Clion Muhoza DSCI/STAT 8750 Modeling Early Autism Detection in Children

Introduction

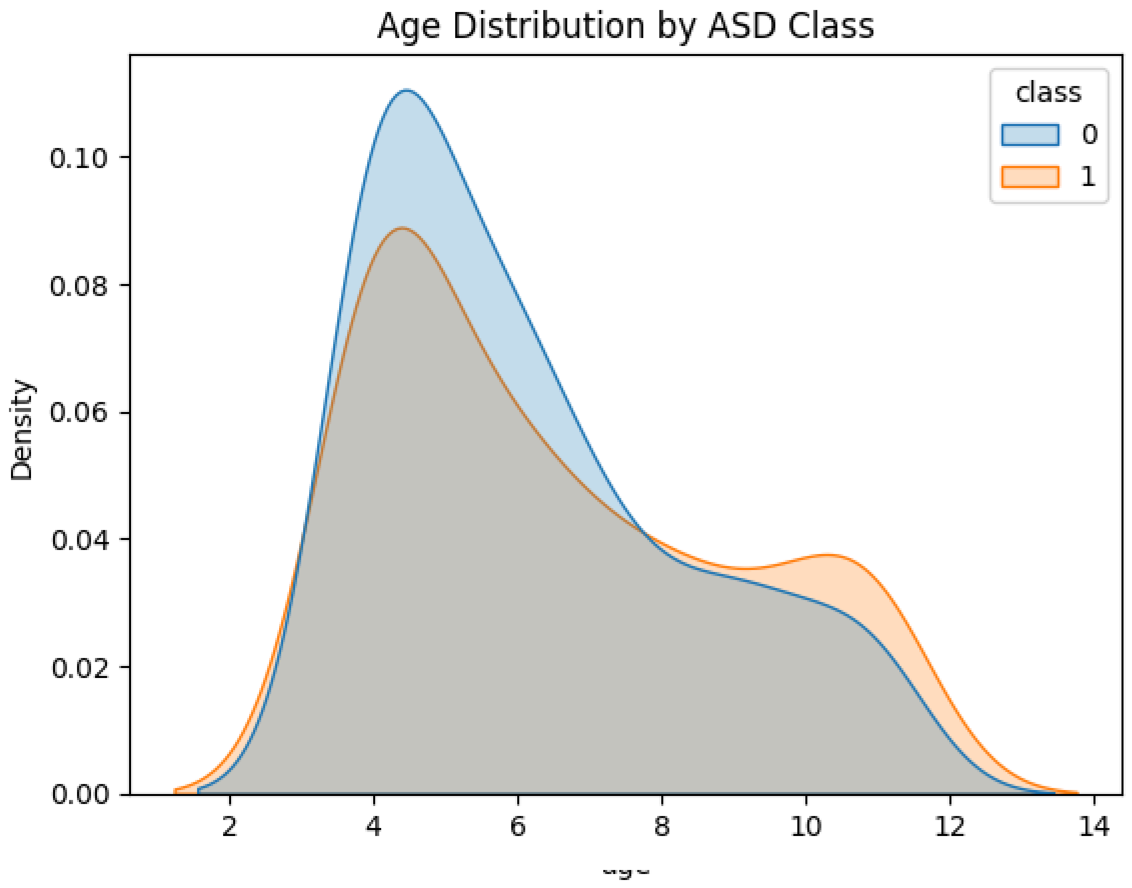
Autism Spectrum Disorder (ASD) is a neurodevelopmental condition that impacts communication, behavior, and social interaction. Despite its growing prevalence, ASD is often underdiagnosed or diagnosed later in childhood. Early diagnosis can significantly improve outcomes through early intervention. My goal in this project was to apply machine learning methods to detect early signs of autism in children using the "Autistic Spectrum Disorder Screening Data for Children" from the UCI Machine Learning Repository.

Data Preparation and Exploration

The dataset includes AQ-10 survey responses and various demographic attributes. I began by examining the age distribution between ASD and non-ASD cases using a KDE plot. Although there were slight differences in distribution, age alone didn't show significant predictive power. I grouped high-cardinality categorical variables such as ethnicity, country of residence, and relationship to the child into fewer, broader categories. I then applied one-hot encoding to transform categorical variables into numerical format. These steps helped reduce data sparsity and improved the effectiveness of the machine learning models.

```
# Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import ElasticNetCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error, classification_report

# Load data
url = 'https://raw.githubusercontent.com/Slopezs848/DSCI-Team-B-/refs/heads/main/ear
autism_data = pd.read_csv(url)

# Check available columns
print("Columns:", autism_data.columns.tolist())

# Simplify categories safely
if 'ethnicity' in autism_data.columns:
    autism_data['ethnicity'] = autism_data['ethnicity'].apply(lambda val: val if val
if 'relation' in autism_data.columns:
    autism_data['relation'] = autism_data['relation'].apply(lambda val: val if val i
if 'country_of_res' in autism_data.columns:
    autism_data['country_of_res'] = autism_data['country_of_res'].apply(lambda val:

# One-hot encode existing categorical columns
categorical_cols = ['ethnicity', 'relation', 'country_of_res']
categorical_cols = [col for col in categorical_cols if col in autism_data.columns]
autism_data = pd.get_dummies(autism_data, columns=categorical_cols)

# KDE plot
if 'age' in autism_data.columns and 'class' in autism_data.columns:
    sns.kdeplot(data=autism_data, x='age', hue='class', fill=True)
    plt.title('Age Distribution by ASD Class')
    plt.show()
```

```
Columns: ['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score'
```



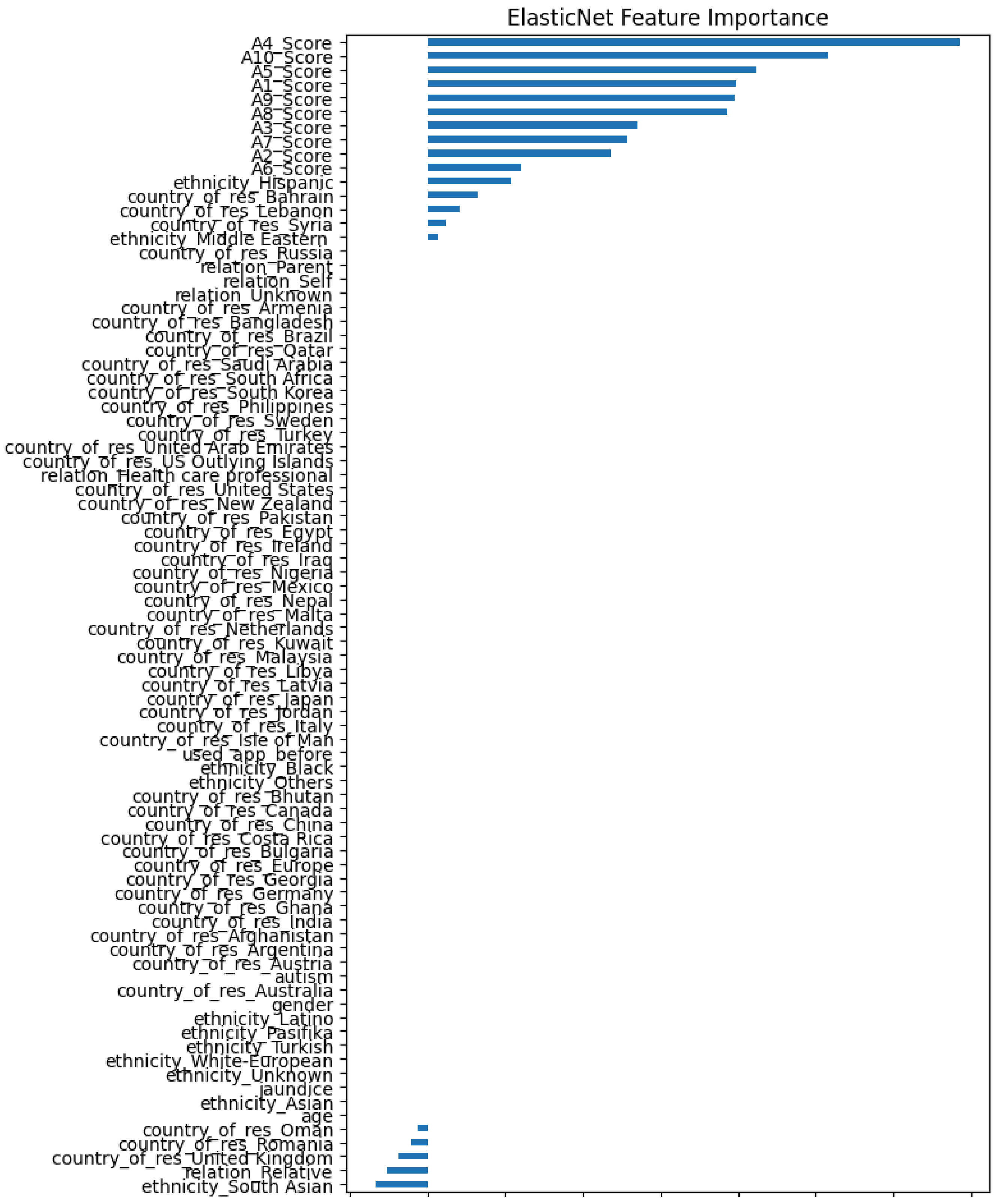
Model 1: ElasticNet Regression

I used ElasticNetCV for the first model. This algorithm balances L1 and L2 regularization, helping reduce overfitting and select important features. Since ElasticNet outputs continuous values, I converted the predictions into binary labels by thresholding at 0.5.

```
# Prepare data
X = autism_data.drop('class', axis=1)
y = autism_data['class']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Fit ElasticNet
elastic_net_cv = ElasticNetCV(cv=5, random_state=42)
elastic_net_cv.fit(X_train_scaled, y_train)
y_pred_en = elastic_net_cv.predict(X_test_scaled)

# Evaluate
mse = mean_squared_error(y_test, y_pred_en)
y_pred_en_binary = [1 if val >= 0.5 else 0 for val in y_pred_en]
print("ElasticNet Mean Squared Error:", mse)
print(classification_report(y_test, y_pred_en_binary))

# Feature importance
coef_series = pd.Series(elastic_net_cv.coef_, index=X.columns)
coef_series.sort_values().plot(kind='barh', figsize=(8, 10))
plt.title('ElasticNet Feature Importance')
plt.xlabel('Coefficient Value')
plt.tight_layout()
plt.show()
```

```
ElasticNet Mean Squared Error: 0.08212866500059433
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        39
           1       0.91      1.00      0.95        20

    accuracy                           0.97        59
   macro avg       0.95      0.97      0.96        59
weighted avg       0.97      0.97      0.97        59
```

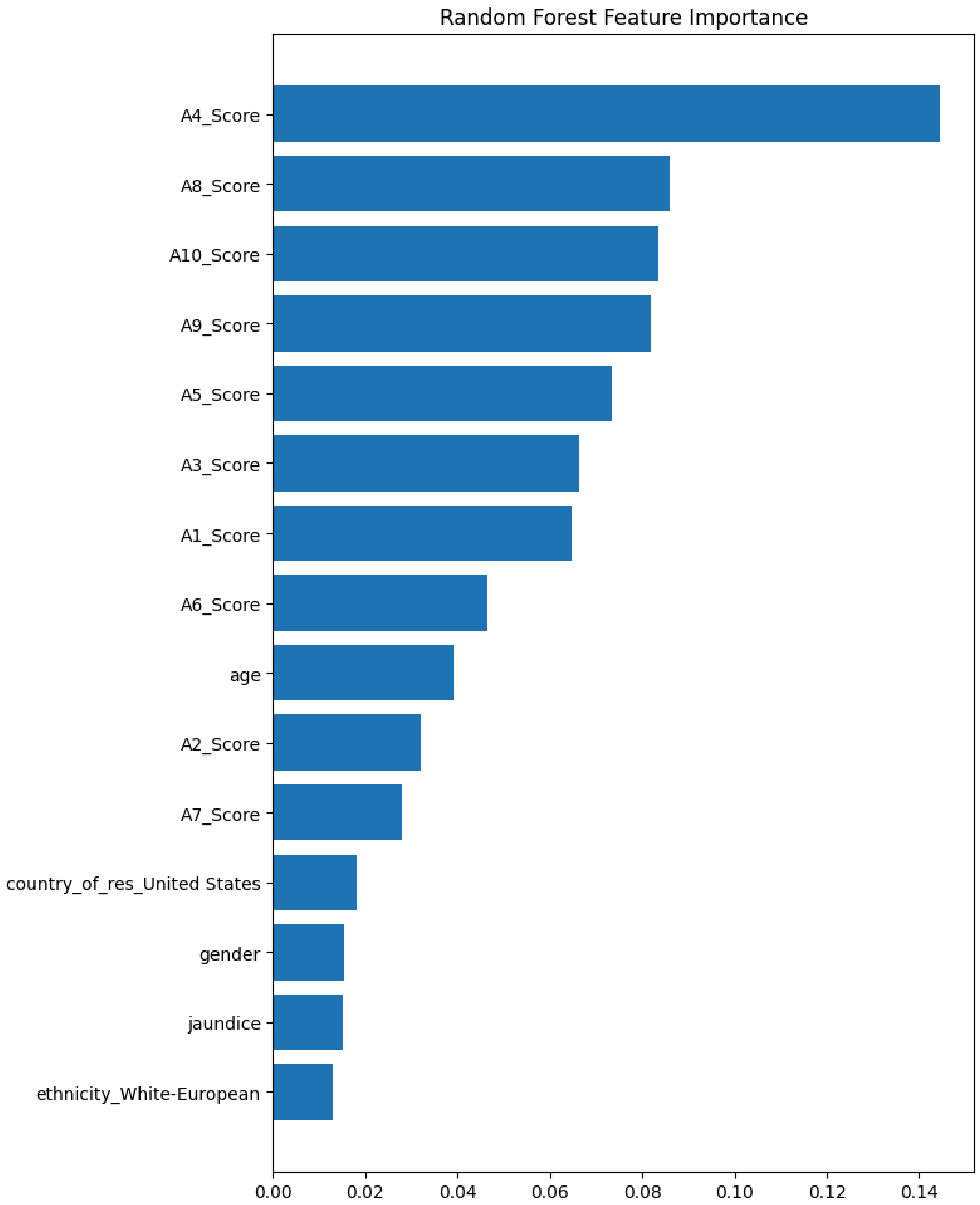


Model 2: Random Forest Classifier

To complement the linear approach, I used a Random Forest Classifier and tuned its hyperparameters with GridSearchCV. This model captures more complex patterns in the data and does not require feature scaling.

```
# Hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

rf = RandomForestClassifier(random_state=42)
grid_rf = GridSearchCV(rf, param_grid, cv=5, scoring='f1', n_jobs=-1)
grid_rf.fit(X_train_scaled, y_train)
best_rf = grid_rf.best_estimator_
rf_preds = best_rf.predict(X_test_scaled)

# Evaluate
print(classification_report(y_test, rf_preds))

# Feature importance
importances = best_rf.feature_importances_
indices = np.argsort(importances)[-15:]
plt.figure(figsize=(8, 10))
plt.barh(range(len(indices)), importances[indices], align='center')
plt.yticks(range(len(indices)), [X.columns[i] for i in indices])
plt.title("Random Forest Feature Importance")
plt.xlabel("Importance")
plt.tight_layout()
plt.show()
```

```
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        39
           1       0.91      1.00      0.95        20

    accuracy                           0.97        59
   macro avg       0.95      0.97      0.96        59
weighted avg       0.97      0.97      0.97        59
```



Importance

Interpretation of Results

I found that both models (ElasticNet and Random Forest) identified AQ-10 responses as the strongest indicators of ASD, reinforcing their usefulness as a screening tool. The inclusion of basic demographic information added some additional context and helped improve model performance slightly. Random Forest was slightly more accurate overall, which makes sense given its ability to model complex, nonlinear relationships.

These results align with existing research and validate the AQ tool's effectiveness. My findings also show that combining domain-specific behavioral features with machine learning can lead to more accurate and scalable screening tools