```
import pandas as pd
from google.colab import files
import io

# Step 1: Upload the dataset
uploaded = files.upload()

# Get the uploaded file name
file_name = list(uploaded.keys())[0]

# Step 2: Load the dataset
df = pd.read_excel(io.BytesIO(uploaded[file_name]), sheet_name='autism')

# Step 3: Display missing values before processing
print("Missing values before processing:\n", df.isnull().sum())

# Step 4: Count "?" values before removal
print("\nCount of '?' in ethnicity:", (df['ethnicity'] == '?').sum())
print("Count of '?' in relation:", (df['relation'] == '?').sum())

# Step 5: Remove rows where 'ethnicity' or 'relation' contains "?"
df = df[(df['ethnicity'] != '?') & (df['relation'] != '?')]

# Step 6: Remove rows where 'age' is NaN
df = df.dropna(subset=['age'])

# Step 7: Remove rows where 'age' is greater than 80
df = df[df['age'] <= 80]

# Step 8: Display missing values after processing
print("\nMissing values after processing:\n", df.isnull().sum())

# Step 9: Save the cleaned dataset
cleaned_file_name = "cleaned_autism.xlsx"
df.to_excel(cleaned_file_name, index=False)

# Step 10: Download the cleaned dataset
files.download(cleaned_file_name)
print("\nDownload started: 'cleaned_autism.xlsx'")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving autism.xlsx to autism.xlsx
Missing values before processing:
 A1_Score          0
A2_Score          0
A3_Score          0
A4_Score          0
A5_Score          0
A6_Score          0
A7_Score          0
A8_Score          0
A9_Score          0
A10_Score         0
age               2
gender            0
ethnicity         0
jundice           0
austim            0
contry_of_res     0
used_app_before   0
result            0
relation          0
Class/ASD         0
dtype: int64

Count of '?' in ethnicity: 95
Count of '?' in relation: 95

Missing values after processing:
 A1_Score          0
A2_Score          0
A3_Score          0
A4_Score          0
A5_Score          0
A6_Score          0
A7_Score          0
A8_Score          0
A9_Score          0
A10_Score         0
age               0
gender            0
ethnicity         0
jundice           0
austim            0
contry_of_res     0
used_app_before   0
result            0
relation          0
Class/ASD         0
dtype: int64

Download started: 'cleaned_autism.xlsx'
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE

# Load the cleaned dataset
df = pd.read_excel("cleaned_autism.xlsx")

# Encode categorical variables
categorical_columns = ['gender', 'ethnicity', 'jundice', 'austim', 'contry_of_res', 'used_a
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Define features (X) and target variable (y)
X = df.drop(columns=['Class/ASD'])
y = df['Class/ASD']

# Standardize numerical features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split data into training (70%) and testing (30%)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state

# Apply SMOTE
smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
log_model = LogisticRegression(penalty='l2', C=0.01, solver='liblinear', max_iter=200)
log_model.fit(X_train, y_train)
y_test_pred_log = log_model.predict(X_test)
print("\nLogistic Regression Testing Accuracy:", accuracy_score(y_test, y_test_pred_log))
print("\nLogistic Regression Classification Report:\n", classification_report(y_test, y_tes
print("\nLogistic Regression Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred_log
```

```
Logistic Regression Testing Accuracy: 0.9347826086956522

Logistic Regression Classification Report:
               precision    recall  f1-score   support

          NO       1.00      0.91      0.95        66
         YES       0.81      1.00      0.90        26

    accuracy                           0.93        92
   macro avg       0.91      0.95      0.92        92
weighted avg       0.95      0.93      0.94        92
```

```
Logistic Regression Confusion Matrix:
 [[60  6]
 [ 0 26]]
```