MACHINE LEARNING

(Name Gender Classifier)

Summer Internship Report Submitted in partial fulfillment

Of the requirement for undergraduate degree of

Bachelor of technology

In

Computer Science Engineering

By

Padakanti Rohith

221710312038

Under the Guidance of

Mrs.K.Neha

Assistant Professor

GITAM

(DEEMED TO BE UNIVERSITY)

VISAKHAPATNAM HYDERABAD BENGALURU

Department of Computer Science Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled “NAME GENDER CLASSIFIER” to GITAM (Deemed To Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of “Bachelor of Technology” in “Computer Science Engineering”. I declare that it was carried out independently by me under the guidance of Mrs. K.Neha, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD	**P.Rohith**
Date:14-07-2020	**2217103012038**

CERTIFICATE

This is to certify that the Industrial Training Report entitled “NAME GENDER CLASSIFIER” is being submitted by Padakanti Rohith in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20.

It is faithful record work carried out by her at the Computer Science Engineering Department, GITAM University Hyderabad Campus

under my guidance and supervision.

Mrs.K.Neha	**Prof.S.Phani Kumar**
Assistant Professor	Professor and HOD
Department of CSE	Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad**, Pro Vice Chancellor,GITAM Hyderabad and **Prof.N.Seetha Ramaiah,** Principal, GITAM Hyderabad.

I would like to thank respected **Prof.S.Phani Kumar** Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mrs.K.Neha** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

P.ROHITH

221710312038

ABSTRACT

Studying the structure of given names and how they associate with gender and ethnicity is an interesting research topic that has recently found practical uses in various areas. Given the paucity of annotated name data, we develop and make available a new dataset containing 14k given names. Using this dataset, we take a datadriven approach to this task and achieve up to 90% accuracy for classifying the gender of unseen names. For ethnicity identification, our system achieves 83% accuracy. We also experiment with a feature analysis method for exploring the most informative features for this task.

We present a Support Vector Machine (SVM) based classification approach for gender prediction of names. We first identify various features based upon morphological analysis that can be useful for such classification and evaluate them. We then state a novel approach of using n-gram-suffixes along with these features which gives us significant advantage over the baseline approach. We believe that we are the first to use n-grams of suffixes instead of the whole word for predictor systems. Our system reports a top F1 score of 94.9% which is expected to improve further with increase in training data size.

P.Rohith

221710312038

Table of Contents

List of Figures:

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

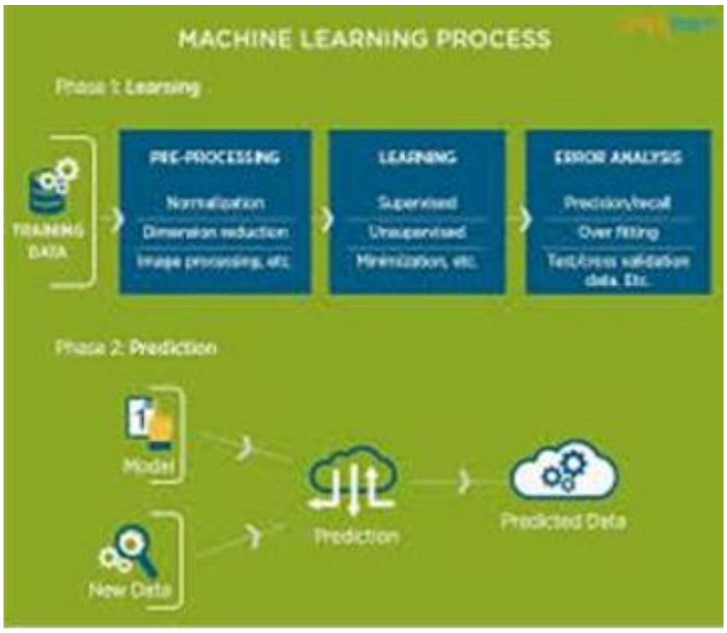


Figure 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data. By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning. Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to “learn” how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan.Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

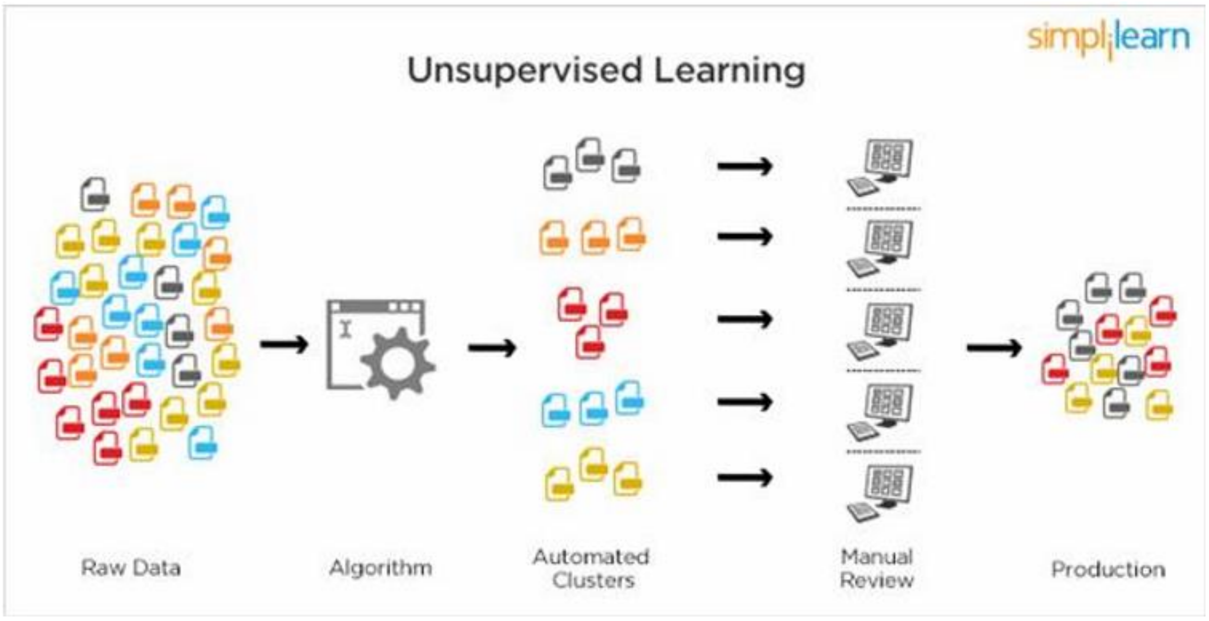


Figure 2 : Unsupervised Learning.

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

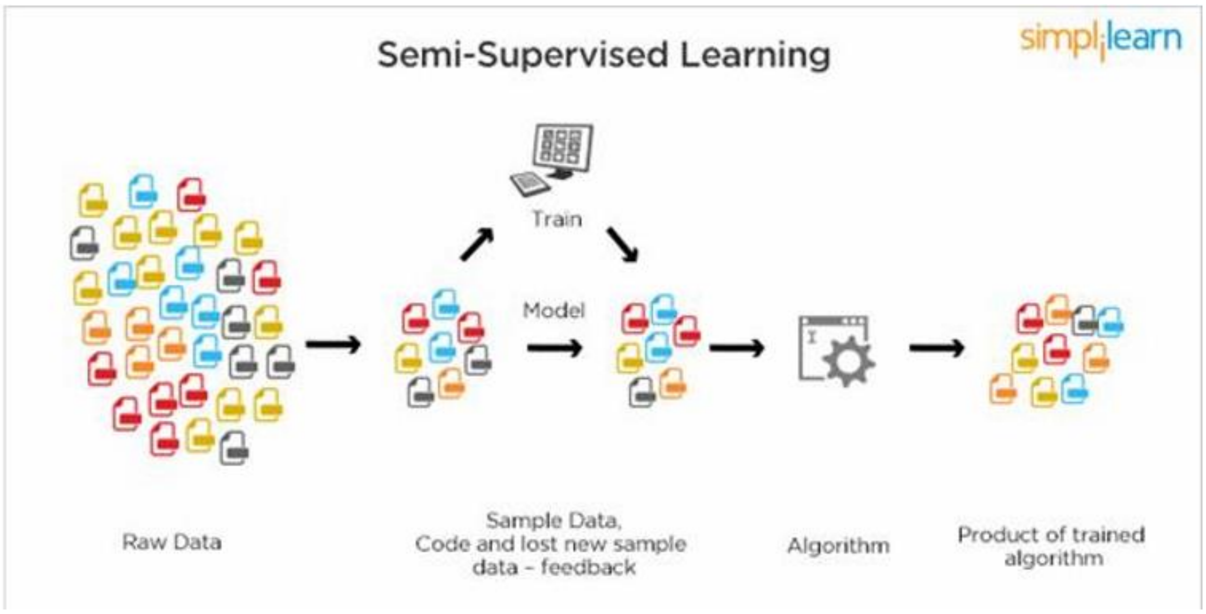

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Figure 3 : Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990’s.
- Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain:** Python's source code is fairly easy-to-maintaining.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

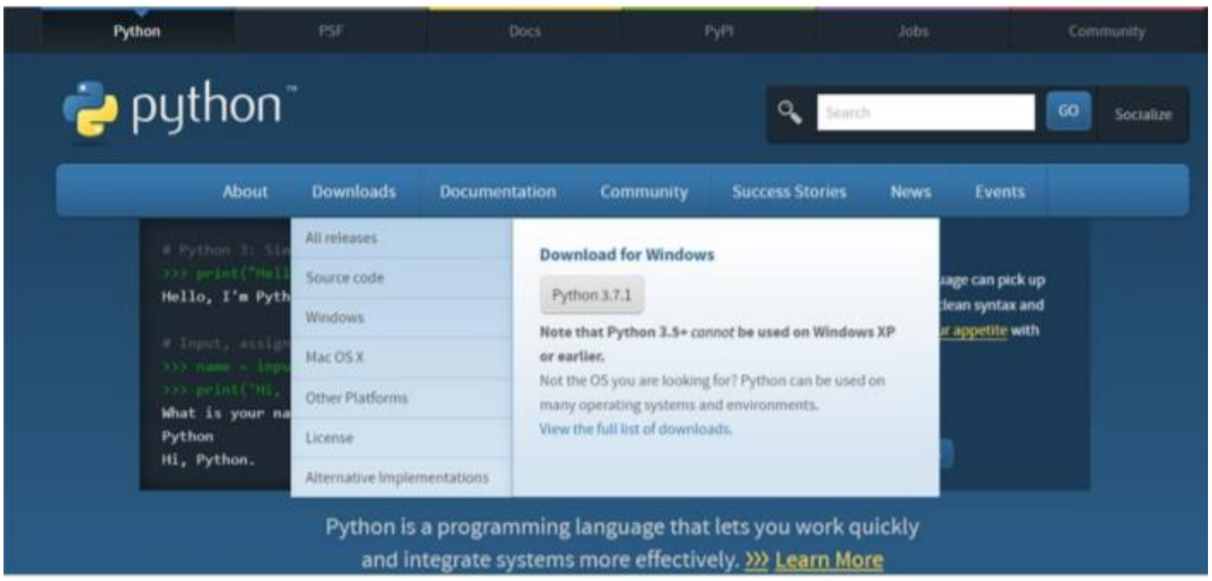
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python



Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

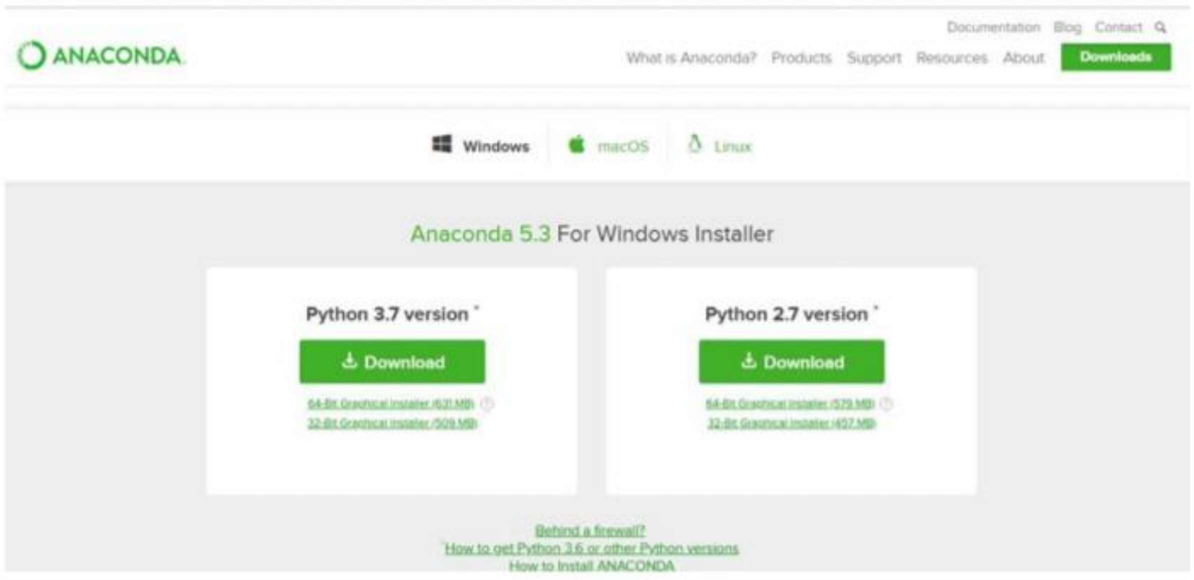
Not the OS you are looking for? Python can be used on many operating systems and environments.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 4 : Python download

2.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.
- In WINDOWS:
- Step 1: Open Anaconda.com/downloads in a web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)



ANACONDA.
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version *
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

Figure 5 : Anaconda download

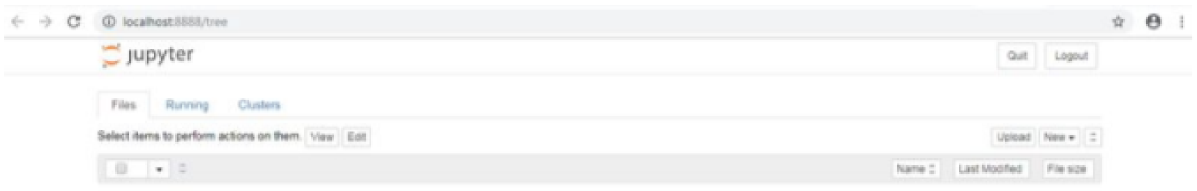


Figure 6 : Jupyter notebook

2.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
    - Numbers
    - Strings
    - Lists
    - Tuples
    - Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types – int (signed integers) long (long integers, they can also be

represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example - Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

2.6.2 Calling a Function:

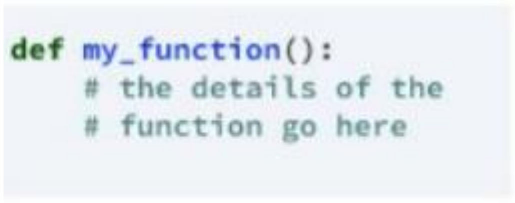
Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOP's CONCEPTS:

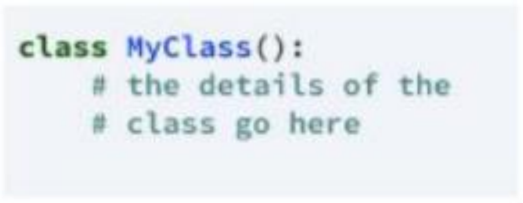
2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

- Defining a Class: o We define a class in a very similar way how we define a function. o Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is indented like a function body is.



```
def my_function():
    # the details of the
    # function go here
```



```
class MyClass():
    # the details of the
    # class go here
```

Figure 7 : Defining a Class

2.7.2 __init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

To predict whether the given Name is Male or Female.

3.2 DATA SET:

The given dataset contains following parameters:

This dataset presents names different names of male and female genders. The dataset contains nearly two to five thousand names.

It contains only name and gender of the particular person.

3.3 OBJECTIVE OF THE CASE STUDY:

Based upon the name we can easily classify whether the name comes under male or female gender.

CHAPTER 4

MODEL BUILDING

4.1 PREPROCESSING OF THE DATA:

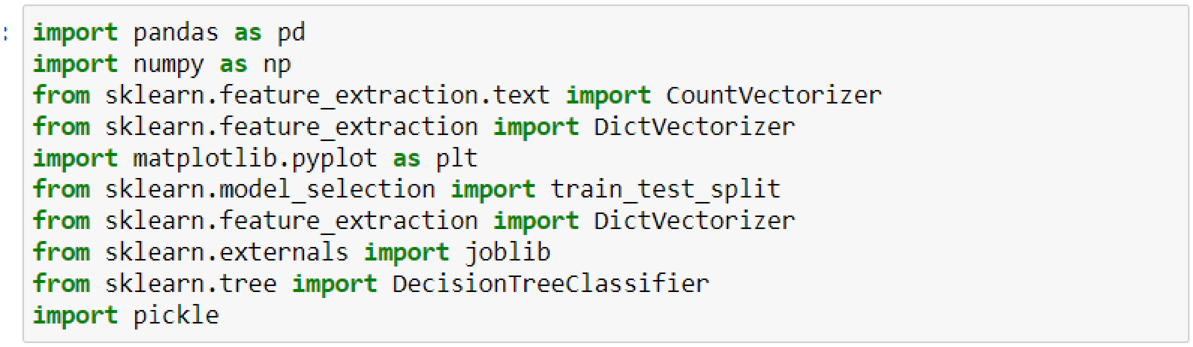
Preprocessing of the data actually involves the following steps:

4.1.1 GETTING THE DATASET:

We can get the data set from the database or we can get the data from the client.

4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.

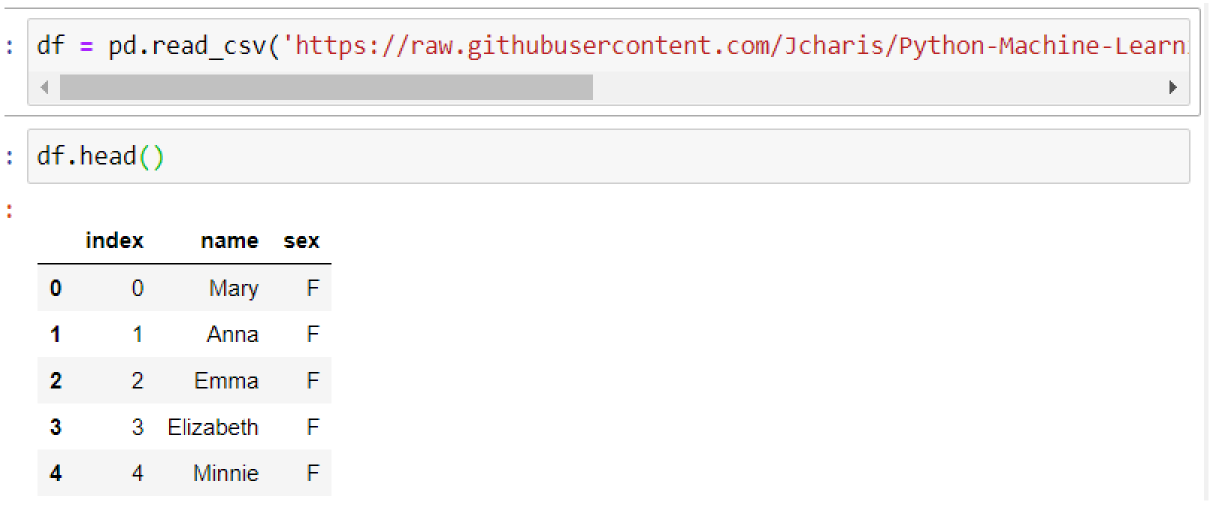


```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import DictVectorizer
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction import DictVectorizer
from sklearn.externals import joblib
from sklearn.tree import DecisionTreeClassifier
import pickle
```

Figure 8 : Importing Libraries

4.1.3 IMPORTING THE DATA-SET:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a DataFrame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the dataframe. Any missing value or NaN value have to be cleaned.



```
: df = pd.read_csv('https://raw.githubusercontent.com/Jcharis/Python-Machine-Learn
```

```
: df.head()
```

```
:
   index       name  sex
0      0       Mary    F
1      1       Anna    F
2      2       Emma    F
3      3  Elizabeth    F
4      4     Minnie    F
```

Figure 9 : Reading the dataset

4.1.4 HANDLING MISSING VALUES:

Missing values can be handled in many ways using some inbuilt methods:

1. dropna()
2. fillna()

1. dropna():

dropna() is a function which drops all the rows and columns which are having the missing values(i.e. NaN).

dropna() function has a parameter called how which works as follows:

● if how = 'all' is passed then it drops the rows where all the columns of the particular row are missing.

● if how = 'any' is passed then it drops the rows where all the columns of the particular row are missing.

2. fillna():

fillna() is a function which replaces all the missing values using different ways

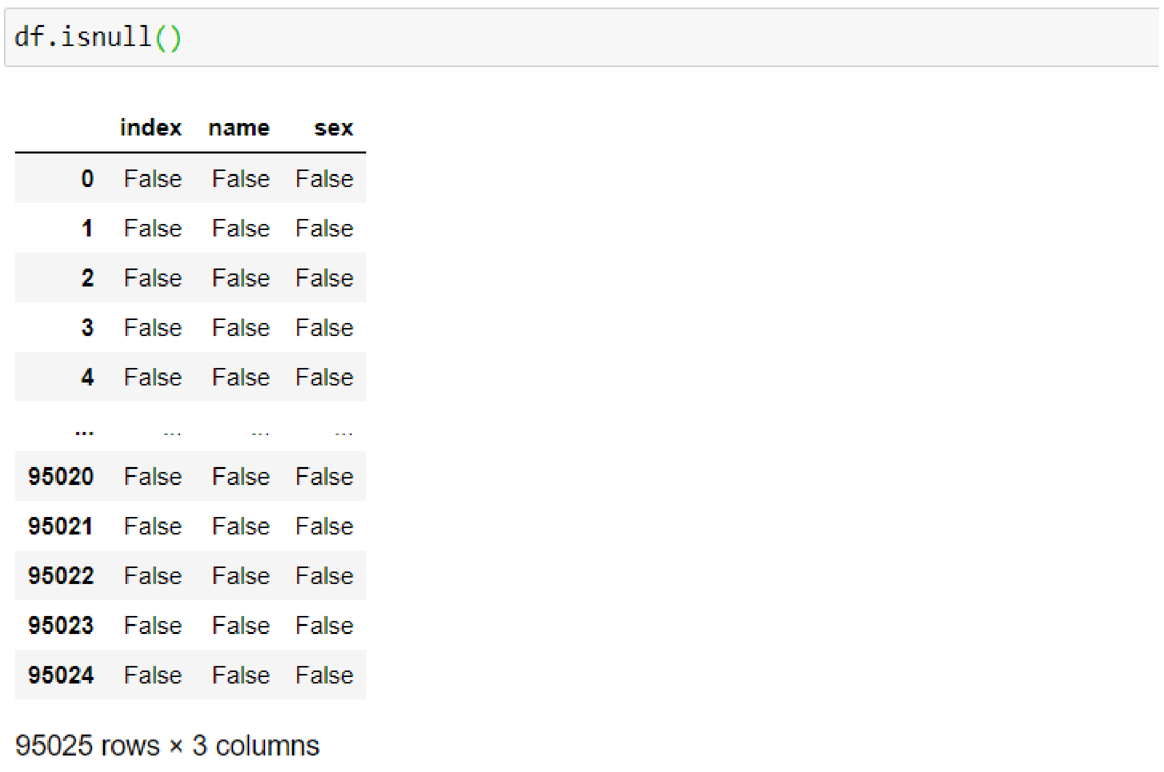
fillna() also have parameters called method and axis.

● if we use method = 'ffill' where ffill is a method called forward fill, which carry forwards the previous row's value .

● if we use method = 'bfill' where bfill is a method called backward fill, which carry backward the next row's value .

● if we use method = 'ffill' , axis = 'columns' then it carry forwards the previous column's value .

● if we use method = ‘bfill’ , axis = ‘columns’ then it carry backward the next column’s value .



```
df.isnull()
```

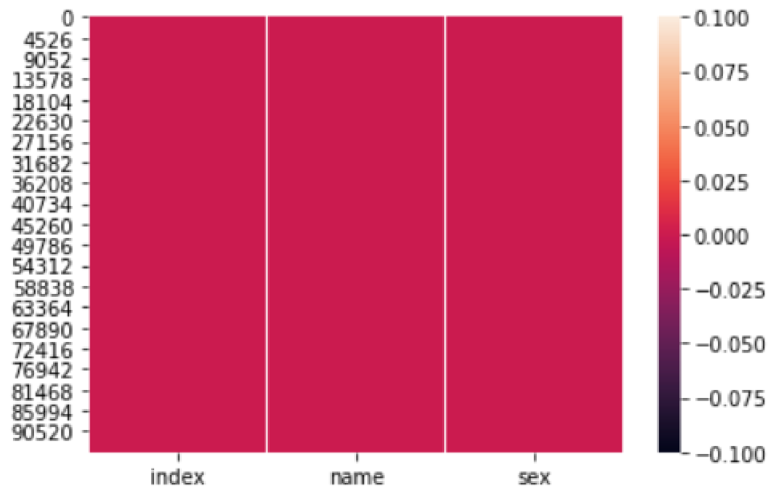
```
        index  name   sex
    0   False  False  False
    1   False  False  False
    2   False  False  False
    3   False  False  False
    4   False  False  False
  ...     ...    ...    ...
95020   False  False  False
95021   False  False  False
95022   False  False  False
95023   False  False  False
95024   False  False  False

95025 rows × 3 columns
```

Figure 10 : Checking missing values.

From the above output we can observe that the given dataset do not contain any missing values.

```
sb.heatmap(df.isna())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a9c0f70548>
```



4.1.5 CATEGORICAL DATA:

- Machine Learning models are based on equations, we need to replace the text by numbers. So that we can include the numbers in the equations.

  Categorical Variables are of two types: Nominal and Ordinal

- **Nominal:**

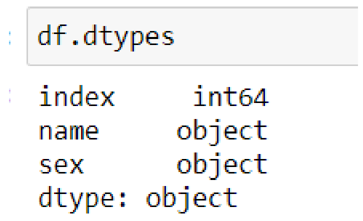
  The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any colour

- **Ordinal:**

  The categories have a numerical ordering in between them. Example: Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium

- Categorical data can be handled by using dummy variables, which are also called as indicator variables.

- Handling categorical data using dummies: In pandas library we have a method called get_dummies() which creates dummy variables for those categorical data in the form of 0’s and 1’s. Once these dummies got created we have to concat this dummy set to
- our dataframe or we can add that dummy set to the dataframe.



```
df.dtypes

index     int64
name     object
sex      object
dtype: object
```

4.2 TRAINING THE MODEL:

In Machine Learning and Data Science we often come across a term called Imbalanced Data Distribution, generally happens when observations in one of the class are much higher or lower than the other classes. As Machine Learning algorithms tend to increase accuracy by reducing the error, they do not consider the class distribution. This problem is prevalent in examples such as Gender Classifier, Fraud Detection, Anomaly Detection, Facial recognition etc.

Machine learning is a booming field in computer science. Its focus is to train algorithms to make predictions and decisions from datasets. These datasets can either be curated or generated in real time.

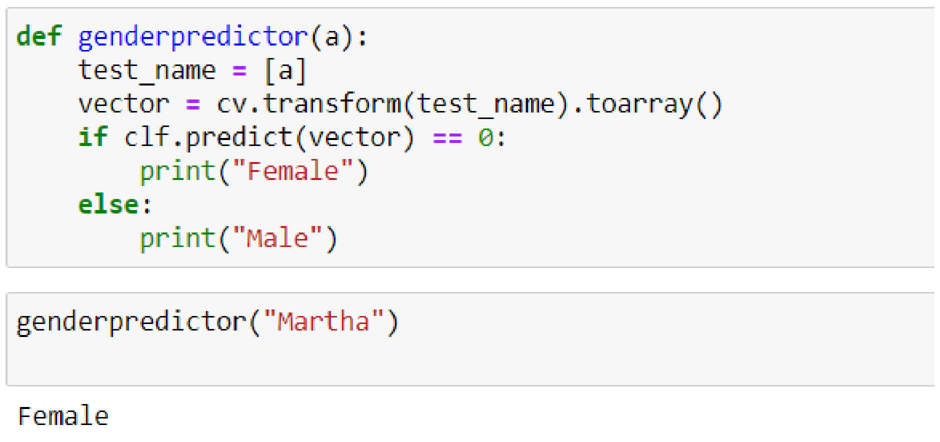
Standard ML techniques such as Decision Tree and Logistic Regression have a bias towards the majority class, and they tend to ignore the minority class. They tend only to predict the majority class, hence, having major misclassification of the minority class in comparison with the majority class. In more technical words, if we have imbalanced data distribution in our dataset then our model becomes more prone to the case when minority class has negligible or very lesser recall.

There are different algorithums that widely used to predict the gender based upon their gender:

1.DictVectorizer

2.Decision tree classifier

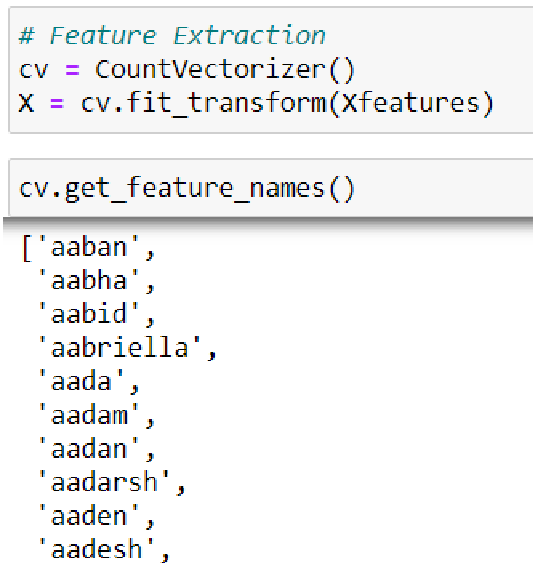
3.gender predictor



```
def genderpredictor(a):
    test_name = [a]
    vector = cv.transform(test_name).toarray()
    if clf.predict(vector) == 0:
        print("Female")
    else:
        print("Male")
```

```
genderpredictor("Martha")
```

```
Female
```



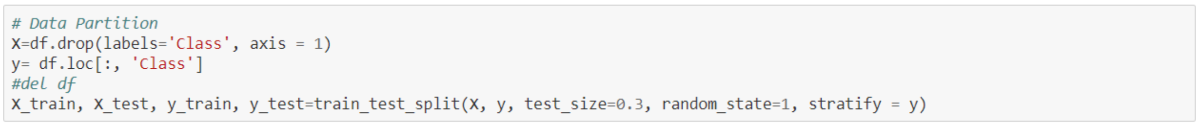
```
# Feature Extraction
cv = CountVectorizer()
X = cv.fit_transform(Xfeatures)
```

```
cv.get_feature_names()
```

```
['aaban',
 'aabha',
 'aabid',
 'aabriella',
 'aada',
 'aadam',
 'aadan',
 'aadarsh',
 'aaden',
 'aadesh',
```

4.2.1 Splitting the data : after the preprocessing is done then the data is split into train and test sets.

- In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.
- training set - a subset to train a model.(Model learns patterns between Input and Output)
- test set - a subset to test the trained model.(To test whether the model has correctly learnt)
- The amount or percentage of Splitting can be taken as specified (i.e. train data = 75% , test data =25% or train data = 80% , test data= 20%) .
- First we need to identify the input and output variables and we need to separate the input set and output set.
- In scikit learn library we have a package called model_selection in which train_test_split method is available .we need to import this method.
- This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables(we need to mention the variables) .



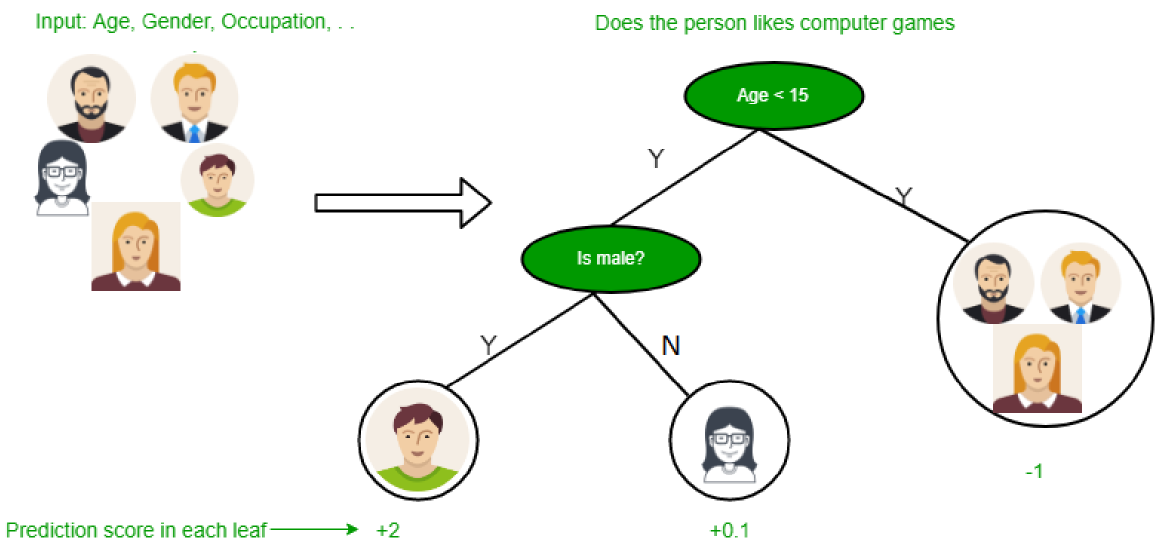
```
# Data Partition
X=df.drop(labels='Class', axis = 1)
y= df.loc[:, 'Class']
#del df
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.3, random_state=1, stratify = y)
```

Figure 12 :splitting data

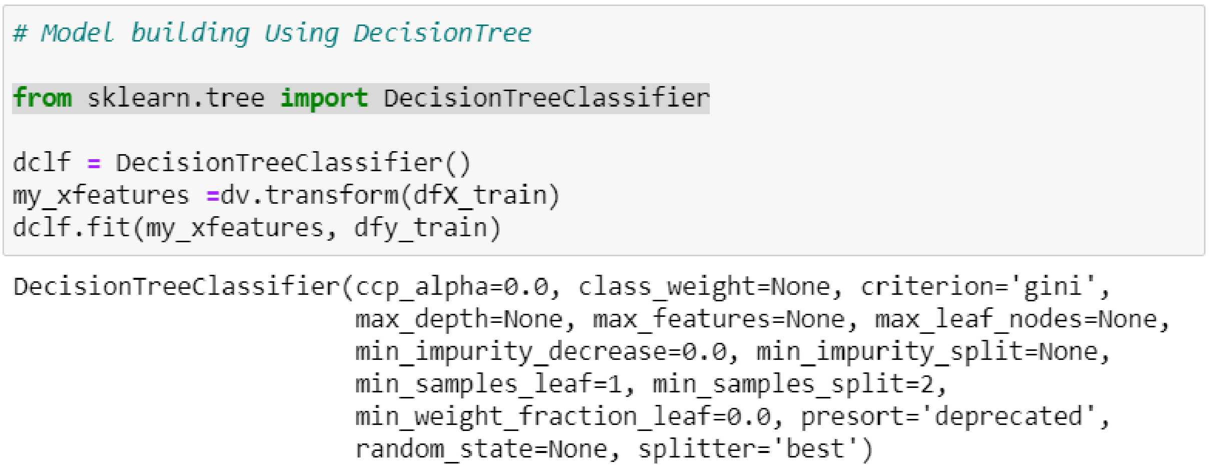
- Then we need to import logistic regression method from linear_model package from scikit learn library
- We need to train the model based on our train set (that we have obtained from splitting)
- Then we have to test the model for the test set ,that is done as follows
    - We have a method called predict , using this method we need to predict the output for the input test set and we need to compare the output with the output test data.
    - If the predicted values and the original values are close then we can say that model is trained with good accuracy.

4.3 Model Building and Evaluation

4.3.1 Decision tree classifier:


Input: Age, Gender, Occupation, . .
Does the person likes computer games
Age < 15
Y
Y
Is male?
Y
N
-1
Prediction score in each leaf
+2
+0.1

Decision tree classifier is used to classifier gender based upon their name.

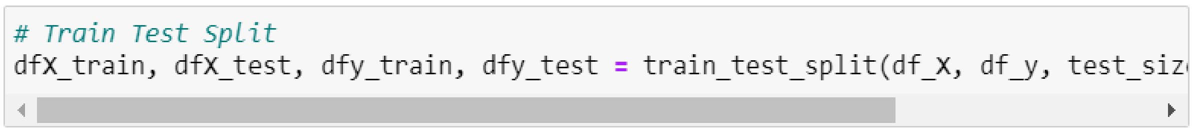


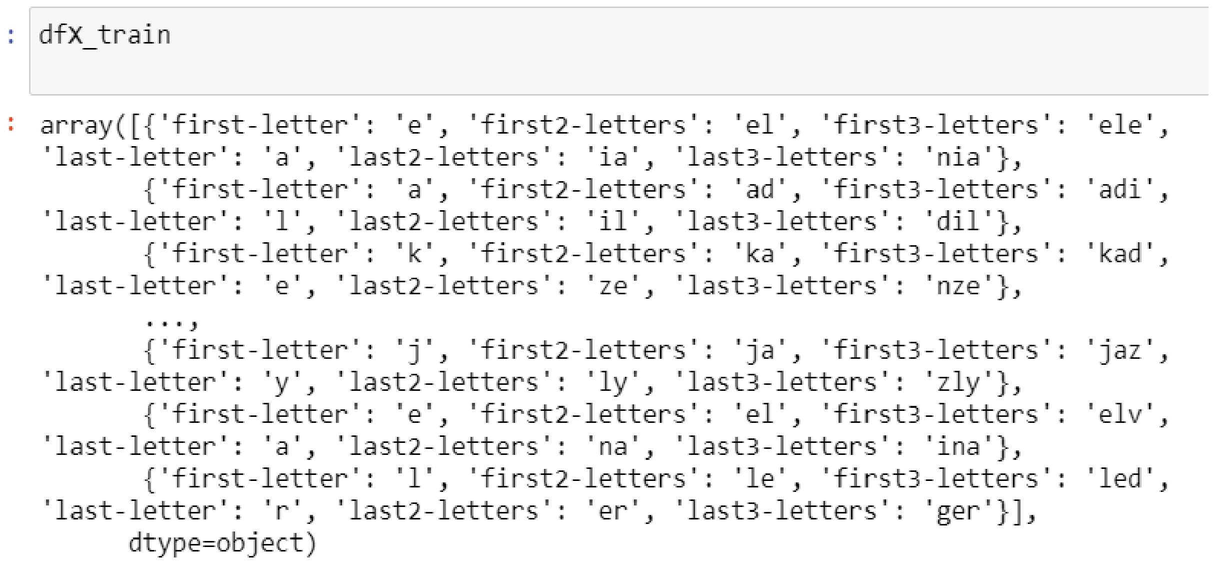
```
# Model building Using DecisionTree

from sklearn.tree import DecisionTreeClassifier

dclf = DecisionTreeClassifier()
my_xfeatures =dv.transform(dfX_train)
dclf.fit(my_xfeatures, dfy_train)
```

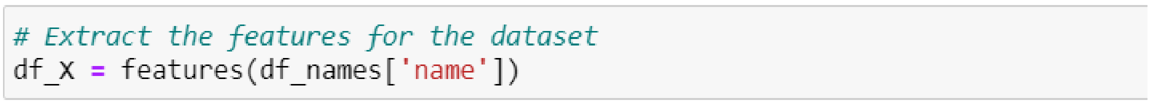
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

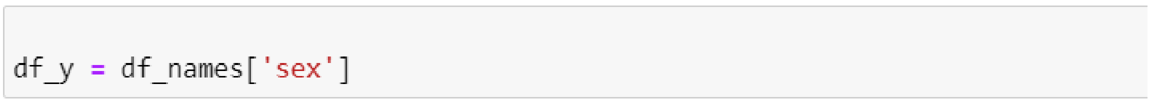
Instead of directly predicting on test data, let us see how well the model predicts the training data.

<u>Predicting on training data</u>

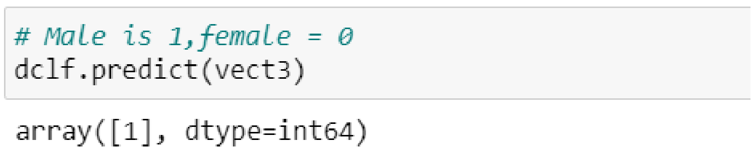

```
# Train Test Split
dfX_train, dfX_test, dfy_train, dfy_test = train_test_split(df_X, df_y, test_siz
```

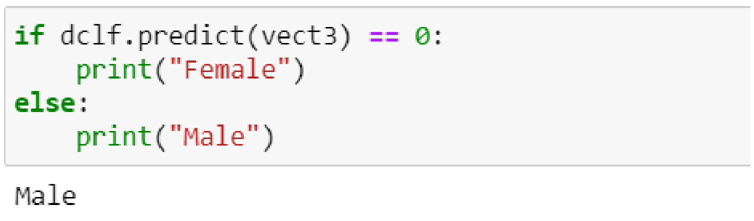

```
: dfX_train
```
```
: array([{'first-letter': 'e', 'first2-letters': 'el', 'first3-letters': 'ele',
  'last-letter': 'a', 'last2-letters': 'ia', 'last3-letters': 'nia'},
         {'first-letter': 'a', 'first2-letters': 'ad', 'first3-letters': 'adi',
  'last-letter': 'l', 'last2-letters': 'il', 'last3-letters': 'dil'},
         {'first-letter': 'k', 'first2-letters': 'ka', 'first3-letters': 'kad',
  'last-letter': 'e', 'last2-letters': 'ze', 'last3-letters': 'nze'},
         ...,
         {'first-letter': 'j', 'first2-letters': 'ja', 'first3-letters': 'jaz',
  'last-letter': 'y', 'last2-letters': 'ly', 'last3-letters': 'zly'},
         {'first-letter': 'e', 'first2-letters': 'el', 'first3-letters': 'elv',
  'last-letter': 'a', 'last2-letters': 'na', 'last3-letters': 'ina'},
         {'first-letter': 'l', 'first2-letters': 'le', 'first3-letters': 'led',
  'last-letter': 'r', 'last2-letters': 'er', 'last3-letters': 'ger'}],
        dtype=object)
```

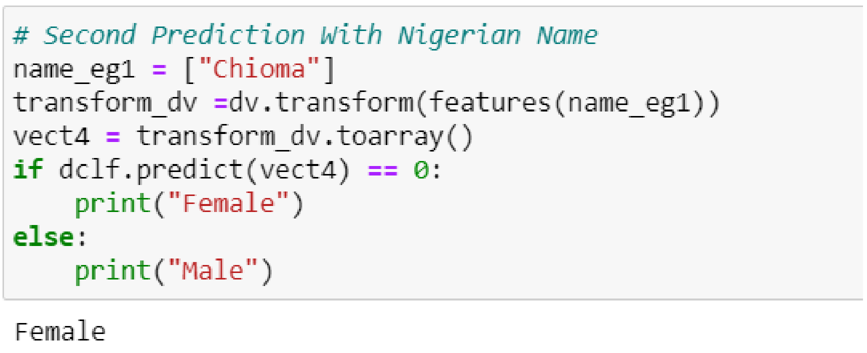

```
# Extract the features for the dataset
df_X = features(df_names['name'])
```


```
df_y = df_names['sex']
```

Predicting on test data

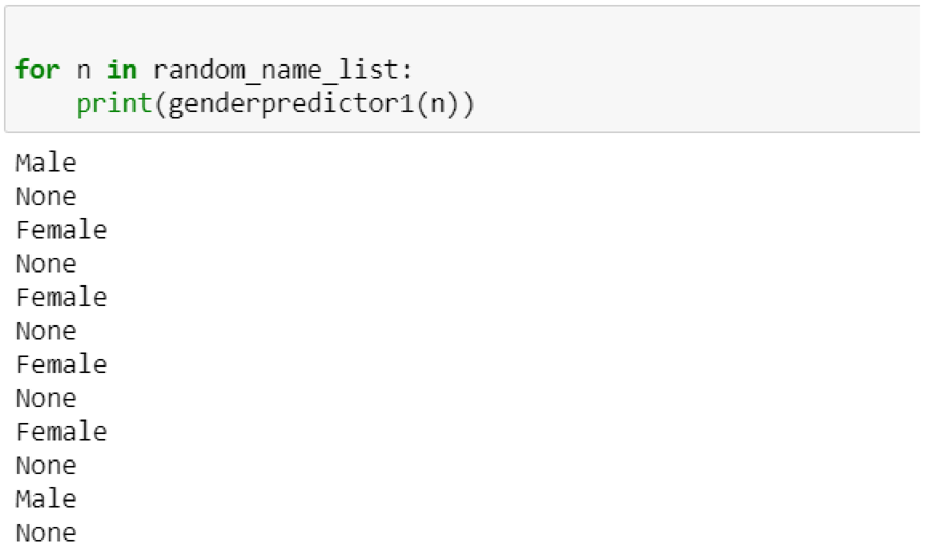

```
# Male is 1,female = 0
dclf.predict(vect3)
```
```
array([1], dtype=int64)
```


```
if dclf.predict(vect3) == 0:
    print("Female")
else:
    print("Male")
```
```
Male
```



```
# Second Prediction With Nigerian Name
name_eg1 = ["Chioma"]
transform_dv =dv.transform(features(name_eg1))
vect4 = transform_dv.toarray()
if dclf.predict(vect4) == 0:
    print("Female")
else:
    print("Male")
```

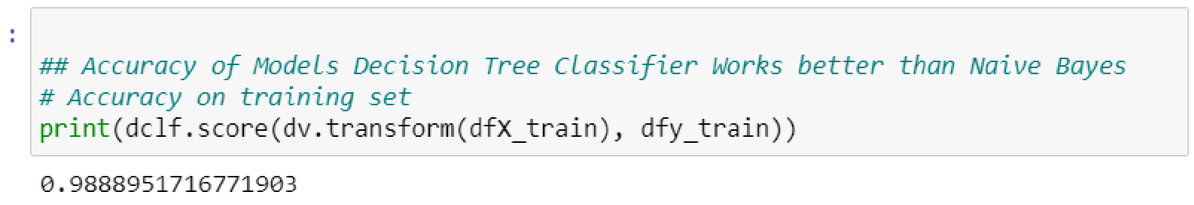
```
Female
```



```
for n in random_name_list:
    print(genderpredictor1(n))
```

```
Male
None
Female
None
Female
None
Female
None
Female
None
Male
None
```

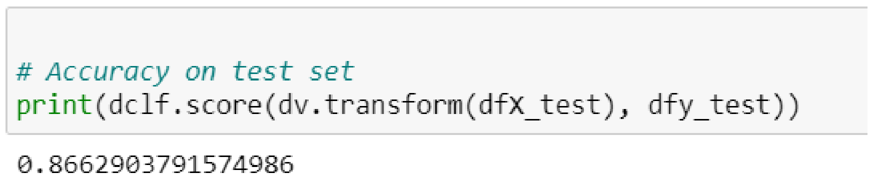
Accuracy of the Decision tree classifier:



```
:
## Accuracy of Models Decision Tree Classifier Works better than Naive Bayes
# Accuracy on training set
print(dclf.score(dv.transform(dfX_train), dfy_train))
```

```
0.9888951716771903
```

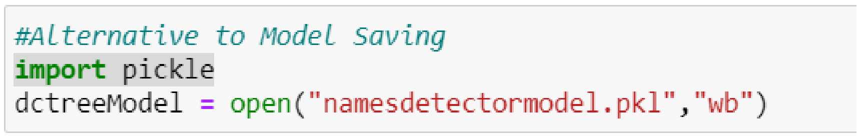
Accuracy on test set:



```
# Accuracy on test set
print(dclf.score(dv.transform(dfX_test), dfy_test))
```

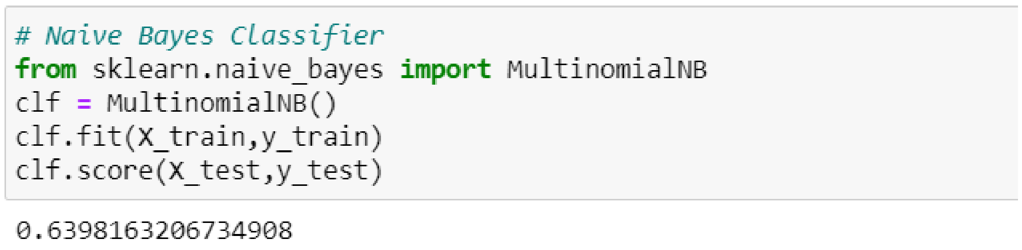
```
0.8662903791574986
```

Alternative to the model:



```
#Alternative to Model Saving
import pickle
dctreeModel = open("namesdetectormodel.pkl","wb")
```

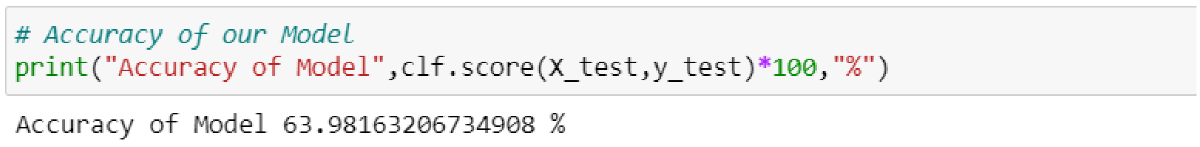
Naïve Bayes Classifier:



```
# Naive Bayes Classifier
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train,y_train)
clf.score(X_test,y_test)
```

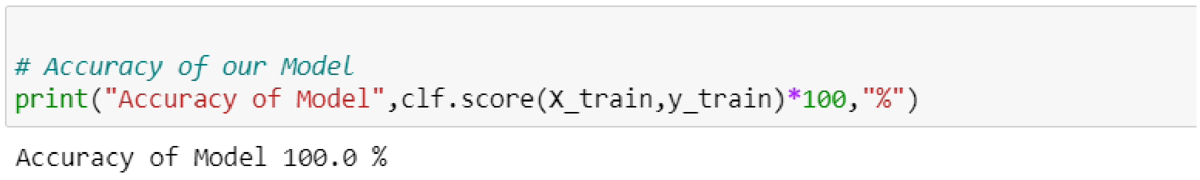
```
0.6398163206734908
```

Accuracy of the model:



```
# Accuracy of our Model
print("Accuracy of Model",clf.score(X_test,y_test)*100,"%")
```

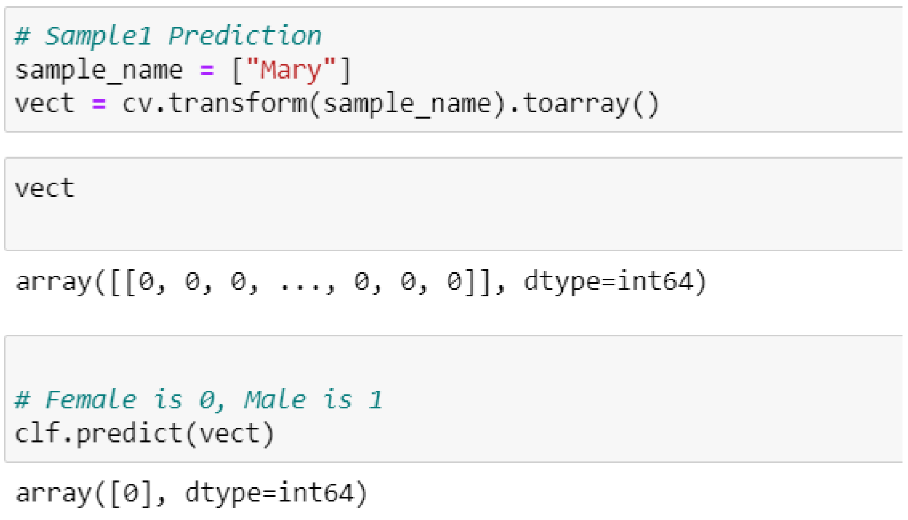
```
Accuracy of Model 63.98163206734908 %
```



```
# Accuracy of our Model
print("Accuracy of Model",clf.score(X_train,y_train)*100,"%")
```

```
Accuracy of Model 100.0 %
```

Sample Predictions:



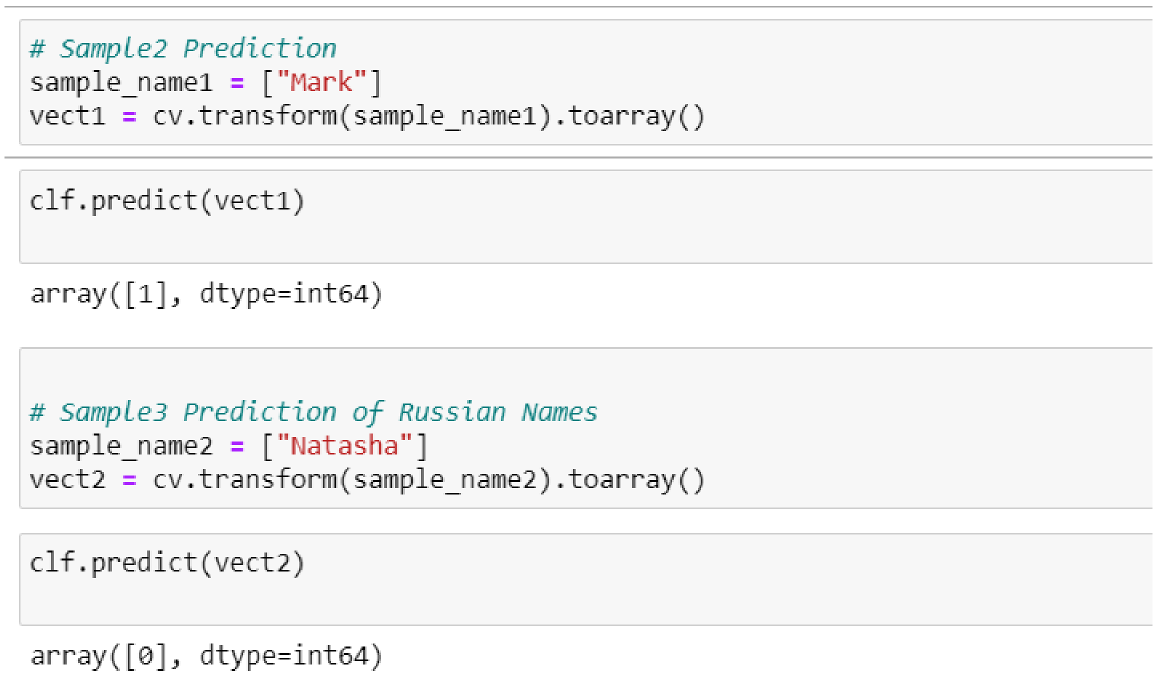
```python
# Sample1 Prediction
sample_name = ["Mary"]
vect = cv.transform(sample_name).toarray()
```

```python
vect
```

```
array([[0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```python
# Female is 0, Male is 1
clf.predict(vect)
```

```
array([0], dtype=int64)
```



```python
# Sample2 Prediction
sample_name1 = ["Mark"]
vect1 = cv.transform(sample_name1).toarray()
```

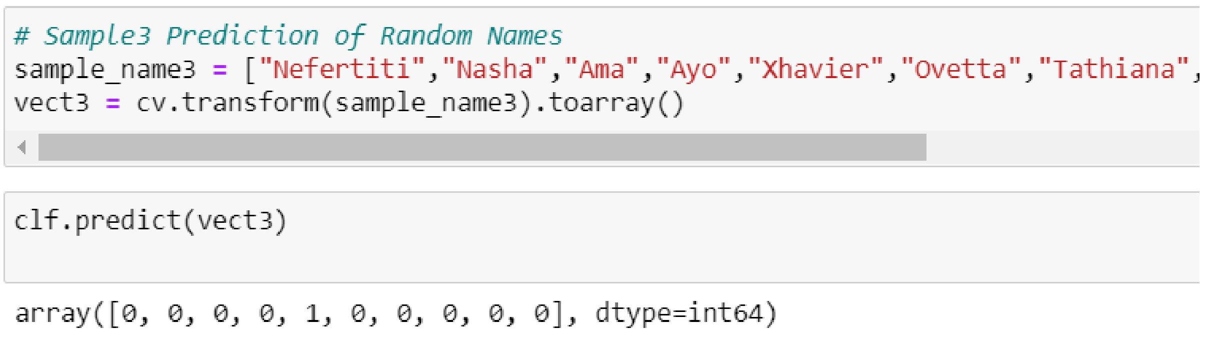
```python
clf.predict(vect1)
```

```
array([1], dtype=int64)
```

```python
# Sample3 Prediction of Russian Names
sample_name2 = ["Natasha"]
vect2 = cv.transform(sample_name2).toarray()
```

```python
clf.predict(vect2)
```

```
array([0], dtype=int64)
```



```python
# Sample3 Prediction of Random Names
sample_name3 = ["Nefertiti","Nasha","Ama","Ayo","Xhavier","Ovetta","Tathiana",
vect3 = cv.transform(sample_name3).toarray()
```

```python
clf.predict(vect3)
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

Using a custom function for feature analysis:

```
: # By Analogy most female names ends in 'A' or 'E' or has the sound of 'A'
def features(name):
    name = name.lower()
    return {
        'first-letter': name[0], # First letter
        'first2-letters': name[0:2], # First 2 letters
        'first3-letters': name[0:3], # First 3 letters
        'last-letter': name[-1],
        'last2-letters': name[-2:],
        'last3-letters': name[-3:],
    }
```

```
: # Vectorize the features function
features = np.vectorize(features)
print(features(["Anna", "Hannah", "Peter","John","Vladmir","Mohammed"]))
```

```
[{'first-letter': 'a', 'first2-letters': 'an', 'first3-letters': 'ann', 'last-l
etter': 'a', 'last2-letters': 'na', 'last3-letters': 'nna'}
 {'first-letter': 'h', 'first2-letters': 'ha', 'first3-letters': 'han', 'last-l
etter': 'h', 'last2-letters': 'ah', 'last3-letters': 'nah'}
 {'first-letter': 'p', 'first2-letters': 'pe', 'first3-letters': 'pet', 'last-l
etter': 'r', 'last2-letters': 'er', 'last3-letters': 'ter'}
 {'first-letter': 'j', 'first2-letters': 'jo', 'first3-letters': 'joh', 'last-l
etter': 'n', 'last2-letters': 'hn', 'last3-letters': 'ohn'}
 {'first-letter': 'v', 'first2-letters': 'vl', 'first3-letters': 'vla', 'last-l
etter': 'r', 'last2-letters': 'ir', 'last3-letters': 'mir'}
 {'first-letter': 'm', 'first2-letters': 'mo', 'first3-letters': 'moh', 'last-l
etter': 'd', 'last2-letters': 'ed', 'last3-letters': 'med'}]
```

5.Conclusion

From the above model building and evaluation we can predict that Decision tree classifier is best for predicting the name based on gender.

6.References:

https://raw.githubusercontent.com/Jcharis/Python-Machine-Learning/master/Gender%20Classification%20With%20%20Machine%20Learning/names_dataset.csv