_Label_Encoded  Gender_Male
\
0              8            8                          2        False

1              1            8                          4         True
```

```
2                4             7                            2        False

3                6             2                            3         True

4                1             1                            2         True


   Location_Bangalore  ...  Degree_MCA  Skills_Accounting, Tally  \
0               False  ...       False                    False
1               False  ...       False                    False
2                True  ...        True                    False
3               False  ...       False                    False
4               False  ...       False                    False

   Skills_C++, IoT  Skills_Excel, Tableau  Skills_Figma, Adobe XD  \
0            False                  False                   False
1            False                  False                    True
2            False                   True                   False
3            False                   True                   False
4            False                  False                   False

   Skills_HR, Recruiting  Skills_Java, Spring  Skills_Python, SQL  \
0                  False                 True                False
1                  False                False                False
2                  False                False                False
3                  False                False                False
4                  False                False                False

   Skills_React, Node.js  Skills_Sales, CRM
0                  False              False
1                  False              False
2                  False              False
3                  False              False
4                  False               True

[5 rows x 36 columns]

Shape of the processed DataFrame: (500, 36)
```

Reasoning: The data has been preprocessed and is now in a suitable format for machine learning. The next logical step is to split the data into training and testing sets to prepare for model training and evaluation. I will use `train_test_split` from `sklearn.model_selection` for this purpose.

```
from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df_processed.drop('Personality_Label_Encoded', axis=1)
y = df_processed['Personality_Label_Encoded']
```

```
# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)

print("\nDistribution of Personality_Label_Encoded in y_train:")
print(y_train.value_counts(normalize=True))

print("\nDistribution of Personality_Label_Encoded in y_test:")
print(y_test.value_counts(normalize=True))
```

```
Shape of X_train: (400, 35)
Shape of X_test: (100, 35)
Shape of y_train: (400,)
Shape of y_test: (100,)

Distribution of Personality_Label_Encoded in y_train:
Personality_Label_Encoded
2    0.2675
3    0.2125
1    0.1900
0    0.1800
4    0.1500
Name: proportion, dtype: float64

Distribution of Personality_Label_Encoded in y_test:
Personality_Label_Encoded
2    0.27
3    0.21
1    0.19
0    0.18
4    0.15
Name: proportion, dtype: float64
```

Reasoning: With the data successfully split into training and testing sets, the next step is to train a multi-class classification model. I will use a `RandomForestClassifier` as it is a robust and commonly used algorithm for this type of problem, and then fit it to the training data.

```
from sklearn.ensemble import RandomForestClassifier

# Initialize the RandomForestClassifier model
# Using random_state for reproducibility
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
```

```
model.fit(X_train, y_train)

print("RandomForestClassifier model trained successfully.")
```

```
RandomForestClassifier model trained successfully.
```

Reasoning: The model has been trained, so the next logical step is to evaluate its performance on the test data. I will use the trained model to make predictions on the `X_test` set and then calculate classification report to assess its accuracy, precision, recall, and F1-score.

```
from sklearn.metrics import classification_report, accuracy_score

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model's performance
print("Model Evaluation:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
```

```
Model Evaluation:
Accuracy: 0.88

Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.94      0.89        18
           1       0.88      0.79      0.83        19
           2       0.89      0.93      0.91        27
           3       0.95      0.95      0.95        21
           4       0.79      0.73      0.76        15

    accuracy                           0.88       100
   macro avg       0.87      0.87      0.87       100
weighted avg       0.88      0.88      0.88       100
```

Summary:

Q&A

The multi-class classification model, a RandomForestClassifier, achieved an accuracy of 88% on the test set.

Data Analysis Key Findings

- The dataset, `Indian_CV_Personality_Dataset.xlsx`, containing 500 records and 14 columns, was successfully loaded.
- Exploratory Data Analysis revealed no missing values across any columns.

- Numerical features like 'Age' (21-35 years) and 'Experience_Years' (0-12 years) showed typical distributions for the synthetic data.
- Personality trait scores ('Openness', 'Conscientiousness', 'Extraversion', 'Agreeableness', 'Neuroticism') ranged from 1 to 10, with means generally between 5 and 6, indicating a balanced distribution of these scores.
- The target variable, 'Personality_Label', consists of 5 unique types: 'Lively', 'Serious', 'Responsible', 'Extroverted', and 'Dependable'. There is a class imbalance, with 'Lively' being the most frequent (134 entries) and 'Serious' the least frequent (75 entries).
- Categorical features ('Gender', 'Location', 'Degree', 'Skills') and the target variable ('Personality_Label') were successfully encoded. 'Personality_Label' was label encoded, and other categorical features were one-hot encoded, resulting in a processed DataFrame with 35 feature columns.
- The data was split into training and testing sets (80% train, 20% test), maintaining the target variable's class distribution in both sets.
- A RandomForestClassifier model was trained and achieved an overall accuracy of 88% on the test set, demonstrating good performance in classifying the personality labels.

Insights or Next Steps

- Given the class imbalance in the target variable, further steps could involve exploring techniques like oversampling (e.g., SMOTE) or undersampling to balance the classes and potentially improve the model's performance for underrepresented personality types.
- Feature importance analysis from the RandomForestClassifier could provide insights into which attributes (e.g., specific skills, degrees, or Big Five traits) are most indicative of different personality labels, offering a deeper understanding of the relationships within the dataset.