Introduction to Data Science

Gender By Name

Group Number: 26
Group Name: Random 4

Prepared by:-

Aneesh Khunteta 20UCS020

Atul Ratnawat 20UCC026

Dhruv Sharma 20UCC038

Kumar Pratyush 20UCC057

Abstract

When creating an account , several large Internet companies no longer require gender information. On the other hand, the prediction of demographic data like gender and age still poses a serious challenge, especially when recommender systems unintentionally exhibit gender and age bias. Therefore, it is necessary to guess the gender of users who did not provide this information during registration. We investigate the difficulty of figuring out a registered user's gender from their given name.

After analysing the beginning names of thousands of users, we found that genders can be classified pretty successfully using the composition of name strings. We offer character-based machine learning models and show that, when determining the gender of users, our algorithms are significantly more accurate than baseline methods. Furthermore, we show that further improving classification performance by including the last names with the first names.

TABLE OF CONTENTS

SECTION I : Introduction

- Introduction to Gender by Name. 4

SECTION II : Dataset

- Description about Dataset. 5
- Preprocessing on Dataset..... 6

SECTION III : Analysis of Dataset

- Number of Male & Female..... 7
- Most Common and Uncommon Names..... 7
- Top Names..... 8
- Length of Names. 8
- The Last Letter in A Name Is a Vowel. 9
- Last character in Data set among male and females. 9
- Heatmap of the Female Probability..... 10

SECTION IV : Naive Bayes Algorithm

- What is Naive Bayes algorithm?. 13
- How Naive Bayes algorithm works?..... 14
- Why we used Naive Bayes Classification?. 15

SECTION V : Project Code

- Description of Project Code. 16

SECTION VI : Output

- Predicting Gender on the basis of Name?. 19

SECTION VII : Conclusion

- Conclusion 21

Introduction

"A considerable portion of registered users do not specify their gender, and this number is anticipated to grow over time. Knowing consumers' demographic information, of which gender is a significant component, is vital for various tasks, such as when examining the fairness of machine learning algorithms for content recommendation. By using data gathered at their registration and/or from their subsequent behaviours, it is necessary to deduce the gender of those registered users who do not disclose it. Despite the fact that gender identity is not always binary, in this study we treat gender inference as a binary classification problem.

Examining user behaviour is one method for determining gender. A content-based approach, which we tested, extracts features from the content that users engage with through actions like surfing. The information is condensed into categorised features such as wiki-entities (such as Wikipedia entries) and custom content categories obtained using ML models. A logistic regression technique uses these category characteristics to categorise user genders. The benefit of such a content-based architecture is that it recognises users based on cookies from their browsers, making it functional for users who are not registered or logged in.

However, a drawback of such content-based strategies is that user signals may be spotty. Signals could also be lost during the categorization of content into feature categories. As a result, we discovered that this content-based method performs only somewhat well.

Because we have more data about registered users, we are therefore interested in improving the aforementioned baseline model. We think that the user's name in particular may be utilised to efficiently infer gender. In light of this, we suggest a character-based machine learning strategy for the problem of gender classification.The major contributions of our work are:

• We propose several character-based techniques to predict users' genders based on their first name.

• We conduct our experiment on a very large real-world dataset of 150k unique first names and show that our models trained on this large dataset extend better than the same models trained on a medium sized public dataset. We also show that our models perform well on unseen or multilingual datasets.

• We demonstrate that the performance could be further enhanced when utilizing both the first and the last names through our proposed algorithm. We show that fusing name and content information can also boost performance.

Throughout the report, we shall denote *P {M)/P {F)* the probability of male/female, and *P {M X)* the probability of male given feature set X. The goal of the proposed machine learning models is to compute *P {M X),* based on the user feature set X. All models in this report minimize the binary entropy loss function.

Description about dataset

This dataset combines raw counts for first names of male and female babies in those time periods, and then calculates a probability for a name given the aggregate count. Source datasets are from government authorities:

- US: Baby Names from Social Security Card Applications - National Data, 1880 to 2019
- UK: Baby names in England and Wales Statistical bulletins, 2011 to 2018
- Canada: British Columbia 100 Years of Popular Baby names, 1918 to 2018
- Australia: Popular Baby Names, Attorney-General's Department, 1944 to 2019

Description of Dataset

- Dataset Source: https://archive.ics.uci.edu/ml/datasets/Gender+by+Name
- Dataset Characteristics: Text
- Subject Area: Social
- Number of Attributes: 4
- Associated Tasks: Classification Number of Instances: 147269
- Programming Language Used: Python
- Platform: Google Colaboratory

- Attribute Information:

  - Name: String
  - Gender: M/F (category/string)
  - Count: Integer
  - Probability: Float

Preprocessing on Dataset

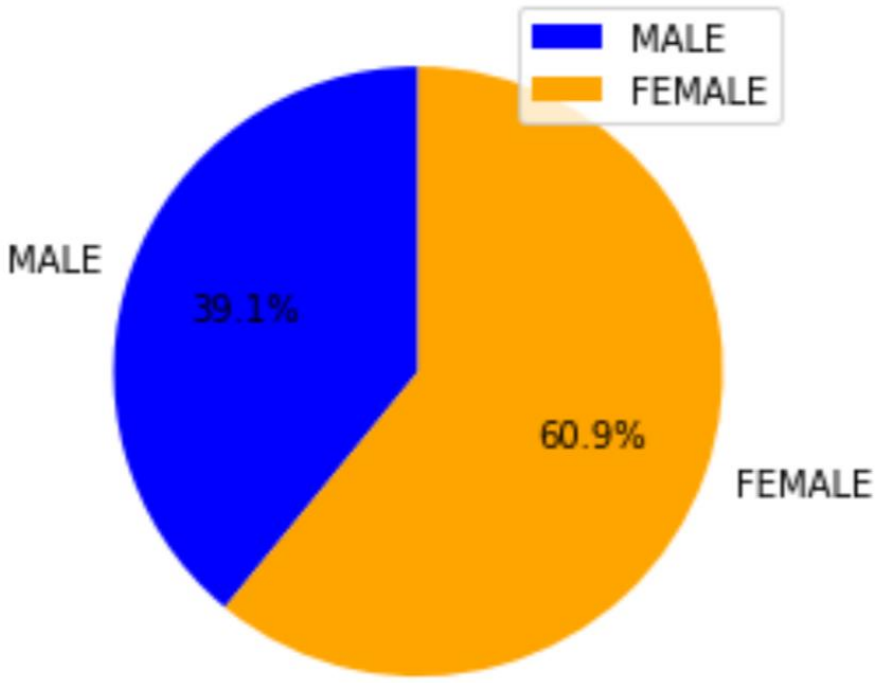
We have replaced “M” to 1 and “F” to 0 in the gender column of the dataset because it makes the algorithm directly use 1 and 0 during the training. Then we have removed any extra character other than the alphabet in the name because we have created a bag of 2 grams of characters and made the name as lowercase because uppercase will not matter in the training purpose. We have calculated frequency arrays for the last letter of male and female names, and frequency array for length of names. We have also pre calculated from the given data-set the frequency for last and second last character pair for male and female both in order to analyze the names.

Data Analysis

From the dataset we have explore the data a little and visualise some summary stats using python script.

1) Number of Male & Female

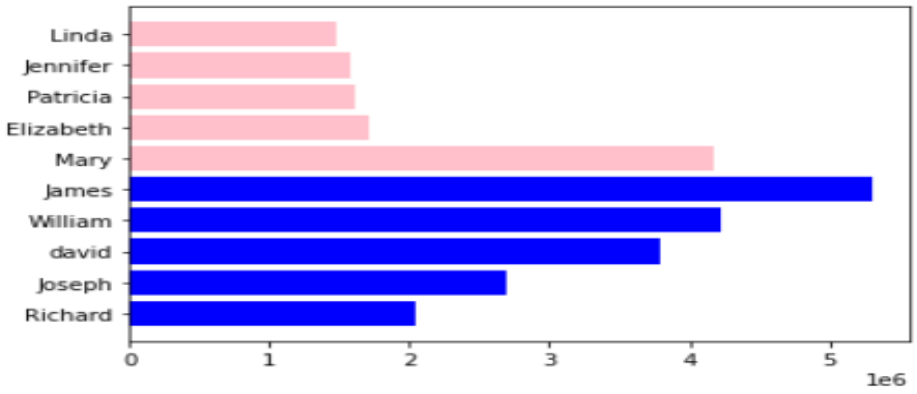
In Our dataset there are total 147,269 names. Out of them 57,520 names are of Male which are 39.1% of the total entries and rest 89,749 are of Females which are remaining 60.9% of total entries *calculated from the preprocessed data*.



2) Most common and uncommon names

A word cloud might serve to visually depict the fact that some names are preferred by parents each year. For instance, Elizabeth, Katherine, and Margaret are common female names, whereas James, William, and John are popular boy names.
It's interesting to learn a little bit about rare names in addition to typical names. Baelyn, Selda, and Demani are three highly fascinating and uncommon names for females, while Rodarius, Jamaul, and Kelvon are three for boys.

3) Top Names



Top 5 names

Reference:- https://www.whattoexpect.com/baby-names/list/top-baby-names-for-boys/

Python Script for above plot:-

```
#for bar graph for top 5 names in the data set
#Richard,Joseph,David,William,James were the top boys name for 2022 according to https://www.whattoexpect.com/baby-names/list/top-baby-names-for-boys/



x = np.array(["Richard","Joseph","david","William","James"])
y = np.array([2039641, 2695970, 3787547, 4226608,5304407])

plt.barh(x,y,color="blue")

#Mary,Elizabeth,Patricia,Jennifer,Linda were the top girl names

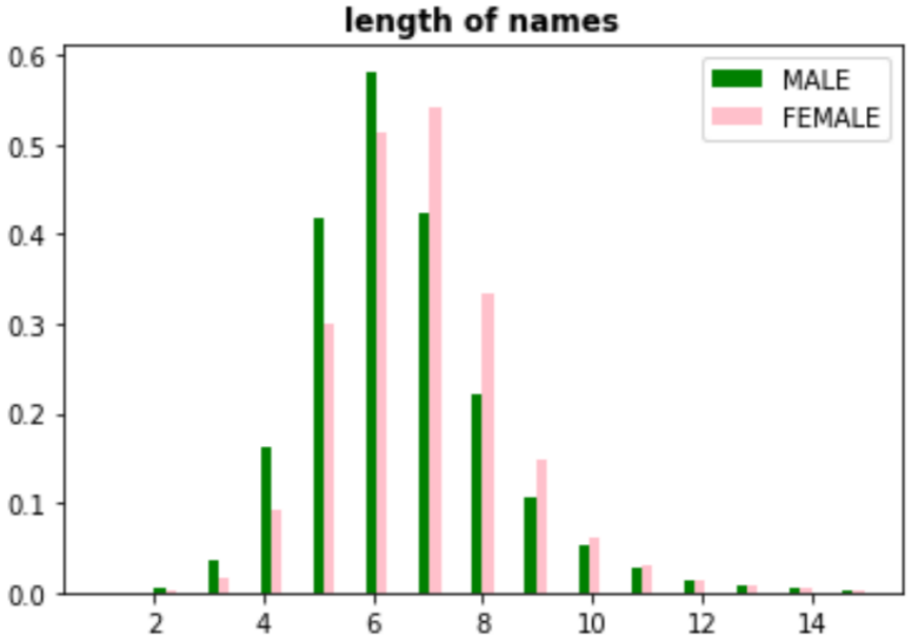
x = np.array(["Mary","Elizabeth","Patricia","Jennifer","Linda"])
y = np.array([4169663, 1704140, 1608260, 1584426,1480592])

plt.barh(x,y,color="pink")


plt.show()
```

4) Length of names

Generally, boys are given shorter names than girls. So, we plotted the name lengths as a histogram. It seems that girls tend to have slightly longer names than boys and most of the names would be about 5 letters long.



Length of names vs Probability

Python Script for above plot:-

```
# ****** Histogram for length of names********
n_bins1 = np.linspace(1,15,30)


colors = ['green','pink']
labels = ['MALE','FEMALE']

plt.hist([lengthM, lengthF], n_bins1,density=True, color = colors, label = labels)

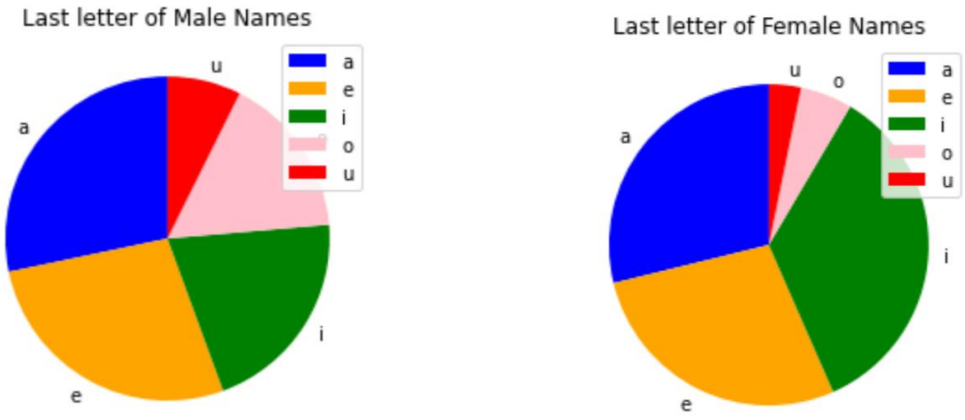
plt.legend(prop ={'size': 10})

plt.title('length of names', fontweight ="bold")

plt.show()
```

Here lengthM and lengthF are the array processed from dataset having frequencies of names stored

5) The last letter in a name is a vowel



Percentage of names ending in vowels

Girls tend to have names that end in vowels, while boys don't. To test this we plotted a pie chart. We were correct in our assumption. Names of girls largely tend to end in vowels, especially ‘a’ and ‘e’, while names of boys predominantly end in ‘e’ or ‘o’.

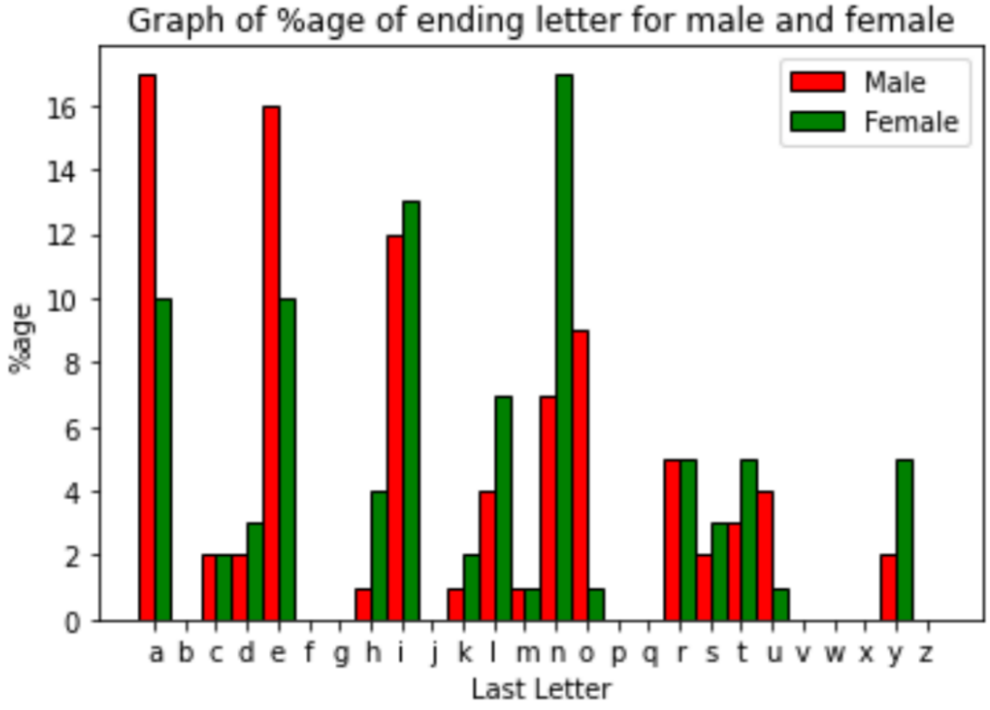
Python Script for above plots:-

```
# Pie chart for last letter vowels for both male and female seperately
arrM=[]
sumM=0
for i in range(0,26):
  x = chr(i+97)
  if x=='a' or x=='e' or x=='i' or x=='o' or x=='u':
    arrM.append(lastIndexM[i])
    sumM+=lastIndexM[i]
for i in range(0,5):
  arrM[i]/=sumM
  arrM[i]*=100
print(arrM)
```

```
arrF=[]
sumF=0
for i in range(0,26):
  x = chr(i+97)
  if x=='a' or x=='e' or x=='i' or x=='o' or x=='u':
    arrF.append(lastIndexF[i])
    sumF+=lastIndexF[i]
for i in range(0,5):
  arrF[i]/=sumF
  arrF[i]*=100
```

6) The last character in a name

Concluded from the graph made from given dataset majority male names end with letter ‘a’, ‘e’ and majority of female names end with ‘i’ and ‘o’. and some letters like ‘b’, ‘v’, ‘p’, ‘q’ , ‘z’, etc. don’t even used for ending the names.



Python Script for above plot:-

```
# Calculating percentage for frequency arrays
for i in range(0,26):
   sum1+=lastIndexM[i]
   sum2+=lastIndexF[i]
for i in range(0,26):
      lastIndexM[i]*=100
      lastIndexF[i]*=100
      lastIndexM[i]=(float)(lastIndexM[i]/sum1)
      lastIndexF[i]/=sum2
```

```
# Plotting the bar-graph
n_bins=[]
for i in range(0,26):
  n_bins.append(chr(i+97))

n=26
r = np.arange(n)
width = 0.5


plt.bar(r, lastIndexM, color = 'r',
        width = width, edgecolor = 'black',
        label='Male')
plt.bar(r + width, lastIndexF, color = 'g',
        width = width, edgecolor = 'black',
        label='Female')
```

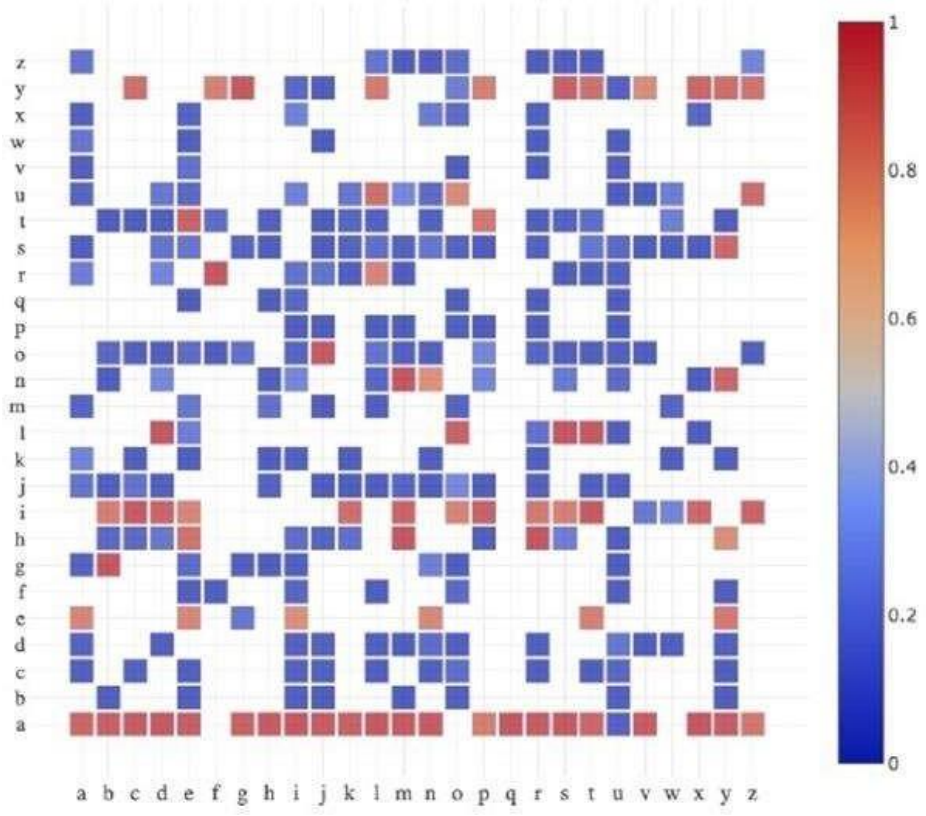
```
plt.xlabel("Last Letter")
plt.ylabel("%age")
plt.title("Graph of %age of ending letter for male and female")

# plt.grid(linestyle='--')
plt.xticks(r + width/2, n_bins)
plt.legend()

plt.show()
```

7) Heatmap of the female probability

Heatmap of the female probability, *P {F)*, given the last two characters. X-axis is the 2nd last character, y-axis the last character. Only showing character 2-grams with *P {F)* > 0.7 or *P {F)* <0.3, and frequency > 40.



Python Script for Generating Heat Map:-

```
#heatmap for females
#with x-axis being the last character
#y-axis being the second last character

rows, cols = (26, 26)
arr = [[0]*cols]*rows

for idx in data.index:
   if data["Gender"][idx]==0:
      x=ord(data["Name"][idx][len(data["Name"][idx])-1])-97
      y=ord(data["Name"][idx][len(data["Name"][idx])-2])-97
      arr[x][y]+=1
for x in range(26):
   sumx=0
   for y in range(26):
      sumx+=arr[x][y]
   for y in range(26):
      arr[x][y]=(arr[x][y])/sumx;
#print(arr)
ax = sns.heatmap( arr , linewidth = 0.5 , cmap = 'coolwarm' )

plt.title( "2-D Heat Map" )
plt.show()
```

What is Naive Bayes algorithm?

It is a classification method built on the Bayes Theorem and predicated on the idea of predictor independence. A Naive Bayes classifier, to put it simply, believes that the presence of one feature in a class has nothing to do with the presence of any other feature.

A fruit might be categorised as an apple, for instance, if it is red, rounded, and around 3 inches in diameter. Even if these characteristics depend on one another or on the presence of other characteristics, each of these traits separately increases the likelihood that this fruit is an apple, which is why it is called "Naive."

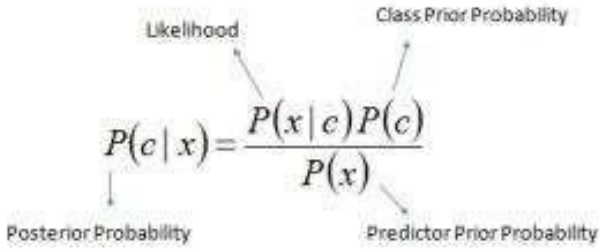
Simple to construct and especially helpful for very big data sets is the naive Bayes model. Along with being straightforward, Naive Bayes is known to perform better than even the most complex classification techniques.

Bayes theorem provides a way of calculating posterior probability P(c x) from P(c), P(x) and P(x c). Look at the equation below:

Below,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.

P(x) is the prior probability of predictor.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

How Naive Bayes algorithm works?

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P (Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes |Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Why we used Naive Bayes Classification?

It is easy and fast to predict class of test data set. It also performs well in multi class prediction.

Here assumption of independence holds, so a Naive Bayes classifier performs better compare to other models like logistic regression and we need less training data.

Major Project Code

```
from matplotlib import pyplot as plt
from os.path import exists
import pandas as pd
import re
import numpy as np
import seaborn as sns
```

We have used pandas and NumPy packages to do data preprocessing and classification of names. To do that we have imported pandas and NumPy packages. Majorly we have derived and saved data in various data structures from the csv file data set (downloaded from preferred website) and used them to process some informations like plotting various graphs and heatmaps.

```
if exists("trained_data.npy"):
    matrix = np.load("trained_data.npy")
    lastIndexM = np.load("lastIndexM.npy")
    lastIndexF = np.load("lastIndexF.npy")
    heatMapM = np.load("heatMapM.npy")
    heatMapF = np.load("heatMapF.npy")
    lengthF = np.load("lengthF.npy")
    lengthM = np.load("lengthM.npy")
```

fsfdsaffdsaf

```
else:
    data = pd.read_csv("name_gender_dataset.csv")
    data["Gender"].replace({"F": 0, "M": 1}, inplace=True)
    rows, cols = (26, 26)
    heatMapM = [[0]*cols]*rows
    heatMapF = [[0]*cols]*rows
    lengthF=[]
    lengthM=[]
    for idx in data.index:
        data["Name"][idx] = re.sub(re.compile("[^a-zA-Z]"),"", data["Name"][idx])
    data["Name"] = data["Name"].str.lower()
    matrix = []
    for idx in data.index:
        array = np.zeros(677)
        ch="i"
        for i in range(len(data["Name"][idx])-1):
            index = (ord(data["Name"][idx][i])-97)*26 + (ord(data["Name"][idx][i+1])-97)
            array[index] = 1
            # print("Hello there")
            ch=data["Name"][idx][i]
        array[676] = data[ "Gender"][idx]
        matrix.append(array)
        x=ord(data["Name"][idx][len(data["Name"][idx])-1])-97
        y=ord(data["Name"][idx][len(data["Name"][idx])-2])-97
        if  data[ "Gender"][idx]==1:
            lengthM.append(len(data["Name"][idx]));
            lastIndexM[ord(ch)-97]+=1
            heatMapM[x][y]+=1
        else:
            lengthF.append(len(data["Name"][idx]));
            heatMapF[x][y]+=1
            lastIndexF[ord(ch)-97]+=1
    for i in range(0,26):
        for j in range(0,26):
            heatMapF[i][j]=(heatMapF[i][j]*1.0)/(heatMapM[i][j]+heatMapF[i][j])

    np.save('trained_data', matrix)
    np.save('lastIndexM', lastIndexM)
    np.save('lastIndexF', lastIndexF)
    np.save('heatMapM', heatMapM)
    np.save('heatMapF', heatMapF)
    np.save('lengthM', lengthM)
    np.save('lengthF', lengthF)
```

We first preprocess the dataset, so for that we have to read the dataset using the function read csv of pandas library. Then we replaced in the gender column “M” to 1 and “F” to 0. So that we can directly use them as 0 and 1 in the algorithm. We used regular expression library functions to remove extra characters other than small and capital letters. We also changed capital letters to smallletters using methods in the string library.

After preprocessing the dataset we have created a matrix of(26x26 + 1) columns. First 26x26 columns store 2-gram occurrences of letters in the name i.e. if we have a name as “Prachi” we will mark the column corresponding to “pr”, “ra”, “ac”, “ch”, “hi” as 1 and rest letters as 0. For the last column we are storing the gender of the name like in the given case we are storing 0 in the last column. We have also saved our training dataset as trained data so that we do not have to do training on given dataset every time.

```
M = 0
F = 0
data_size = len(matrix)
for array in matrix:
   if array[676] == 1:
      M += 1
   else:
      F += 1
p_m = M/data_size
p_f = F/data_size
```

To use Naive Bayes Classifier we first calculated the probability of male names and females names in the trained dataset. To do that we are iterating our trained dataset and incrementing number of males and females whenever we encounter a 1 and 0 respectively in the gender column. After that we divided the number of males with total names as well as number of females with total names to get individual probability of males and females.

Naïve Bayes Classifier Script:-

```
# --------------------------------Taking Input of names to identify their genders----------------------------------------------------------



name = input("Enter Name: ")
name = re.sub(re.compile("[^a-zA-Z]"),"", name).lower()
for i in range(len(name)-1):
    index = (ord (name[i])-97)*26 + (ord (name[i+1])-97)
    m = 0
    f = 0
    for array in matrix:
        if array[index]==1:
            if array[676]==1:
                m += 1
            else:
                f += 1
    p_m *= m/M
    p_f *= f/F
if p_m >= p_f:
    print("Male")
else:
    print("Female")
print("M", round (p_m/(p_m+p_f), 2))
print("F", round(p_f/(p_m+p_f), 2))
```

We are then taking names as input and then we are removing any whitespace or any other characters other than the alphabet from the name and we have also done lowercasing of all the letters in the name. Then we iterate for the name and calculate the product of each 2-gram given its a male. Similarly, we have done for females i.e., we have calculated the product of the probability of 2-gram given female. We then compare these two products and assign the gender to the one with high value. We have also returned the normalized probability that we computed using those above two products. These normalized probabilities represent that with what probability our algorithm classifies a given name as male and female.

Output

As in the output we can see for the names “Lokendra Singh Hada” our algorithm has correctly classified as male with probability of 0.63 and also for names like “Diya Thakur” it classifies correctly as female with probability of 0.85 and outstanding results were shown for the name “Dhruv Sharma” where it classified with 1.0 probability as male.

```
print("F", round(p_f/(p_m+p_f), 2))
```

```
Enter Name: Aesha Jain
Female
M 0.19
F 0.81
```

```
Enter Name: Dhruv Sharma
Male
M 1.0
F 0.0
```

```
Enter Name: Diya Thakur
Female
M 0.15
F 0.85
```

```
Enter Name: Kumar Pratyush
Male
M 0.91
F 0.09
```

```
Enter Name: Lokendra Singh Hada
Male
M 0.63
F 0.37
```

```
Enter Name: Riya Mehta
Female
M 0.07
F 0.93
```

Conclusion

With the rise of social media in recent years, there has been a surge in interest in automatically identifying users based on their informal content. In this context, the research of other aspects intrinsic to users, such as political inclinations, personality, and gender, as well as the categorization of users in demographic categories such as age, ethnicity, origin, and race has gained a lot of interest notably based on Twitter data. Foreseeing orientation by the primary name is definitely not a straightforward undertaking. In numerous applications, particularly in the regular language handling (NLP) field, this assignment might be fundamental, mostly while thinking about unfamiliar names. In this task, we considered the issue of deducing orientation from clients' names. We proposed character-based AI models, and exhibited that our models can gather the sexes of clients with a lot higher precision than models in light of content data, or in view of name embeddings. One limit of involving names as the fundamental sign for ordering sexual orientations is that clients may not unveil their actual names. Another restriction is that orientation might be hard to surmise assuming the principal name is gender neutral. We showed independently that it is useful to utilize the first and last names, or the primary name and content data. For future work, we mean to research whether consolidating these the two of them together (names and content data) can give us far and away superior execution.