Autism Prediction using Machine Learning & Python

Project Goal: Develop machine learning models to predict ASD based on relevant features.

Data:

- Raw data in CSV format (adult_data.csv, toddler_data.csv).
- Preprocessed data split into training and testing sets (potentially adult_*.csv, toddler_*.csv).

Model Training:

- Random Forest model (possibly other models) trained using a script (train-autism-prediction-rf.py).
- Trained models saved as serialized files (trained_model_adult.pkl, trained_model_toddler.pkl).

Model Evaluation:

- Script (testing_model.py) to load trained models and evaluate performance on unseen testing data.
- Performance metrics like accuracy and classification reports generated to assess model effectiveness.

Optional GUI:

- Potential script (form_gui.py) to create a user interface for data input and prediction using the trained models.

```
Project-ASD/
├── csvdata/
│   ├── adult_data.csv
│   ├── adult_testing_data.csv
│   ├── adult_training_data.csv
│   ├── toddler_data.csv
│   ├── toddler_testing_data.csv
│   └── toddler_training_data.csv
├── dataset.py       # Optional: Data manipulation
├── form_gui.py      # Optional: User interface
├── matplot.py       # Optional: Visualization
├── train-autism-prediction-rf.py  # Your model training script
├── train_test_split.py            # Data splitting script
└── testing_model.py               # Evaluation script
```

---

toddlers.py

```
import os
import pandas as pd

# Define the directory path for CSV files
csv_dir = '/home/aaryamourya/Project-ASD/csvdata/'
```

```
# Create the directory if it doesn't exist
os.makedirs(csv_dir, exist_ok=True)

# Define the data for toddlers
toddler_data = {
    'A1_Score': [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'A2_Score': [0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1],
    'A3_Score': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'A4_Score': [0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1],
    'A5_Score': [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'A6_Score': [0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'A7_Score': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'A8_Score': [0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1],
    'A9_Score': [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'A10_Score': [0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'age': [3, 4, 5, 6, 3, 4, 5, 6, 3, 4, 5, 6, 3, 4, 5, 6, 3, 4, 5, 6, 3, 4, 5, 6, 3],
    'gender': ['M', 'F', 'M', 'M', 'F', 'F', 'M', 'M', 'F', 'F', 'M', 'F', 'M', 'M', 'F', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M'],
    'ethnicity': ['White-European', 'Asian', 'Latino', 'Others', 'Black', 'Middle Eastern', 'South Asian', 'Hispanic', 'Native American', 'Pacific Islander', 'White-European', 'Asian', 'Latino', 'Others', 'Black', 'Middle Eastern', 'South Asian', 'Hispanic', 'Native American', 'Pacific Islander', 'White-European', 'Asian', 'Latino', 'Others', 'Black'],
    'jundice': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'austim': [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'contry_of_res': ['UK', 'USA', 'Canada', 'Australia', 'France', 'Germany', 'Spain', 'Italy', 'India', 'China', 'Japan', 'Russia', 'Brazil', 'South Africa', 'Egypt', 'Mexico', 'Netherlands', 'Sweden', 'Norway', 'Finland', 'Denmark', 'Belgium', 'Ireland', 'Switzerland', 'Austria'],
    'used_app_before': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'result': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'age_desc': ['4-11 years'] * 25,
    'relation': ['Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent'],
    'Class/ASD': ['YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES']
}

# Create DataFrame for toddlers
df_toddlers = pd.DataFrame(toddler_data)

# Save to CSV
csv_file_path_toddlers = os.path.join(csv_dir, 'toddler_data.csv')
df_toddlers.to_csv(csv_file_path_toddlers, index=False)
print(f"Toddler CSV file saved to: {csv_file_path_toddlers}")
```

Import Libraries:

- `os`: Used for interacting with the operating system, specifically creating directories.
- `pandas as pd`: Imports the pandas library with the alias `pd` for easier use.

Define Directory Path:

- `csv_dir`: Defines the directory path to store the CSV file.

- `os.makedirs(csv_dir, exist_ok=True)`: Creates the directory specified by `csv_dir` if it doesn't exist. The `exist_ok=True` argument ensures no error is raised if the directory already exists.

Create Toddler Data Dictionary:

- `toddler_data`: This dictionary stores the data for toddlers. Each key represents a column name, and the value is a list containing the corresponding data points.

Create DataFrame:

- `df_toddlers = pd.DataFrame(toddler_data)`: Creates a pandas DataFrame named `df_toddlers` from the `toddler_data` dictionary.

Save DataFrame to CSV:

- `csv_file_path_toddlers`: Creates the complete file path for the CSV file using `os.path.join`.
- `df_toddlers.to_csv(csv_file_path_toddlers, index=False)`: Saves the `df_toddlers` DataFrame to a CSV file at the specified path. The `index=False` argument prevents the row index from being saved as a separate column.

Print Confirmation Message:

- Prints a message indicating the successful saving of the CSV file and its location.

A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jundice	austim	contry_of_res	used_app_before	result	age_desc	relation	Class/ASD
1	0	1	0	1	0	1	0	1	0	25	M	White-European	1	0	UK	1	1	18 and more	Self	YES
0	1	1	1	0	1	1	1	0	1	30	F	Asian	0	1	USA	0	0	18 and more	Parent	NO
1	1	0	0	1	0	0	0	1	0	28	M	Latino	1	0	Canada	1	1	18 and more	Relative	YES
0	0	1	1	0	1	1	1	0	1	35	M	Others	0	1	Australia	0	0	18 and more	Others	NO
1	0	0	1	1	0	0	1	1	0	32	F	Black	1	0	France	1	1	18 and more	Parent	YES
1	0	1	0	1	1	1	0	1	1	40	F	Middle Eastern	0	1	Germany	0	0	18 and more	Self	NO
0	1	1	1	0	1	1	1	0	1	22	M	South Asian	1	0	Spain	1	1	18 and more	Parent	YES
1	1	0	0	1	0	0	0	1	0	27	M	Hispanic	0	1	Italy	0	0	18 and more	Relative	NO
0	0	1	1	0	1	1	1	0	1	38	F	Native American	1	0	India	1	1	18 and more	Others	YES
1	0	0	1	1	0	0	1	1	0	45	F	Pacific Islander	0	1	China	0	0	18 and more	Parent	NO
1	0	1	0	1	1	1	0	1	1	29	M	White-European	1	0	Japan	1	1	18 and more	Self	YES
0	1	1	1	0	1	1	1	0	1	33	F	Asian	0	1	Russia	0	0	18 and more	Parent	NO
1	1	0	0	1	0	0	0	1	0	31	M	Latino	1	0	Brazil	1	1	18 and more	Relative	YES
0	0	1	1	0	1	1	1	0	1	26	M	Others	0	1	South Africa	0	0	18 and more	Others	NO
1	0	0	1	1	0	0	1	1	0	39	F	Black	1	0	Egypt	1	1	18 and more	Parent	YES
1	0	1	0	1	1	1	0	1	1	41	F	Middle Eastern	0	1	Mexico	0	0	18 and more	Self	NO
0	1	1	1	0	1	1	1	0	1	36	M	South Asian	1	0	Netherlands	1	1	18 and more	Parent	YES
1	1	0	0	1	0	0	0	1	0	34	F	Hispanic	0	1	Sweden	0	0	18 and more	Relative	NO
0	0	1	1	0	1	1	1	0	1	24	M	Native American	1	0	Norway	1	1	18 and more	Others	YES
1	0	0	1	1	0	0	1	1	0	37	F	Pacific Islander	0	1	Finland	0	0	18 and more	Parent	NO
0	1	1	0	0	1	1	0	0	1	30	M	White-European	1	0	Denmark	1	1	18 and more	Self	YES
1	0	1	1	1	1	1	1	1	1	25	F	Asian	0	1	Belgium	0	0	18 and more	Parent	NO
0	1	0	0	0	0	0	0	0	0	32	M	Latino	1	0	Ireland	1	1	18 and more	Relative	YES
1	0	1	1	1	1	1	1	1	1	40	F	Others	0	1	Switzerland	0	0	18 and more	Others	NO
0	1	0	1	0	0	0	1	0	0	29	M	Black	1	0	Austria	1	1	18 and more	Parent	YES
1	1	1	1	1	1	1	1	1	1	20	F	Asian	1	1	USA	0	1	18 and more	Parent	NO
1	1	1	1	1	1	1	1	1	1				0	0		0	1	4-11 years		YES
0	1	0	1	0	1	0	1	0	1	44	M	White-European	0	0	USA	1	1	4-11 years	Parent	YES
0	1	0	0	0	0	0	1	0	0	20	F	White-European	0	0	Canada	0	0	4-11 years	Self	NO
1	1	1	1	1	1	1	1	1	1	33	F	Others	1	1	Canada	0	1	4-11 years	Parent	YES
0	1	0	1	0	1	1	1	1	1	55	M	Others	0	1	USA	0	1	4-11 years	Parent	YES
0	1	0	1	0	1	0	1	1	1	20	F	Asian	1	1	USA	0	1	4-11 years	Others	YES

Table-01 toddler_data.csv

---

adults.py

```
import pandas as pd

# Define the data for adults
adult_data = {
    'A1_Score': [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'A2_Score': [0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1],
    'A3_Score': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'A4_Score': [0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1],
    'A5_Score': [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'A6_Score': [0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'A7_Score': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'A8_Score': [0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1],
```

```
    'A9_Score': [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    'A10_Score': [0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0],
    'age': [25, 30, 28, 35, 32, 40, 22, 27, 38, 45, 29, 33, 31, 26, 39, 41, 36, 34, 24, 37, 30, 25, 32, 40, 29],
    'gender': ['M', 'F', 'M', 'M', 'F', 'F', 'M', 'M', 'F', 'F', 'M', 'F', 'M', 'M', 'F', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M'],
    'ethnicity': ['White-European', 'Asian', 'Latino', 'Others', 'Black', 'Middle Eastern', 'South Asian', 'Hispanic', 'Native American', 'Pacific Islander', 'White-European', 'Asian', 'Latino', 'Others', 'Black', 'Middle Eastern', 'South Asian', 'Hispanic', 'Native American', 'Pacific Islander', 'White-European', 'Asian', 'Latino', 'Others', 'Black'],
    'jundice': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'austim': [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
     'contry_of_res': ['UK', 'USA', 'Canada', 'Australia', 'France', 'Germany', 'Spain', 'Italy', 'India', 'China', 'Japan', 'Russia', 'Brazil', 'South Africa', 'Egypt', 'Mexico', 'Netherlands', 'Sweden', 'Norway', 'Finland', 'Denmark', 'Belgium', 'Ireland', 'Switzerland', 'Austria'],
    'used_app_before': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'result': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'age_desc': ['18 and more'] * 25,
     'relation': ['Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent', 'Self', 'Parent', 'Relative', 'Others', 'Parent'],
     'Class/ASD': ['YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES', 'NO', 'YES']
}

# Create DataFrame for adults
df_adults = pd.DataFrame(adult_data)

# Save to CSV
csv_file_path_adults = '/home/aaryamourya/Project-ASD/csvdata/adult_data.csv'
df_adults.to_csv(csv_file_path_adults, index=False)
print(f"Adult CSV file saved to: {csv_file_path_adults}")
```

A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jundice	austim	contry_of_res	used_app_before	result	age_desc	relation	Class/ASD
1	0	1	0	1	0	1	0	1	0	25	M	White-European	1	0	UK	1	1	18 and more	Self	YES
0	1	1	1	0	1	1	1	0	1	30	F	Asian	0	1	USA	0	0	18 and more	Parent	NO
1	1	0	0	1	0	0	0	1	0	28	M	Latino	1	0	Canada	1	1	18 and more	Relative	YES
0	0	1	1	0	1	1	1	0	1	35	M	Others	0	1	Australia	0	0	18 and more	Others	NO
1	0	0	1	1	0	0	1	1	0	32	F	Black	1	0	France	1	1	18 and more	Parent	YES
1	0	1	0	1	1	1	0	1	1	40	F	Middle Eastern	0	1	Germany	0	0	18 and more	Self	NO
0	1	1	1	0	1	1	1	0	1	22	M	South Asian	1	0	Spain	1	1	18 and more	Parent	YES
1	1	0	0	1	0	0	0	1	0	27	M	Hispanic	0	1	Italy	0	0	18 and more	Relative	NO
0	0	1	1	0	1	1	1	0	1	38	F	Native American	1	0	India	1	1	18 and more	Others	YES
1	0	0	1	1	0	0	1	1	0	45	F	Pacific Islander	0	1	China	0	0	18 and more	Parent	NO
1	0	1	0	1	1	1	0	1	1	29	M	White-European	1	0	Japan	1	1	18 and more	Self	YES
0	1	1	1	0	1	1	1	0	1	33	F	Asian	0	1	Russia	0	0	18 and more	Parent	NO
1	1	0	0	1	0	0	0	1	0	31	M	Latino	1	0	Brazil	1	1	18 and more	Relative	YES
0	0	1	1	0	1	1	1	0	1	26	M	Others	0	1	South Africa	0	0	18 and more	Others	NO
1	0	0	1	1	0	0	1	1	0	39	F	Black	1	0	Egypt	1	1	18 and more	Parent	YES
1	0	1	0	1	1	1	0	1	1	41	F	Middle Eastern	0	1	Mexico	0	0	18 and more	Self	NO
0	1	1	1	0	1	1	1	0	1	36	M	South Asian	1	0	Netherlands	1	1	18 and more	Parent	YES
1	1	0	0	1	0	0	0	1	0	34	F	Hispanic	0	1	Sweden	0	0	18 and more	Relative	NO
0	0	1	1	0	1	1	1	0	1	24	M	Native American	1	0	Norway	1	1	18 and more	Others	YES
1	0	0	1	1	0	0	1	1	0	37	F	Pacific Islander	0	1	Finland	0	0	18 and more	Parent	NO
0	1	1	0	0	1	1	0	0	1	30	M	White-European	1	0	Denmark	1	1	18 and more	Self	YES
1	0	1	1	1	1	1	1	1	1	25	F	Asian	0	1	Belgium	0	0	18 and more	Parent	NO
0	1	0	0	0	0	0	0	0	0	32	M	Latino	1	0	Ireland	1	1	18 and more	Relative	YES
1	0	1	1	1	1	1	1	1	1	40	F	Others	0	1	Switzerland	0	0	18 and more	Others	NO
0	1	0	1	0	0	0	1	0	0	29	M	Black	1	0	Austria	1	1	18 and more	Parent	YES
1	1	1	1	1	1	1	1	1	1	20	F	Asian	1	1	USA	0	1	18 and more	Parent	NO
1	1	1	1	1	1	1	1	1	1				0	0		0	1	4-11 years		YES
0	1	0	1	0	1	0	1	0	1	44	M	White-European	0	0	USA	1	1	4-11 years	Parent	YES
0	1	0	0	0	0	0	1	0	0	20	F	White-European	0	0	Canada	0	0	4-11 years	Self	NO
1	1	1	1	1	1	1	1	1	1	33	F	Others	1	1	Canada	0	1	4-11 years	Parent	YES
0	1	0	1	0	1	1	1	1	1	55	M	Others	0	1	USA	0	1	4-11 years	Parent	YES
0	1	0	1	0	1	0	1	1	1	20	F	Asian	1	1	USA	0	1	4-11 years	Others	YES

Table 02- adult_data.csv

---

1. **Data Definition:**
    - `adult_data`: This dictionary defines the data for adults with the same structure as the `toddler_data` dictionary.

2. **DataFrame Creation:**
    - `df_adults = pd.DataFrame(adult_data)`: Creates a pandas DataFrame named `df_adults` from the `adult_data` dictionary.
3. **CSV Saving:**
    - `csv_file_path_adults`: Defines the specific file path for the adult data CSV file.
    - `df_adults.to_csv(csv_file_path_adults, index=False)`: Saves the `df_adults` DataFrame to a CSV file at the specified path.
4. **Confirmation Message:**
    - Prints a message indicating the successful saving of the adult CSV file and its location.

---

train_test_split.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load adult dataset
csv_file_path_adult = '/home/aaryamourya/Project-ASD/csvdata/adult_data.csv'
adult_df = pd.read_csv(csv_file_path_adult)

# Load toddler dataset
csv_file_path_toddler = '/home/aaryamourya/Project-ASD/csvdata/toddler_data.csv'
toddler_df = pd.read_csv(csv_file_path_toddler)

# Encode categorical variables
encoder = LabelEncoder()

# Adult dataset encoding
for column in ['gender', 'ethnicity', 'contry_of_res', 'relation', 'Class/ASD']:
    adult_df[column] = encoder.fit_transform(adult_df[column])

# Toddler dataset encoding
for column in ['gender', 'ethnicity', 'contry_of_res', 'relation', 'Class/ASD']:
    toddler_df[column] = encoder.fit_transform(toddler_df[column])

# Splitting the Adult Dataset
X_adult = adult_df.drop(columns=['Class/ASD'])
y_adult = adult_df['Class/ASD']
X_train_adult, X_test_adult, y_train_adult, y_test_adult = train_test_split(X_adult, y_adult, test_size=0.2, random_state=42)

# Splitting the Toddler Dataset
X_toddler = toddler_df.drop(columns=['Class/ASD'])
y_toddler = toddler_df['Class/ASD']
X_train_toddler, X_test_toddler, y_train_toddler, y_test_toddler = train_test_split(X_toddler, y_toddler, test_size=0.2, random_state=42)
```

```
# Print dataset shapes after splitting
print(f"Shapes after splitting:")
print(f"Adult Train/Test: {X_train_adult.shape} {X_test_adult.shape}")
print(f"Toddler Train/Test: {X_train_toddler.shape} {X_test_toddler.shape}")
```

```
aaryamourya@aaryamourya-1563:~/Project-ASD$ python3 train_test_split.py
Shapes after splitting:
Adult Train/Test: (25, 20) (7, 20)
Toddler Train/Test: (20, 20) (6, 20)
aaryamourya@aaryamourya-1563:~/Project-ASD$
```

Import Libraries:

- `pandas as pd`: Used for data manipulation (reading CSV, creating DataFrames).
- `from sklearn.model_selection import train_test_split`: Used to split data into training and testing sets.
- `from sklearn.preprocessing import LabelEncoder`: Used to encode categorical variables.

Load Datasets:

- `adult_df`: Reads the adult data from the CSV file using `pd.read_csv`.
- `toddler_df`: Reads the toddler data from the CSV file using `pd.read_csv`.

Encode Categorical Variables:

- `encoder = LabelEncoder()`: Creates a LabelEncoder object for encoding categorical variables.
- Loops through lists of columns for adult and toddler data:
    - `adult_df[column] = encoder.fit_transform(adult_df[column])`: Encodes each column in the loop using the LabelEncoder. This replaces string values in the specified column with integer codes.

Splitting Data (Adult & Toddler):

- `X_adult`: Creates a new DataFrame containing all features (excluding the target variable) for the adult data using `drop`.
- `y_adult`: Selects the target variable ('Class/ASD') for the adult data.
- `X_train_adult, X_test_adult, y_train_adult, y_test_adult = train_test_split(X_adult, y_adult, test_size=0.2, random_state=42)`: Splits the adult data's features (X_adult) and target variable (y_adult) into training and testing sets using `train_test_split`. The `test_size` argument specifies that 20% of the data will be used for testing, and `random_state=42` ensures reproducibility by setting a seed for the random split. Similar steps are performed for the toddler data (`X_toddler`, `y_toddler`, etc.).

Print Data Shapes:

- Prints the shapes (number of rows and columns) of the training and testing sets for both adult and toddler data.

---

train-autism-prediction-rf.py

```
import pandas as pd
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, precision_score, recall_score, f1_score, confusion_matrix

# Specify full file paths
csv_file_adult = '/home/aaryamourya/Project-ASD/csvdata/adult_data.csv'
csv_file_toddler = '/home/aaryamourya/Project-ASD/csvdata/toddler_data.csv'

# Load data
df_adult = pd.read_csv(csv_file_adult)
df_toddler = pd.read_csv(csv_file_toddler)

# Handle missing values (fill with mean for numerical features)
numerical_features_adult = df_adult.select_dtypes(include=['float64', 'int64']).columns
numerical_features_toddler = df_toddler.select_dtypes(include=['float64', 'int64']).columns

df_adult[numerical_features_adult] =
df_adult[numerical_features_adult].fillna(df_adult[numerical_features_adult].mean())
df_toddler[numerical_features_toddler] =
df_toddler[numerical_features_toddler].fillna(df_toddler[numerical_features_toddler].mean())

# Encode categorical features
le = LabelEncoder()
categorical_features_adult = df_adult.select_dtypes(include=['object']).columns
categorical_features_toddler = df_toddler.select_dtypes(include=['object']).columns

for col in categorical_features_adult:
    df_adult[col] = le.fit_transform(df_adult[col])

for col in categorical_features_toddler:
    df_toddler[col] = le.fit_transform(df_toddler[col])

# Feature scaling
scaler = StandardScaler()
df_adult[numerical_features_adult] = scaler.fit_transform(df_adult[numerical_features_adult])
df_toddler[numerical_features_toddler] = scaler.fit_transform(df_toddler[numerical_features_toddler])

# Separate features and target variables
X_train_adult, X_test_adult, y_train_adult, y_test_adult = train_test_split(
    df_adult.drop("Class/ASD", axis=1), df_adult["Class/ASD"], test_size=0.2, random_state=42
)
```

```
X_train_toddler, X_test_toddler, y_train_toddler, y_test_toddler = train_test_split(
    df_toddler.drop("Class/ASD", axis=1), df_toddler["Class/ASD"], test_size=0.2, random_state=42
)

# Train models
clf_adult = RandomForestClassifier(n_estimators=100, random_state=42)
clf_toddler = RandomForestClassifier(n_estimators=100, random_state=42)

clf_adult.fit(X_train_adult, y_train_adult)
clf_toddler.fit(X_train_toddler, y_train_toddler)

# Evaluation function
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    cm = confusion_matrix(y_test, y_pred)

    print("Model Performance:")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")
    print("\nConfusion Matrix:\n", cm)

# Evaluate models
print("Adult Model Performance:")
evaluate_model(clf_adult, X_test_adult, y_test_adult)

print("\nToddler Model Performance:")
evaluate_model(clf_toddler, X_test_toddler, y_test_toddler)
```

```
aaryamourya@aaryamourya-1563:~/Project-ASD$ python3 train-autism-prediction-rf.py
Adult Model Performance:
Model Performance:
Accuracy: 0.86
Precision: 0.89
Recall: 0.86
F1 Score: 0.86

Confusion Matrix:
 [[3 0]
 [1 3]]

Toddler Model Performance:
Model Performance:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00

Confusion Matrix:
 [[2 0]
 [0 4]]
aaryamourya@aaryamourya-1563:~/Project-ASD$
```

---

1. **Full File Paths:**
   - `csv_file_adult` and `csv_file_toddler`: These variables now explicitly define the full file paths for the CSV files, making the code more portable.
2. **Missing Value Handling:**
   - New sections identify numerical features for adults (`numerical_features_adult`) and toddlers (`numerical_features_toddler`).
   - `df_adult[numerical_features_adult]` and `df_toddler[numerical_features_toddler]`: These lines select the numerical features from each DataFrame.
   - `fillna(df_adult[numerical_features_adult].mean())` and `.fillna(df_toddler[numerical_features_toddler].mean())`: These lines replace missing values in the numerical features with the mean value of each feature in the respective DataFrame.
3. **Feature Scaling (New):**
   - `StandardScaler()` is imported for feature scaling.
   - `scaler = StandardScaler()`: Creates a StandardScaler object.
   - `df_adult[numerical_features_adult] = scaler.fit_transform(df_adult[numerical_features_adult])` and similar line for toddlers: These lines standardize the numerical features (scaling them to have a mean of 0 and standard deviation of 1) using the scaler object. This can improve the performance of machine learning algorithms.
4. **Evaluation Function (New):**
   - `evaluate_model` function: This function takes a model, testing features (`X_test`), and testing target variable (`y_test`) as input.

- `y_pred = model.predict(X_test)`: Predicts the target variable for the testing data using the model.
- The function calculates and prints the following evaluation metrics:
  - Accuracy: Proportion of correctly predicted labels.
  - Precision: Ratio of true positives to all predicted positives (how good the model is at identifying actual positives).
  - Recall: Ratio of true positives to all actual positives (how good the model is at finding all positives).
  - F1 Score: Harmonic mean of precision and recall.
  - Confusion Matrix: A table showing the distribution of actual vs. predicted labels.

5. **Model Training and Evaluation:**
   - `RandomForestClassifier` is used for both adult and toddler models with `n_estimators=100` and `random_state=42` arguments.
   - The models are trained using `fit` and evaluated using the `evaluate_model` function for both adults and toddlers.

---

autism_prediction_random_forest.py

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# Define file paths
csv_file_path_adult = '/home/aaryamourya/Project-ASD/csvdata/adult_data.csv'
csv_file_path_toddler = '/home/aaryamourya/Project-ASD/csvdata/toddler_data.csv'

# Load datasets
adult_df = pd.read_csv(csv_file_path_adult)
toddler_df = pd.read_csv(csv_file_path_toddler)

# Display the first few rows of each dataset to understand their structure
print("Adult Dataset:")
print(adult_df.head())

print("\nToddler Dataset:")
print(toddler_df.head())

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Encode categorical variables for adults
for column in ['gender', 'ethnicity', 'contry_of_res', 'relation', 'Class/ASD']:
    adult_df[column] = label_encoder.fit_transform(adult_df[column])

# Encode categorical variables for toddlers
for column in ['gender', 'ethnicity', 'contry_of_res', 'relation', 'Class/ASD']:
    toddler_df[column] = label_encoder.fit_transform(toddler_df[column])
```

```
# Handle missing values for adults
# Exclude non-numeric columns like 'age_desc' from mean calculation
numeric_columns_adult = adult_df.select_dtypes(include=['number']).columns
adult_df[numeric_columns_adult] = adult_df[numeric_columns_adult].fillna(adult_df[numeric_columns_adult].mean())

# Handle missing values for toddlers
numeric_columns_toddler = toddler_df.select_dtypes(include=['number']).columns
toddler_df[numeric_columns_toddler] = toddler_df[numeric_columns_toddler].fillna(toddler_df[numeric_columns_toddler].mean())

# Display the datasets after preprocessing
print("\nEncoded and Preprocessed Adult Dataset:")
print(adult_df.head())

print("\nEncoded and Preprocessed Toddler Dataset:")
print(toddler_df.head())

# Example of splitting data into training and testing sets
# For demonstration purposes, you may adjust test_size and random_state as needed
X_adult = adult_df.drop('Class/ASD', axis=1)
y_adult = adult_df['Class/ASD']
X_train_adult, X_test_adult, y_train_adult, y_test_adult = train_test_split(X_adult, y_adult, test_size=0.2, random_state=42)

X_toddler = toddler_df.drop('Class/ASD', axis=1)
y_toddler = toddler_df['Class/ASD']
X_train_toddler, X_test_toddler, y_train_toddler, y_test_toddler = train_test_split(X_toddler, y_toddler, test_size=0.2, random_state=42)

# Example:
# adult_df.to_csv('/home/aaryamourya/Project-ASD/preprocessed_adult_data.csv', index=False)
# toddler_df.to_csv('/home/aaryamourya/Project-ASD/preprocessed_toddler_data.csv', index=False)
```

```
aaryamourya@aaryamourya-1563:~/Project-ASD$ python3 autism_prediction_random_forest.py
Adult Dataset:
   A1_Score  A2_Score  A3_Score  A4_Score  ...  result     age_desc  relation  Class/ASD
0         1         0         1         0  ...       1  18 and more      Self        YES
1         0         1         1         1  ...       0  18 and more    Parent         NO
2         1         1         0         0  ...       1  18 and more  Relative        YES
3         0         0         1         1  ...       0  18 and more    Others         NO
4         1         0         0         1  ...       1  18 and more    Parent        YES

[5 rows x 21 columns]

Toddler Dataset:
   A1_Score  A2_Score  A3_Score  A4_Score  ...  result    age_desc  relation  Class/ASD
0         1         0         1         0  ...       1  4-11 years      Self        YES
1         0         1         1         1  ...       0  4-11 years    Parent         NO
2         1         1         0         0  ...       1  4-11 years  Relative        YES
3         0         0         1         1  ...       0  4-11 years    Others         NO
4         1         0         0         1  ...       1  4-11 years    Parent        YES

[5 rows x 21 columns]

Encoded and Preprocessed Adult Dataset:
   A1_Score  A2_Score  A3_Score  ...     age_desc  relation  Class/ASD
0         1         0         1  ...  18 and more         3          1
1         0         1         1  ...  18 and more         1          0
2         1         1         0  ...  18 and more         2          1
3         0         0         1  ...  18 and more         0          0
4         1         0         0  ...  18 and more         1          1

[5 rows x 21 columns]

Encoded and Preprocessed Toddler Dataset:
   A1_Score  A2_Score  A3_Score  A4_Score  ...  result    age_desc  relation  Class/ASD
0         1         0         1         0  ...       1  4-11 years         3          1
1         0         1         1         1  ...       0  4-11 years         1          0
2         1         1         0         0  ...       1  4-11 years         2          1
3         0         0         1         1  ...       0  4-11 years         0          0
4         1         0         0         1  ...       1  4-11 years         1          1

[5 rows x 21 columns]
aaryamourya@aaryamourya-1563:~/Project-ASD$
```

Load and Display Datasets:

- Loads data from CSV files using `pd.read_csv`.
- Prints the first few rows (`head()`) of each DataFrame to understand their structure.

Encode Categorical Variables:

- Creates a `LabelEncoder` object.
- Iterates through lists of columns for adults and toddlers, encoding categorical variables using `fit_transform`.

Handle Missing Values:

- Identifies numeric columns for adults (`numeric_columns_adult`) and toddlers (`numeric_columns_toddler`) using `select_dtypes`.
- Fills missing values in numeric columns with the mean value of each column using `.fillna(df[numeric_columns].mean())`.

Display Preprocessed Data:

- Prints the first few rows of the adult and toddler DataFrames after encoding and handling missing values.

Splitting Data (Example):

- Separates features (`X_adult`, `X_toddler`) from target variables (`y_adult`, `y_toddler`) for both datasets.
- Splits adult and toddler data into training and testing sets using `train_test_split`. The `test_size` argument specifies the proportion of data for testing (here, 20%). `random_state=42` ensures reproducibility.

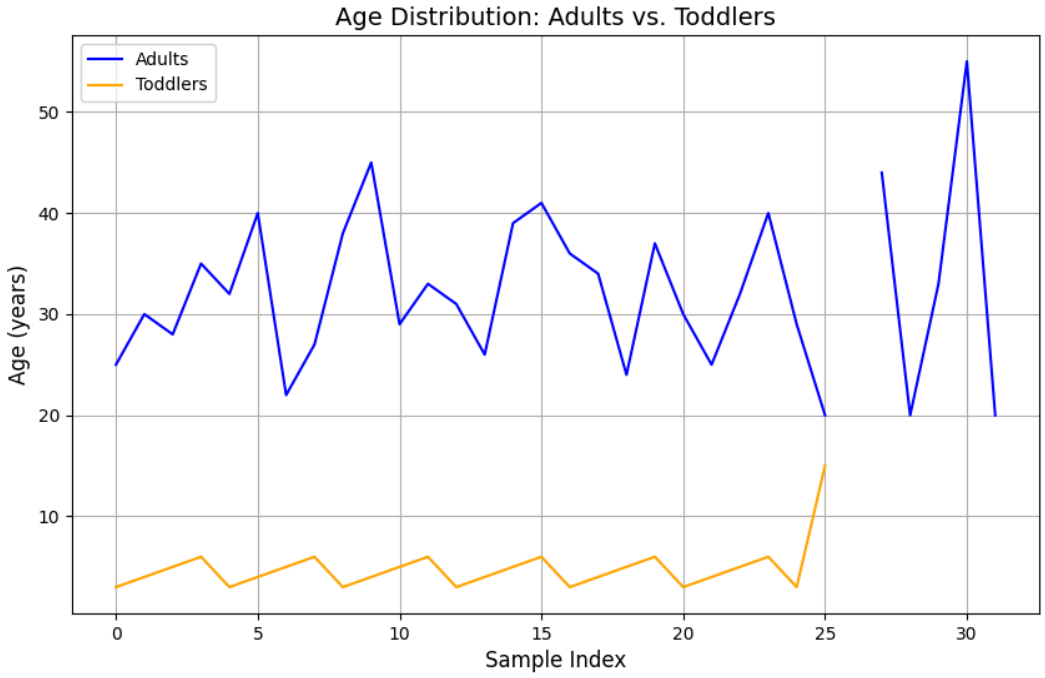
---

matplot.py

```
import pandas as pd
import matplotlib.pyplot as plt

# Load adult and toddler data
df_adult = pd.read_csv("/home/aaryamourya/Project-ASD/csvdata/adult_data.csv")
df_toddler = pd.read_csv("/home/aaryamourya/Project-ASD/csvdata/toddler_data.csv")

# Plot curves for age distribution of adults and toddlers
plt.figure(figsize=(10, 6))
plt.plot(df_adult['age'], label='Adults', color='blue')
plt.plot(df_toddler['age'], label='Toddlers', color='orange')
plt.xlabel('Sample Index', fontsize=12)
plt.ylabel('Age (years)', fontsize=12)
plt.title('Age Distribution: Adults vs. Toddlers', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()

# Plot horizontal bar chart of gender distribution
gender_counts_adult = df_adult['gender'].value_counts()
gender_counts_toddler = df_toddler['gender'].value_counts()

fig, ax = plt.subplots(figsize=(8, 6))

bar_width = 0.35
index = range(len(gender_counts_adult))

bar1 = ax.barh(index, gender_counts_adult.values, bar_width, label='Adults', color='blue')
```

```
bar2 = ax.barh([i + bar_width for i in index], gender_counts_toddler.values, bar_width, label='Toddlers', color='orange')

ax.set_xlabel('Count', fontsize=12)
ax.set_ylabel('Gender', fontsize=12)
ax.set_title('Gender Distribution: Adults vs. Toddlers', fontsize=14)
ax.set_yticks([i + bar_width / 2 for i in index])
ax.set_yticklabels(gender_counts_adult.index)
ax.legend()

plt.grid(True)
plt.show()

# Plot scatter plot of age vs. result for adults and toddlers
plt.figure(figsize=(10, 6))
plt.scatter(df_adult['age'], df_adult['result'], marker='o', label='Adults', color='blue', alpha=0.7)
plt.scatter(df_toddler['age'], df_toddler['result'], marker='s', label='Toddlers', color='orange', alpha=0.7)
plt.xlabel('Age (years)', fontsize=12)
plt.ylabel('Result', fontsize=12)
plt.title('Age vs. Result: Adults vs. Toddlers', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

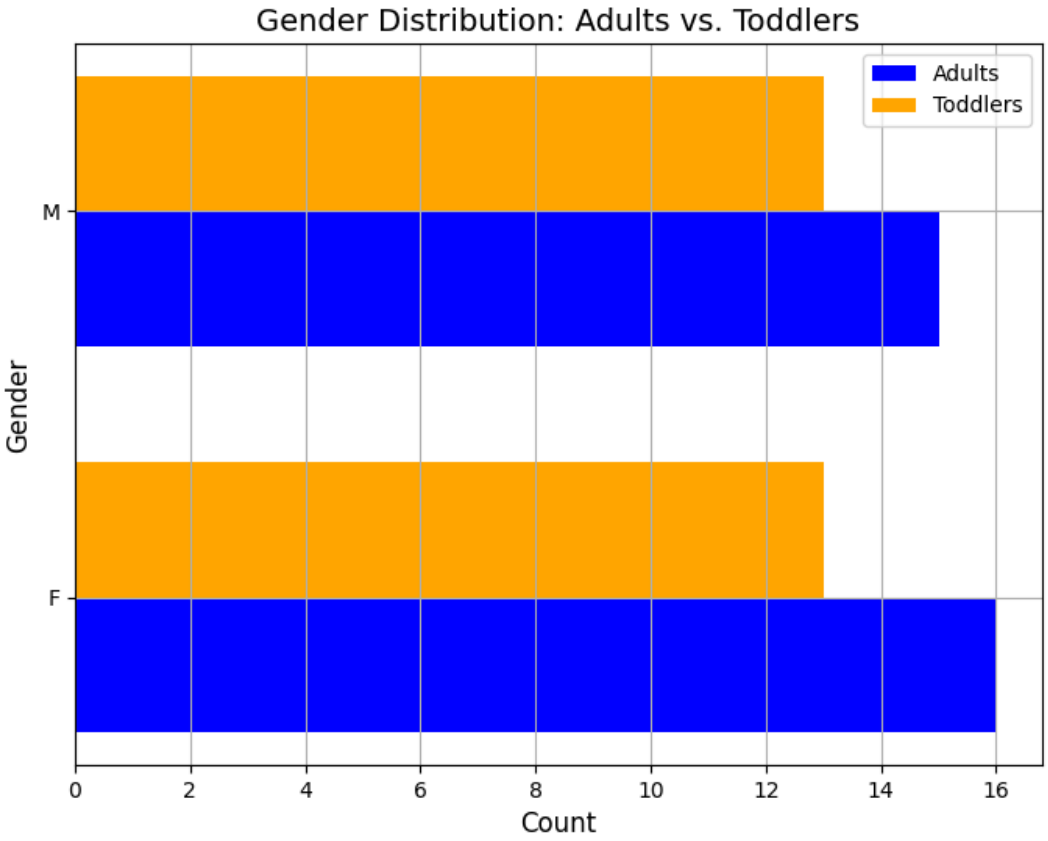
Age Distribution:



- `plt.plot` is used to create line plots for the age distribution of adults (blue) and toddlers (orange).

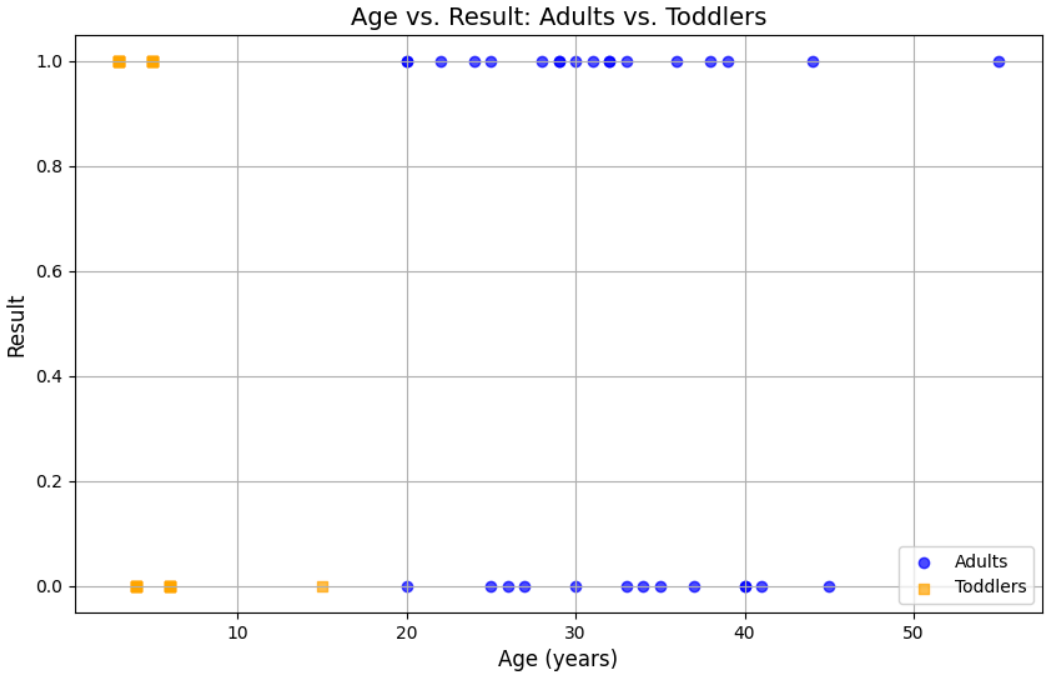
- Labels, titles, and grid lines are added for clarity.

Gender Distribution:



- `value_counts` is used to count the occurrences of each gender in both datasets.
- A horizontal bar chart is created using `barh` with separate bars for adults and toddlers.
- The x-axis represents the count, and the y-axis represents gender labels.
- Labels, titles, and grid lines are added for better interpretation.

Age vs. Result:



- `plt.scatter` is used to create scatter plots for adults (blue circles) and toddlers (orange squares).
- The x-axis represents age, and the y-axis represents the result variable.
- Labels, titles, and grid lines are added for clarity.

---

form_gui.py

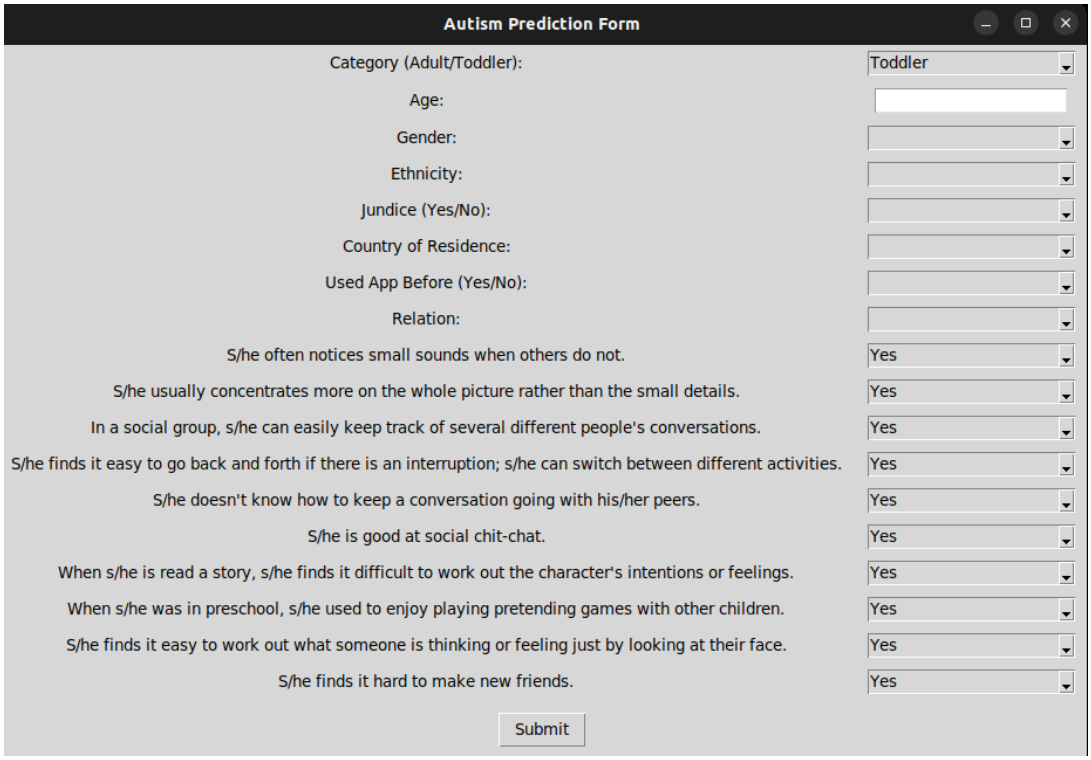
```
import tkinter as tk
from tkinter import ttk
import pandas as pd

# Define options for dropdowns
ethnicity_options = ["White-European", "Asian", "Latino", "Others", "Black", "Middle Eastern", "South Asian"]
country_options = ["UK", "USA", "Canada", "Australia", "France", "Germany", "Spain", "Other"]
relation_options = ["Self", "Parent", "Relative", "Others"]

# Questions for adults and toddlers
adult_questions = [
    "I often notice small sounds when others do not.",
    "I usually concentrate more on the whole picture rather than the small details.",
    "I find it easy to do more than one thing at once.",
    "If there is an interruption, I can switch back to what I was doing very quickly.",
    "I find it easy to read between the lines when someone is talking to me.",
    "I know how to tell if someone listening to me is getting bored.",
    "When I'm reading a story I find it difficult to work out the character's intentions.",
    "I like to collect information about categories of things.",
```

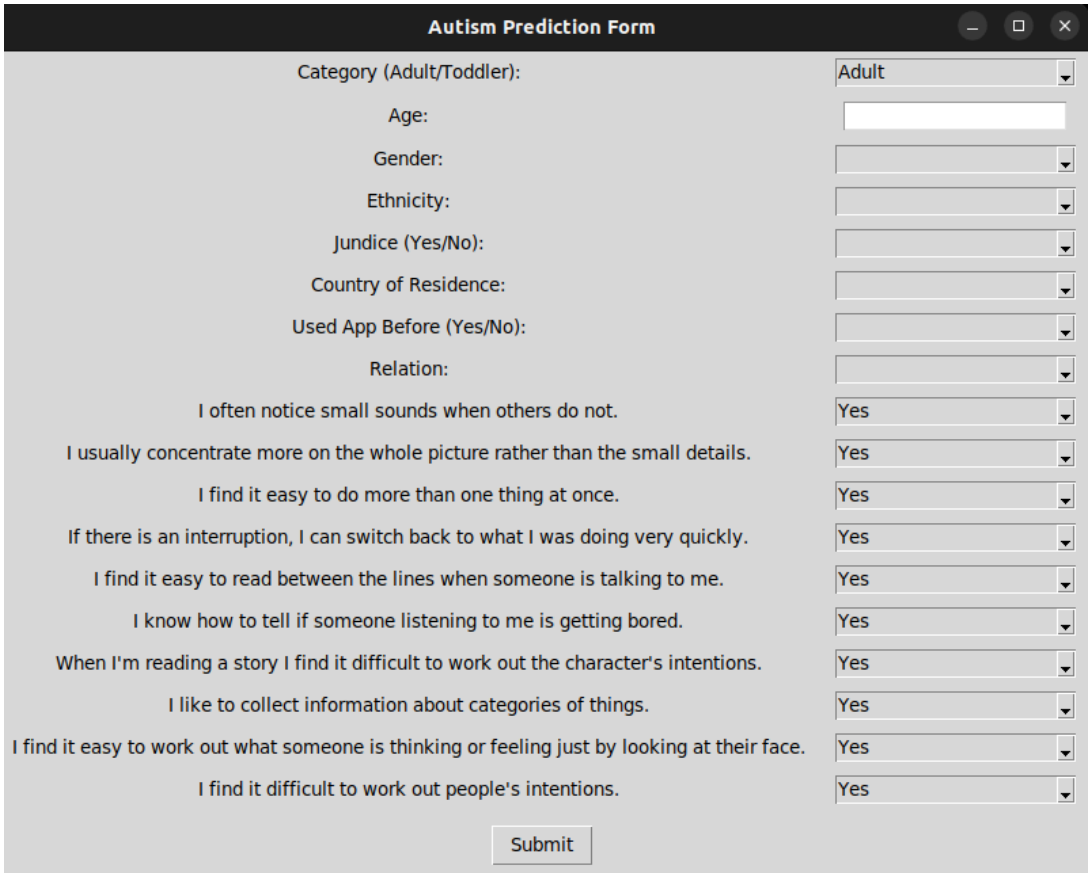
```
    "I find it easy to work out what someone is thinking or feeling just by looking at their face.",
    "I find it difficult to work out people's intentions."
]

toddler_questions = [
    "S/he often notices small sounds when others do not.",
    "S/he usually concentrates more on the whole picture rather than the small details.",
    "In a social group, s/he can easily keep track of several different people's conversations.",
    "S/he finds it easy to go back and forth if there is an interruption; s/he can switch between different activities.",
    "S/he doesn't know how to keep a conversation going with his/her peers.",
    "S/he is good at social chit-chat.",
    "When s/he is read a story, s/he finds it difficult to work out the character's intentions or feelings.",
    "When s/he was in preschool, s/he used to enjoy playing pretending games with other children.",
    "S/he finds it easy to work out what someone is thinking or feeling just by looking at their face.",
    "S/he finds it hard to make new friends."
]

# Function to handle form submission
def submit_form():
    age_value = age_entry.get()
    category_value = category_var.get()
    gender_value = gender_var.get()
    ethnicity_value = ethnicity_var.get()
    jundice_value = 1 if jundice_var.get() == "Yes" else 0
    country_value = country_var.get()
    used_app_value = 1 if used_app_var.get() == "Yes" else 0
    relation_value = relation_var.get()

    # Collecting answers to the questions
    answers = [
        1 if yesno_var1.get() == "Yes" else 0,
        1 if yesno_var2.get() == "Yes" else 0,
        1 if yesno_var3.get() == "Yes" else 0,
        1 if yesno_var4.get() == "Yes" else 0,
        1 if yesno_var5.get() == "Yes" else 0,
        1 if yesno_var6.get() == "Yes" else 0,
        1 if yesno_var7.get() == "Yes" else 0,
        1 if yesno_var8.get() == "Yes" else 0,
        1 if yesno_var9.get() == "Yes" else 0,
        1 if yesno_var10.get() == "Yes" else 0
    ]

    # Determine ASD prediction based on scores
    total_score = sum(answers)
    if total_score >= 5:
        asd_result = 1
        asd_class = "YES"
        autism_value = 1
    else:
```

```
        asd_result = 0
        asd_class = "NO"
        autism_value = 0

    # Determine the CSV path based on category
        csv_path = '/home/aaryamourya/Project-ASD/csvdata/adult_data.csv' if category_value == "Adult" else
'/home/aaryamourya/Project-ASD/csvdata/toddler_data.csv'

    # Create DataFrame and append to CSV
    df = pd.DataFrame([{
        'A1_Score': answers[0], 'A2_Score': answers[1], 'A3_Score': answers[2], 'A4_Score': answers[3],
        'A5_Score': answers[4], 'A6_Score': answers[5], 'A7_Score': answers[6], 'A8_Score': answers[7],
        'A9_Score': answers[8], 'A10_Score': answers[9], 'age': age_value, 'gender': gender_value,
        'ethnicity': ethnicity_value, 'jundice': jundice_value,
        'contry_of_res': country_value, 'used_app_before': used_app_value, 'result': asd_result,
        'age_desc': '4-11 years', 'relation': relation_value, 'Class/ASD': asd_class, 'autism': autism_value
    }])
    df.to_csv(csv_path, mode='a', header=False, index=False)

    # Clear form fields
    age_entry.delete(0, tk.END)
    gender_var.set('')
    ethnicity_var.set('')
    jundice_var.set('')
    country_var.set('')
    used_app_var.set('')
    relation_var.set('')
    for var in [yesno_var1, yesno_var2, yesno_var3, yesno_var4, yesno_var5,
                yesno_var6, yesno_var7, yesno_var8, yesno_var9, yesno_var10]:
        var.set('Yes')

    # Display ASD prediction result on a new page
    new_window = tk.Toplevel(root)
    new_window.title("Result")
      result_message = "Thank you! The person might have ASD." if asd_result == 1 else "Thank you! The person
might not have ASD."
    result_label = tk.Label(new_window, text=result_message, font=('Helvetica', 16))
    result_label.pack(padx=20, pady=20)

    # Close the form after 3 seconds
    root.after(3000, root.quit)

# Creating main application window
root = tk.Tk()
root.title("Autism Prediction Form")

# Labels and entry fields
category_label = tk.Label(root, text="Category (Adult/Toddler):")
category_label.grid(row=0, column=0, padx=10, pady=5)
```

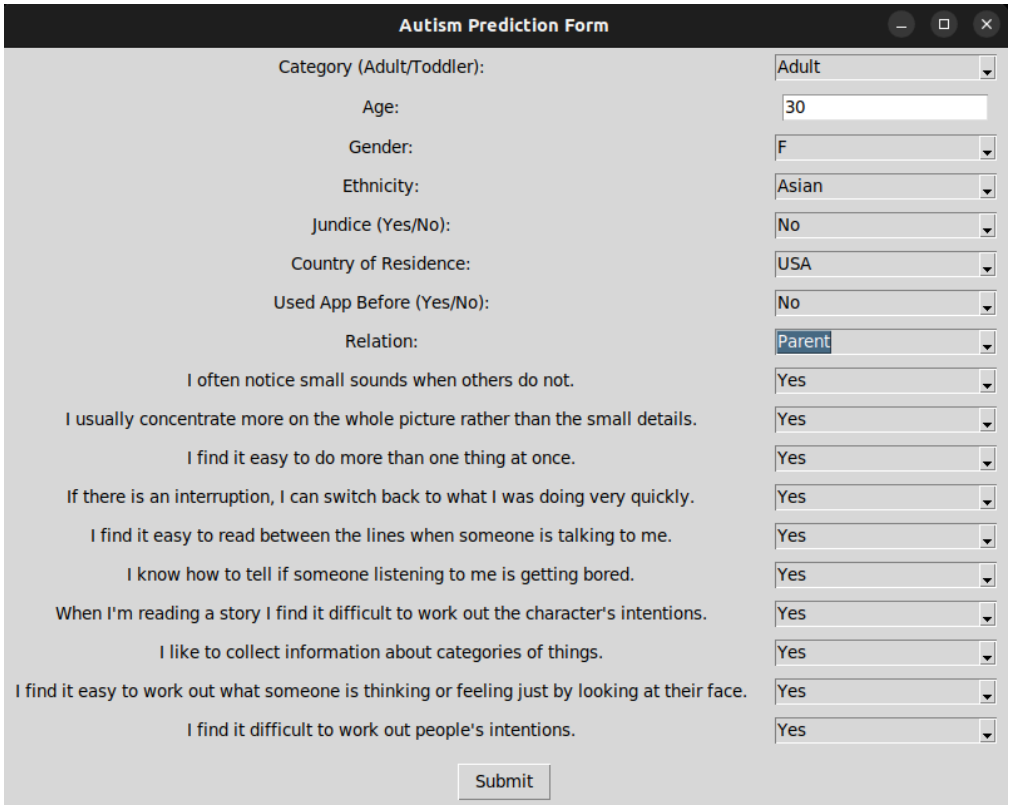
```
category_var = tk.StringVar()
category_dropdown = ttk.Combobox(root, textvariable=category_var, values=["Adult", "Toddler"], state="readonly")
category_dropdown.grid(row=0, column=1, padx=10, pady=5)
category_dropdown.current(0)

age_label = tk.Label(root, text="Age:")
age_label.grid(row=1, column=0, padx=10, pady=5)
age_entry = tk.Entry(root)
age_entry.grid(row=1, column=1, padx=10, pady=5)

gender_label = tk.Label(root, text="Gender:")
gender_label.grid(row=2, column=0, padx=10, pady=5)
gender_var = tk.StringVar()
gender_dropdown = ttk.Combobox(root, textvariable=gender_var, values=["M", "F"], state="readonly")
gender_dropdown.grid(row=2, column=1, padx=10, pady=5)

ethnicity_label = tk.Label(root, text="Ethnicity:")
ethnicity_label.grid(row=3, column=0, padx=10, pady=5)
ethnicity_var = tk.StringVar()
ethnicity_dropdown = ttk.Combobox(root, textvariable=ethnicity_var, values=ethnicity_options, state="readonly")
ethnicity_dropdown.grid(row=3, column=1, padx=10, pady=5)

jundice_label = tk.Label(root, text="Jundice (Yes/No):")
jundice_label.grid(row=4, column=0, padx=10, pady=5)
jundice_var = tk.StringVar()
jundice_dropdown = ttk.Combobox(root, textvariable=jundice_var, values=["Yes", "No"], state="readonly")
jundice_dropdown.grid(row=4, column=1, padx=10, pady=5)

country_label = tk.Label(root, text="Country of Residence:")
country_label.grid(row=5, column=0, padx=10, pady=5)
country_var = tk.StringVar()
country_dropdown = ttk.Combobox(root, textvariable=country_var, values=country_options, state="readonly")
country_dropdown.grid(row=5, column=1, padx=10, pady=5)

used_app_label = tk.Label(root, text="Used App Before (Yes/No):")
used_app_label.grid(row=6, column=0, padx=10, pady=5)
used_app_var = tk.StringVar()
used_app_dropdown = ttk.Combobox(root, textvariable=used_app_var, values=["Yes", "No"], state="readonly")
used_app_dropdown.grid(row=6, column=1, padx=10, pady=5)

relation_label = tk.Label(root, text="Relation:")
relation_label.grid(row=7, column=0, padx=10, pady=5)
relation_var = tk.StringVar()
relation_dropdown = ttk.Combobox(root, textvariable=relation_var, values=relation_options, state="readonly")
relation_dropdown.grid(row=7, column=1, padx=10, pady=5)

# Question labels and dropdowns
yesno_var1 = tk.StringVar(value="Yes")
```

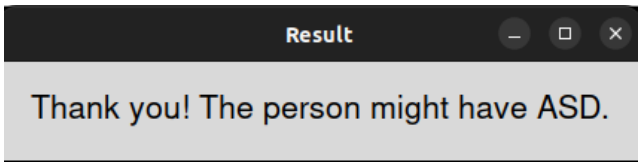
```
yesno_var2 = tk.StringVar(value="Yes")
yesno_var3 = tk.StringVar(value="Yes")
yesno_var4 = tk.StringVar(value="Yes")
yesno_var5 = tk.StringVar(value="Yes")
yesno_var6 = tk.StringVar(value="Yes")
yesno_var7 = tk.StringVar(value="Yes")
yesno_var8 = tk.StringVar(value="Yes")
yesno_var9 = tk.StringVar(value="Yes")
yesno_var10 = tk.StringVar(value="Yes")

questions_vars = [
    yesno_var1, yesno_var2, yesno_var3, yesno_var4, yesno_var5,
    yesno_var6, yesno_var7, yesno_var8, yesno_var9, yesno_var10
]

def update_questions(*args):
    if category_var.get() == "Adult":
        questions = adult_questions
    else:
        questions = toddler_questions

    for idx, question in enumerate(questions):
        question_labels[idx].config(text=question)

question_labels = []
for i in range(10):
    question_label = tk.Label(root, text="")
    question_label.grid(row=8+i, column=0, padx=10, pady=5)
    question_labels.append(question_label)
            question_dropdown = ttk.Combobox(root, textvariable=questions_vars[i], values=["Yes", "No"],
state="readonly")
    question_dropdown.grid(row=8+i, column=1, padx=10, pady=5)

category_var.trace("w", update_questions)
update_questions()

# Submit button
submit_button = tk.Button(root, text="Submit", command=submit_form)
submit_button.grid(row=18, columnspan=2, pady=10)

# Start the application
root.mainloop()
```

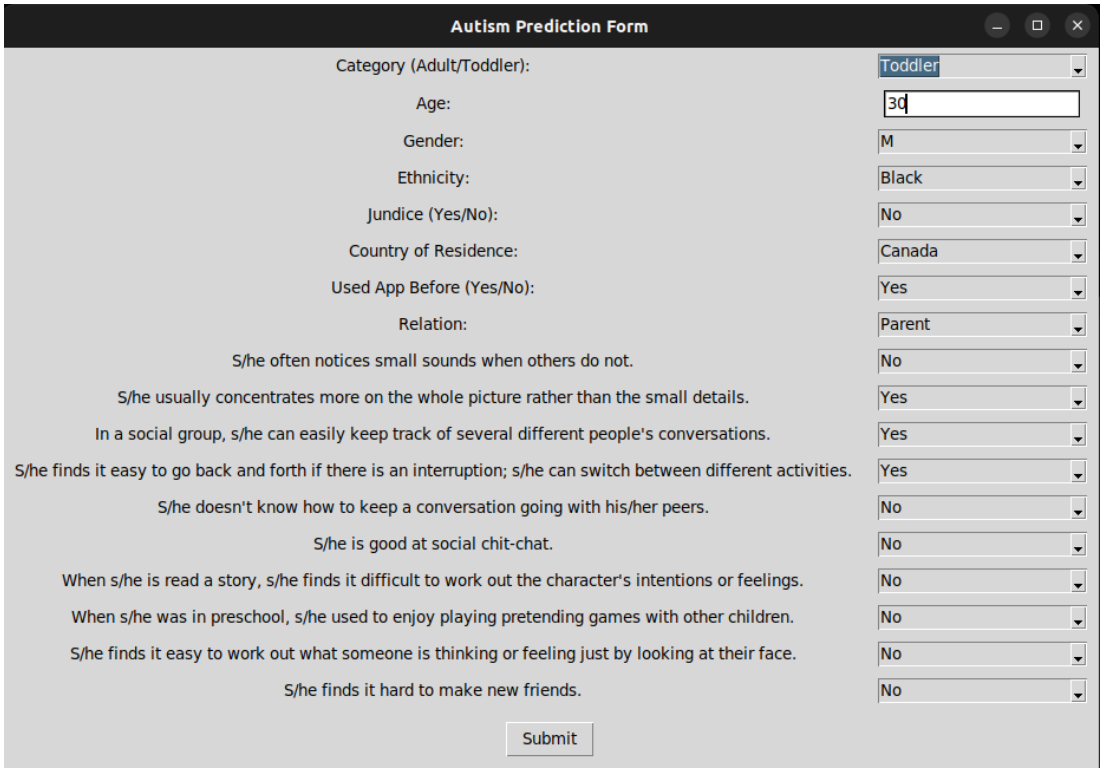

Autism Prediction Form
Category (Adult/Toddler):
Toddler
Age:
Gender:
Ethnicity:
Jundice (Yes/No):
Country of Residence:
Used App Before (Yes/No):
Relation:
S/he often notices small sounds when others do not.
Yes
S/he usually concentrates more on the whole picture rather than the small details.
Yes
In a social group, s/he can easily keep track of several different people's conversations.
Yes
S/he finds it easy to go back and forth if there is an interruption; s/he can switch between different activities.
Yes
S/he doesn't know how to keep a conversation going with his/her peers.
Yes
S/he is good at social chit-chat.
Yes
When s/he is read a story, s/he finds it difficult to work out the character's intentions or feelings.
Yes
When s/he was in preschool, s/he used to enjoy playing pretending games with other children.
Yes
S/he finds it easy to work out what someone is thinking or feeling just by looking at their face.
Yes
S/he finds it hard to make new friends.
Yes
Submit

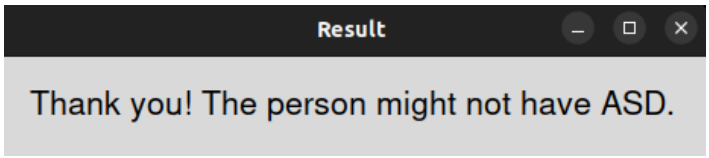

Autism Prediction Form
Category (Adult/Toddler):
Adult
Age:
Gender:
Ethnicity:
Jundice (Yes/No):
Country of Residence:
Used App Before (Yes/No):
Relation:
I often notice small sounds when others do not.
Yes
I usually concentrate more on the whole picture rather than the small details.
Yes
I find it easy to do more than one thing at once.
Yes
If there is an interruption, I can switch back to what I was doing very quickly.
Yes
I find it easy to read between the lines when someone is talking to me.
Yes
I know how to tell if someone listening to me is getting bored.
Yes
When I'm reading a story I find it difficult to work out the character's intentions.
Yes
I like to collect information about categories of things.
Yes
I find it easy to work out what someone is thinking or feeling just by looking at their face.
Yes
I find it difficult to work out people's intentions.
Yes
Submit

Example-


Autism Prediction Form
Category (Adult/Toddler):
Adult
Age:
30
Gender:
F
Ethnicity:
Asian
Jundice (Yes/No):
No
Country of Residence:
USA
Used App Before (Yes/No):
No
Relation:
Parent
I often notice small sounds when others do not.
Yes
I usually concentrate more on the whole picture rather than the small details.
Yes
I find it easy to do more than one thing at once.
Yes
If there is an interruption, I can switch back to what I was doing very quickly.
Yes
I find it easy to read between the lines when someone is talking to me.
Yes
I know how to tell if someone listening to me is getting bored.
Yes
When I'm reading a story I find it difficult to work out the character's intentions.
Yes
I like to collect information about categories of things.
Yes
I find it easy to work out what someone is thinking or feeling just by looking at their face.
Yes
I find it difficult to work out people's intentions.
Yes
Submit


Result
Thank you! The person might have ASD.





User Interface:

- **Dropdown Menus:** The GUI provides dropdown menus for category (Adult/Toddler), gender, ethnicity, country, jundice (Yes/No), used app before (Yes/No), and relation.
- **Entry Field:** An entry field allows users to enter the age.
- **Radio Buttons (replaced with Comboboxes):** The original code used Radio buttons for Yes/No answers, but these have been replaced with Comboboxes for a cleaner and more user-friendly experience.
- **Question Labels:** The application displays questions dynamically based on the selected category (Adult or Toddler).
- **Submit Button:** Clicking the "Submit" button triggers the `submit_form` function to process the data.

Functionality:

- **`submit_form` Function:**
  - Gathers user input from all fields.
  - Calculates a total score based on the Yes/No answers (1 for Yes, 0 for No).
  - Determines the ASD prediction (YES if score >= 5, NO otherwise).
  - Sets corresponding values for ASD class and autism (1 for predicted ASD, 0 otherwise).
  - Selects the CSV path based on the category (adult or toddler data).
  - Creates a DataFrame with user input and prediction results.

    - Appends the DataFrame to the appropriate CSV file in append mode.
    - Clears the form fields after submission.
    - Displays a pop-up window with the ASD prediction result ("might have ASD" or "might not have ASD").
    - Closes the main window after 3 seconds.

Improvements:

- **Dynamic Question Labels:** The questions update based on the selected category using the `update_questions` function triggered by changes in the `category_var`.
- **Clear and Consistent Variable Naming:** Variable names are descriptive and consistent throughout the code.

Important Note:

- While the code provides a user interface for collecting data, it's essential to understand that this is for educational or informational purposes only. Autism Spectrum Disorder (ASD) diagnosis requires professional evaluation by qualified healthcare providers.

Data Preparation:

1. **Raw Data (adult_data.csv, toddler_data.csv): These files likely contain the original datasets you collected, potentially including features relevant to ASD diagnosis, such as age, gender, ethnicity, behavioral observations, etc.**
2. **Preprocessing (potentially using adult_*.csv, toddler*_.csv): These files might be generated scripts to clean, transform, and prepare the raw data for modeling. This could involve handling missing values, encoding categorical variables, feature scaling, and splitting the data into training and testing sets using `train_test_split.py`.**

Model Training:

1. **Training Script (train-autism-prediction-rf.py): As you suspected, this script likely trains a Random Forest model (or potentially other models) on the prepared training data. It defines the model architecture, hyperparameters (tuning knobs), and training process.**
2. **Model Training: The script trains the model using the training data, allowing it to learn patterns that might differentiate between individuals with and without ASD.**
3. **Evaluation During Training (optional): The script might include code to evaluate the model's performance on a small validation set during training to prevent overfitting.**

Model Saving:

1. **Trained Models (trained_model_adult.pkl, trained_model_toddler.pkl): These files likely store the serialized versions of the trained models (adult and toddler) using libraries like `joblib`. This allows you to save the model and load it later for prediction without retraining.**

Model Evaluation:

1. **Testing Script (testing_model.py): This script loads the trained models and uses them to predict ASD on unseen data from the testing sets (adult_testing_data.csv, toddler_testing_data.csv).**
2. **Performance Metrics: The script might calculate performance metrics like accuracy, precision, recall, F1-score, and generate a classification report to assess how well the models differentiated between ASD and non-ASD cases in the testing data.**

Optional: User Interface (GUI):

1. **GUI Script (form_gui.py): This script (if it exists) might be used to create a user-friendly interface allowing users to input data relevant to ASD diagnosis (age, gender, etc.) for the trained model to make predictions.**

Additional Files:

- **Helper Scripts: Scripts like `dataset.py` and `matplot.py` might be utilities for data manipulation, visualization of data distributions, model performance, or other insights to aid in analysis.**
- **Visualization Files: These files (potentially PNGs) could be plots generated during training, testing, or model evaluation to visualize data or model performance.**

Overall Workflow:

1. **Prepare data (potentially including preprocessing).**
2. **Train models on training data.**
3. **Save trained models.**
4. **Evaluate model performance on unseen testing data.**
5. **(Optional) Develop a GUI for user interaction and prediction.**
