**Gender Identification**

This is a program using machine learning to predict the gender associated with a given name

Contents

[Summary 2](#_Toc181183860)

[Introduction 2](#_Toc181183861)

[Program Explanation 2](#_Toc181183862)

[1. \*\*Imports\*\*: 2](#_Toc181183863)

[2. \*\*File Variables\*\*: 2](#_Toc181183864)

[3. \*\*Loading and Saving Learned Data\*\*: 2](#_Toc181183865)

[4. \*\*Loading the Dataset\*\*: 2](#_Toc181183866)

[5. \*\*Training the Model\*\*: 2](#_Toc181183867)

[6. \*\*Loading the Model\*\*: 3](#_Toc181183868)

[7. \*\*Predicting Gender\*\*: 3](#_Toc181183869)

[8. \*\*Function Main()\*\*: 3](#_Toc181183870)

[9. \*\*Execution Block\*\*: 3](#_Toc181183872)

[Algorithm Outline 3](#_Toc181183873)

[1. \*\*Input\*\*: 3](#_Toc181183874)

[2. \*\*Parse Command-Line Arguments\*\*: 4](#_Toc181183875)

[3. \*\*Training Mode(if action is train)\*\*: 4](#_Toc181183876)

[4. \*\*Prediction Mode(if action is predict)\*\*: 4](#_Toc181183877)

# Summary

This Python code predicts gender based on names using a machine learning model with logistic regression and allows for interactive feedback to improve future predictions.

# Introduction

This document presents the Gender identification. where users enter names, see gender predictions, and add new names to improve model accuracy

# Program Explanation 7

## 1. \*\*Imports\*\*:

- pandas and numpy for data handling.

- pickle to save and load data in binary format.

- argparse to parse command-line arguments.

- scikit-learn functions for machine learning, including train\_test\_split, LogisticRegression, and others used for training the model and evaluating it.

## 2. \*\*File Variables\*\*:

Two files are used to save data:

- learned\_data\_file: Stores custom learnings, a dictionary of names with their corresponding genders (from user feedback).

- model\_file: Stores the trained model for future use.

## 3. \*\*Loading and Saving Learned Data\*\*:

These functions manage the learned data, which is stored in learned\_data\_file.

* load\_learnings() checks if the file exists. If yes, it loads the data with pickle. If no file is found, it starts with an empty dictionary.
* save\_learnings(learnings) saves the learned data to learned\_data\_file using pickle.

## 4. \*\*Loading the Dataset\*\*

- load\_dataset(filename) loads a CSV file specified by filename, which should contain three columns: name, gender, and probability.

## 5. \*\*Training the Model\*\*:

- The train\_model(df) function takes a Data-frame as input and trains a logistic regression model for gender prediction:.

1.Vectorization: Uses CountVectorizer to turn names into a set of bi-gram and trig-ram character features.

2. Encoding Labels: Uses Label Encoder to encode the gender column into numeric values.

3.Sample Weights: The probability column is used as sample weights, allowing names with higher confidence levels to have more influence during training.

4.Train-Test Split: Splits data into training and testing sets (80/20 split) using train\_test\_split.

5. Model Training: Trains a logistic regression model on the training set with weights and then evaluates it on the test set.

6. Saving the Model: saves the trained model, vectorizer, and label encoder to model\_file using pickle.

## 6. \*\*Loading the Model\*\*:

- The function checks if a trained model file exists. If so, it loads the model, vectorizer, and label encoder. If not, it displays a message to indicate that training is required.

## 7. \*\*Predicting Gender\*\*:

- The predict\_gender\_ml function is the main function for making gender predictions:

- First, it checks if the name is already in learning's. If so, it returns the learned gender without using the model.

- If the name is not in learning's, it transforms the name into feature vectors using vectorizer and predicts gender using model.

- The predicted label is decoded back to male or female using Label Encoder.

## 8. \*\*Function `main()`\*\*:

- The main() function handles the overall program flow. It uses arg parse to allow command-line arguments:.

- csvfile: Optional, specifies the data set file for training.

- action: Either train (to train the model) or predict (to make predictions).

- If train is specified, it loads the data set and trains the model. Otherwise, it loads a pre-trained model for prediction. Once loaded, it enters an interactive loop:

- Continuously prompts the user for a name, predicts gender, and asks for feedback on the accuracy.

- If the prediction is incorrect, it prompts the user for the correct gender and updates learning's, which is then saved to learned\_data\_file.

## 9. \*\*Execution Block\*\*:

- The loop continues until the user decides to stop, at which point the program exits.

# Algorithm Outline

## 1. \*\*Input\*\*:

- Define file paths for storing learned data and the model.

## 2. \*\* **Parse Command-Line Arguments**: \*\*:

- Check if the action is train or predict.

## 3. \*\* **Training Mode** (if action is train): \*\*:

- Load dataset from CSV.

- Transform names into character-based features (bigrams and trigrams).

- Encode gender labels into numeric values.

- Train a logistic regression model with sample weights.

- Save the model, vectorizer, and encoder.

## 4. \*\* **Prediction Mode** (if action is predict): \*\*:

- Load the saved model, vectorizer, and encoder (if available).

Start an interactive loop:

- Prompt user for a name.

Check if the name is in learned data:

-If yes, use the stored gender. If no, predict gender using the model.

-Display predicted gender and ask if it’s correct. If incorrect, get the correct gender from the user and save it in learned data.

5. \*\*Exit\*\*:

- Allow user to continue entering names or type exit to quit the loop.

- Save updated learned data back to the file.