**Gender Prediction Of Bangladeshi Names**

**ABSTRACT:** Bengali Names have rich connotation and have strong distinction of gender. In this paper, English names` dataset and Bengali names` dataset have been analyzed using different classifiers & features to detect gender and compared if they work the same way for both types of names.

**Keywords:** Gender prediction, classifier, machine learning algorithm,

**Chapter I: Introduction**

**Introduction:** Classifying gender for many purposes is necessary & can be a lot more convenient for many purposes. Knowing how many female or male are participating in a program, for labs and classrooms if they need different set up or arrangement, having the information can be a plus point for making arrangements. For different websites or online services, knowing which gender is the user can be helpful to suggest items for them. Also, to target the right audience or analyze the most users, identifying gender can be advantageous. Implementing the identifier on chatbots or virtual assistants will help choosing the preferred voice.

In the field of machine learning, identifying genders using different ways are at issue now. Research on identifying genders through images,tweets,likes or other traits using image processing , pattern recognition, behavior analysis and in a lot more ways already exist. Research on gender detection using names have been done before, such as for English (US), Chinese, Indian, Indonesian names using different classifiers. In this research English names have been analyzed using a English names` dataset available online which contains 00000 data and gives accuracy of 89% .Next , Bangladeshi names have been analysed the same way and with adding different classifiers to compare if they work the same way and also to improve the accuracy of detecting or identifying gender with Bangladeshi names.

# **Gender Prediction Models:**

# Bangladesh is a developing country.It is supposed to be fully digital in a few years.So, in its current vigorous/high spirited time? To be able to predict gender and implementing this can be a great step /achievement towards being digital.Different research has been done regarding predicting gender.In a paper named **Research on Gender Recognition of Names Based on Machine Learning Algorithm** support vector classifier , maximum entropy, Naive Bayes classifiers have been used.In another paper named **Gender Prediction of Indian Names** Support Vector Machinebased approach has been used.IN the paper **Predicting the Gender of Indonesian Names** character-level Long-Short Term Memory (char-LSTM) has been used to predict gender.

So, to summarize, research on names exists using char-LSTM, SVM,

maximum entropy, artificial neural network but on different languges.In this research ,we have analyzed both English(US) names and Bangladeshi names, compared their results to see how the classifiers & features work for both the type of names.

# **Aim of Study:**

# This research aims to analyse classifiers,compare & improve performance using different features.For this, Naive Bayes,Logistic regression,K-Nearest Neighbors,Support Vector Machines,Decision Tree,Linear Discriminant Analysis have been applied as classifiers to train models and improve performance on test phase.

# **Purpose of Study:**

**Research Objective:**

Accurate prediction of an unknown individual’s gender is desirable for use in marketing, social science, and many other applications in academia and industry. Perhaps the most obvious and telling indicator of a person’s gender is their first name. Most previous work in classifying gender via first name has concerned using a large corpus of known names to give a probabilistic prediction on the names that are known. This research attempts to explore the space of names that are *unknown* by examining the facets of a first name — specifically focusing on the sequences of characters within the name - that contain non-trivial gender-revealing information.

This gender classification problem is similar to but fundamentally different from a larger class of problem in Natural Language Understanding.

Our gender classification problem becomes even more interesting when you abstract it from names to all words in general. Linguistically, many languages are structured with the concept of a *grammatical gender* wherein classes of nouns in the language are formally associated with one of a discrete set of genders. In such languages, certain sequences of characters in a word can almost surely identify the grammatical gender of the word. Learning these character sequences — whether explicitly or implicitly — is an inherent part of learning the language. Furthermore, understanding of such character patterns may assist in understanding of unseen

words. For these reasons, studying information embedded in character sequences seems like an interesting and integral topic to Linguistics and NLU that is beyond the scope of this research.

**Research Questions:**

1. Is it possible to predict gender accurately only through names

2. How an effective model will be built?

3. How accurately can the model predict?

4.

**Chapter 2: Background Study**

**2.1 Introduction**

**2.2 Gender Prediction**

Consider the names “Fahim” and “Promi” — most people would instantly mark Fahim as a male name and Promi as a female one. Is this the case *primarily* because we have seen so many examples of male Fahim and female Promi that our brains have built up a latent association between the specific name and the corresponding gender? Probably.

But some component of the name itself (its spelling / combination of letters) contributes to the gender with which it is associated to a large degree as well. Consider the names “Samin” and “Sameen.” They are phonetically identical , however most people would categorize “Samin” as male and “Sameen” as female upon seeing the spellings. The suffix of a name can indicate the name’s gender; however, the rules are not cut and dry. For example, names “ending in *-a or -e* appear to be predominantly female, despite the fact that names has the sound of -A tends to be female; and names ending in *-ch* are usually male, even though names that end in *-h* tend to be female”. There are many more character patterns that correspond to a certain gender classification than just the suffix — this task is not trivial.

The gender classification of a name becomes increasingly difficult when you consider the space of all names from around the world — the examples I have given thus far are admittedly from a standard Bangladesh viewpoint. Let’s now consider two Indian names, when I see the names “Priyanka” and “Srikanth,” That instantly assume Priyanka to be female and Srikanth to be male. Why is that? How do our brains extract the gender revealing information encoded in the sequence of characters that compose a name?

**2.3 Data Analysis**

Data analysis is a process of inspecting, cleansing, transforming and modeling data with the goal of discovering useful information, in forming conclusions and supporting decision-making.The purpose of Data Analysis is to extract useful information from data and taking the decision based upon the data analysis. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains. In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively[1].

**2.3.1 Data Mining**

Data mining is defined as a process used to extract usable data from a larger set of any raw data. It implies analyzing data patterns in large batches of data using one or more software. Data mining has applications in multiple fields, like science and research. Its focus is to train algorithms to make predictions and decisions from datasets. These datasets can either be curated or generated in real time.A classifier has been built to guess the gender of a name using characteristics of the name.Accurate prediction of an unknown individual’s gender is desirable for use in marketing, social science, and many other applications in academia and industry. Perhaps the most obvious and telling indicator of a person’s gender is their first name.

.In a nutshell, the flow to building this model can be visualized as:

A name =>Classifier =>Male/Female

In this research,Python is the tool which is going to play the role of data mining and dig some information for us.We implemented several machine learning algorithms in Python using **Scikit-learn**, the most popular machine learning tool for Python.Python library is a collection of functions and methods that allows to perform lots of actions without writing own code.

In our research we worked with **numpy, scipy, pandas,bs4** etc.

It is also to be mentioned that there are many algorithms available and the best suitable one for collected dataset needs to be discovered after examining them.Scikit-Learn provides easy access to numerous different classification algorithms. Among these classifiers are:

* K-Nearest Neighbors
* Support Vector Machines
* Decision Tree Classifiers/Random Forests
* Naive Bayes
* Linear Discriminant Analysis
* Logistic Regression
* Multinomial NB

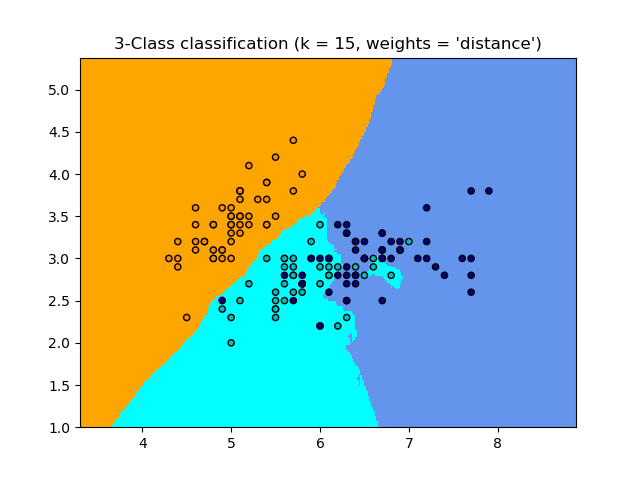
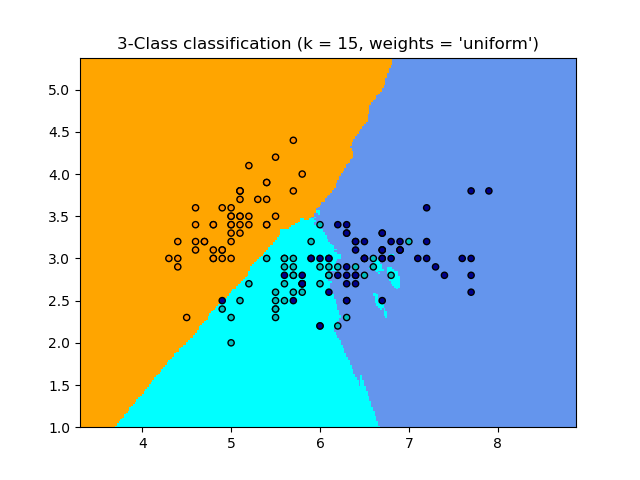
**2.3.1.1 Nearest Neighbors**

sklearn.neighbors provides functionality for unsupervised and supervised neighbors-based learning methods.

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice.

Euclidean Distance = sqrt(sum i to N (x1\_i – x2\_i)^2)

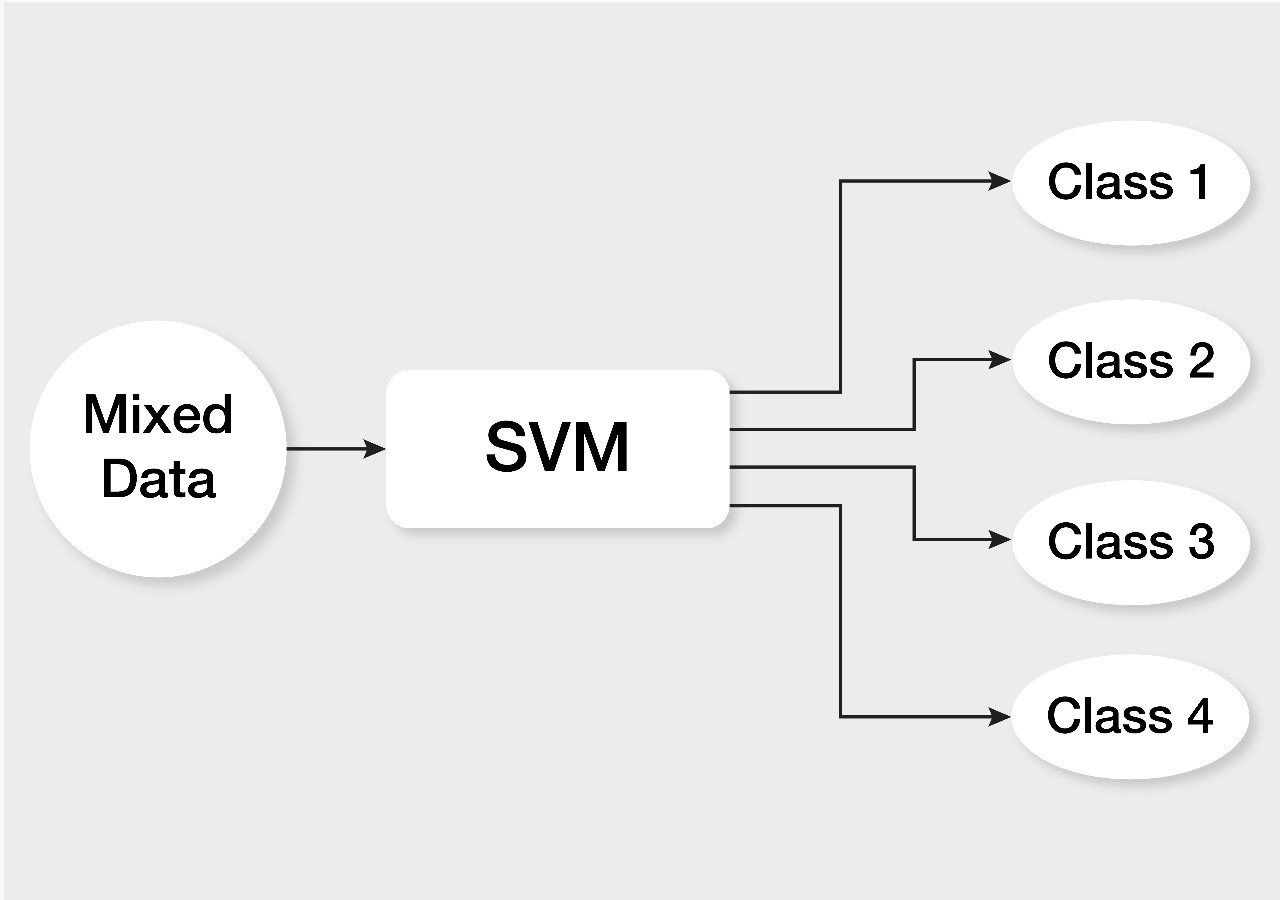
The -neighbors classification in **KNeighborsClassifier** is the most commonly used technique. The optimal choice of the value is highly data-dependent: in general a larger suppresses the effects of noise, but makes the classification boundaries less distinct.Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

The basic nearest neighbor classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns uniform weights to each neighbor. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied to compute the weights. ****

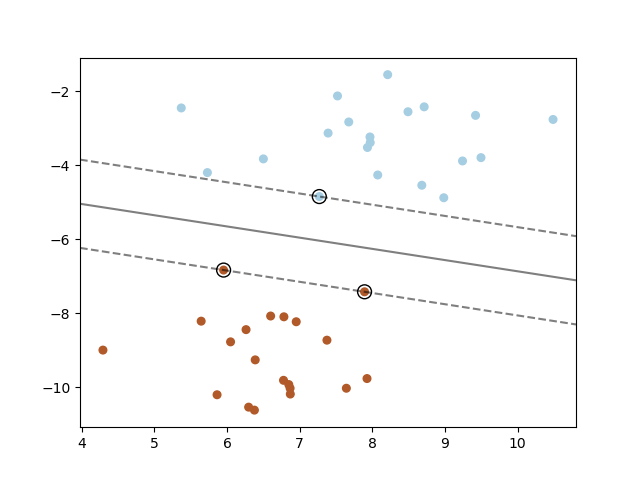
Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular.

**2.3.1.2 Support Vector Machine**

Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset.



SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.



The general meaning of this hyperplane is:

WtX + b = 0

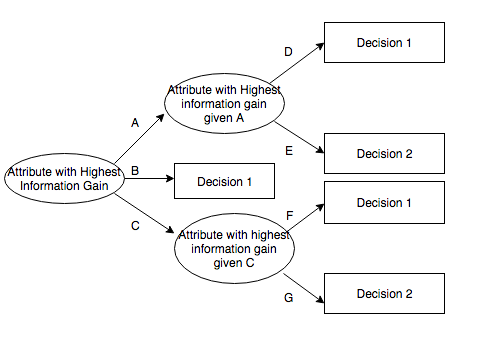
The advantages of support vector machines are:

* Effective in high dimensional spaces.
* Still effective in cases where the number of dimensions is greater than the number of samples.
* Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
* Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

**2.3.1.3 Decision tree**

A decision tree is one of the most frequently and widely used supervised machine learning algorithms that can perform both regression and classification tasks. The intuition behind the decision tree algorithm is simple, yet also very powerful.

For each attribute in the dataset, the decision tree algorithm forms a node, where the most important attribute is placed at the root node. For evaluation we start at the root node and work our way down the tree by following the corresponding node that meets our condition or "decision". This process continues until a leaf node is reached, which contains the prediction or the outcome of the decision tree.Once the network has divided the data down to one example, the example will be put into a class that corresponds to a key. When multiple random forest classifiers are linked together they are called *Random Forest Classifiers*.

[](https://commons.wikimedia.org/wiki/File:ID3_algorithm_decision_tree.png)

**Steps for Making decision tree**

* Get list of rows (dataset) which are taken into consideration for making decision tree (recursively at each node).
* Calculate uncertainty of our dataset or Gini impurity or how much our data is mixed up etc.
* Generate list of all questions which needs to be asked at that node.
* Partition rows into True rows and False rows based on each question asked.
* Calculate information gain based on gini impurity and partition of data from previous step.
* Update highest information gain based on each question asked.
* Update best question based on information gain (higher information gain).
* Divide the node on best question. Repeat again from step 1 again until we get pure node (leaf nodes).

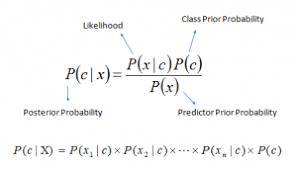
Overfitting is one of the major problems for every model in machine learning. If model is overfitted it will poorly generalized to new samples. To avoid decision tree from overfitting we remove the branches that make use of features having low importance. This method is called as Pruning or post-pruning.This way we will reduce the complexity of tree, and hence improves predictive accuracy by the reduction of overfitting.

There are several advantages of using decision trees for predictive analysis:

1. Decision trees can be used to predict both continuous and discrete values i.e. they work well for both regression and classification tasks.
2. They require relatively less effort for training the algorithm.
3. They can be used to classify non-linearly separable data.
4. They're very fast and efficient compared to KNN and other classification algorithms.

**2.3.1.4 Naive Bayes**

Naive Bayes classifier belongs to the generative model in which how to choose the generation model and the discriminant model, mainly depends on whether or not the joint distribution is required. If the conditional independence hypothesis (a strict condition) is injected, the convergence rate of the simple Bias classifier will be faster than the discriminant model, such as logical regression.so you only need less training data and are not sensitive to the missing data. Even if the assumption of NB conditional independence does not hold up, NB classifier still performs well in practice. Its main drawback is that it cannot learn the interaction between features. The formula is as follows:



p (class | feature)=P(feature | class)p(class) / p(feature)

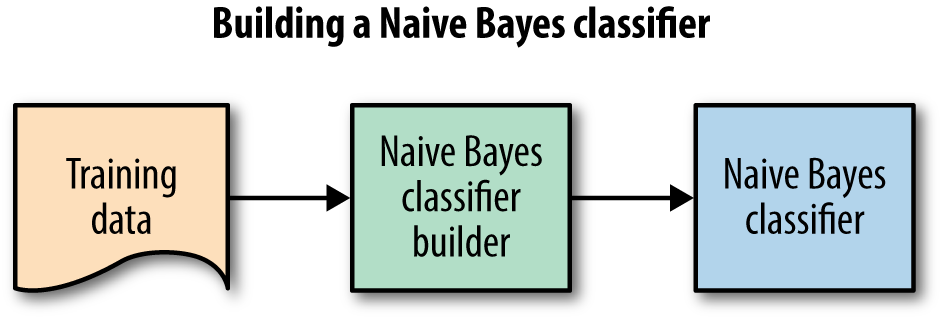
* P(class | feature) is the posterior probability of class (target) given predictor (attribute).
* P(class) is the prior probability of class.
* P(feature | class) is the likelihood which is the probability of predictor given class.
* P(feature) is the prior probability of predictor.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(feature | class).

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

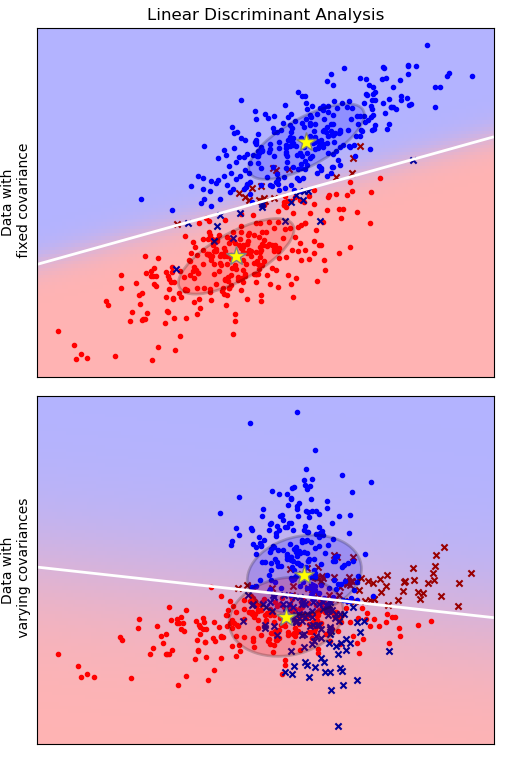
On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.



**2.3.1.5 Linear Discriminant Analysis**

Linear Discriminant Analysis works by reducing the dimensionality of the dataset, projecting all of the data points onto a line. Then it combines these points into classes based on their distance from a chosen point or centroid.

Linear discriminant analysis, as you may be able to guess, is a linear classification algorithm and best used when the data has a linear relationship.These classifiers are attractive because they have closed-form solutions that can be easily computed, are inherently multiclass, have proven to work well in practice, and have no hyperparameters to tune.

****

The plot shows decision boundaries for Linear Discriminant Analysis. The bottom row demonstrates that Linear Discriminant Analysis can only learn linear boundaries.

## **2.3.1.5.1 LDA Models**

LDA makes some simplifying assumptions about your data:

1. That your data is Gaussian, that each variable is is shaped like a bell curve when plotted.
2. That each attribute has the same variance, that values of each variable vary around the mean by the same amount on average.

With these assumptions, the LDA model estimates the mean and variance from your data for each class. It is easy to think about this in the univariate (single input variable) case with two classes.

The mean (mu) value of each input (x) for each class (k) can be estimated in the normal way by dividing the sum of values by the total number of values.

muk = 1/nk \* sum(x)

Where muk is the mean value of x for the class k, nk is the number of instances with class k. The variance is calculated across all classes as the average squared difference of each value from the mean.

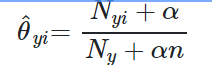
sigma^2 = 1 / (n-K) \* sum((x – mu)^2)

Where sigma^2 is the variance across all inputs (x), n is the number of instances, K is the number of classes and mu is the mean for input x.

**2.3.1.6 Multinomial NB**

MultinomialNB implements the naive Bayes algorithm for multinomial distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors  
for each class , where is the number of features (in text classification, the size of the vocabulary) and is the probability of feature appearing in a sample belonging to class y.

The parameters is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:



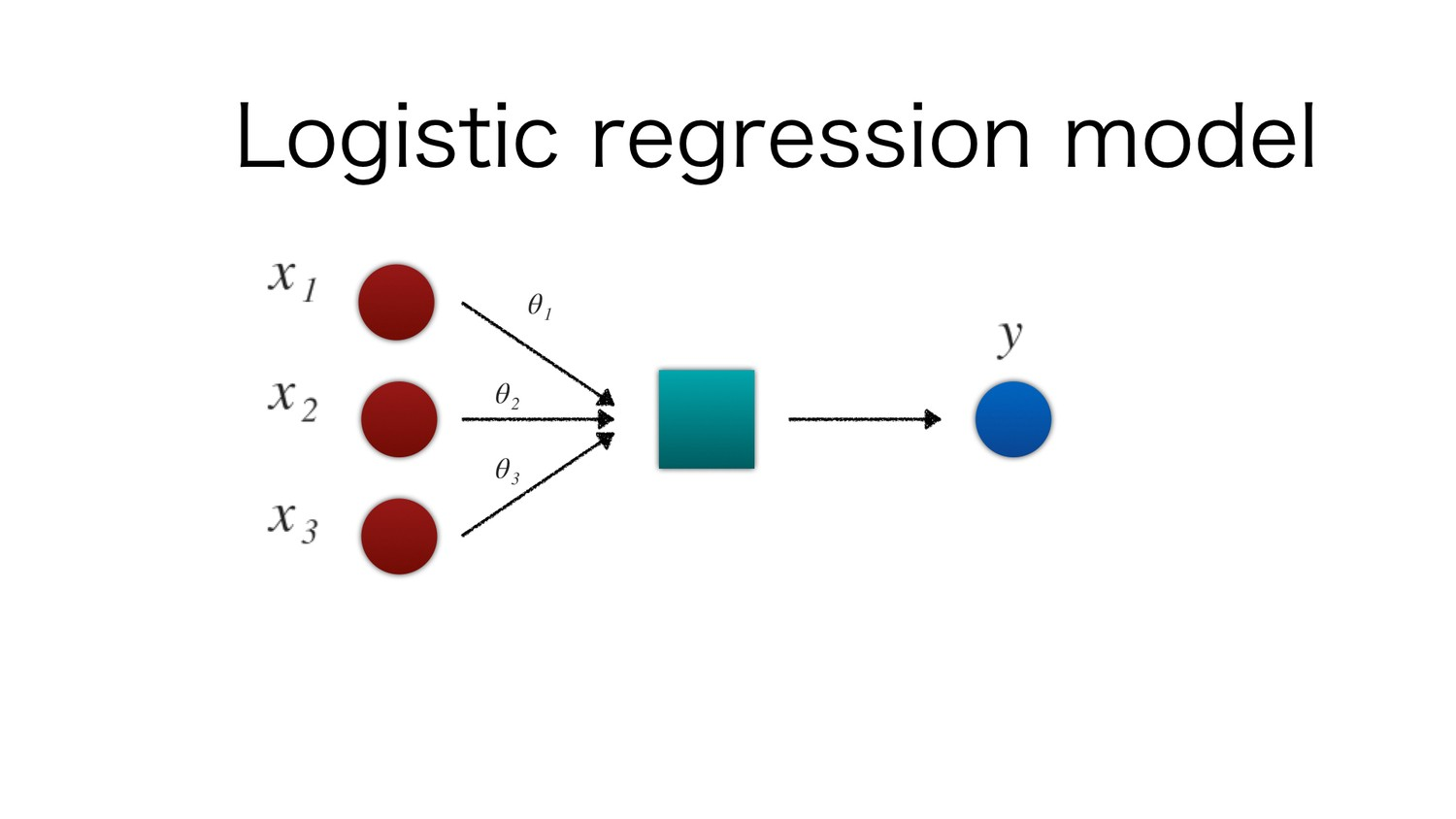
where is the number of times feature appears in a sample of class y in the training set T, and  is the total count of all features for class y.

The smoothing priors accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting is called Laplace smoothing, while is called Lidstone smoothing.

**2.3.1.7 Logistic Regression**

Logistic regression is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting the data to a logit function.This binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the probability of a given outcome by a specific percentage.

Like all regression analyses, logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.



The logistic function is given by:

f(x) = L/(1+e ^-k(x-x0))

where

L – Curve’s maximum value

k – Steepness of the curve

x0 – x value of Sigmoid’s midpoint

A standard logistic function is called sigmoid function (k=1,x0=0,L=1)

S(x) = 1/1+ ( e ^ - x)

When using machine learning, there are many ways to go wrong. Some of the most common issues in machine learning are overfitting and underfitting.

**Overfitting**

A model suffers from **Overfitting** when it has learned too much from the training data, and does not perform well in practice as a result. This is usually caused by the model having too much exposure to the training data. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

**UnderFitting**

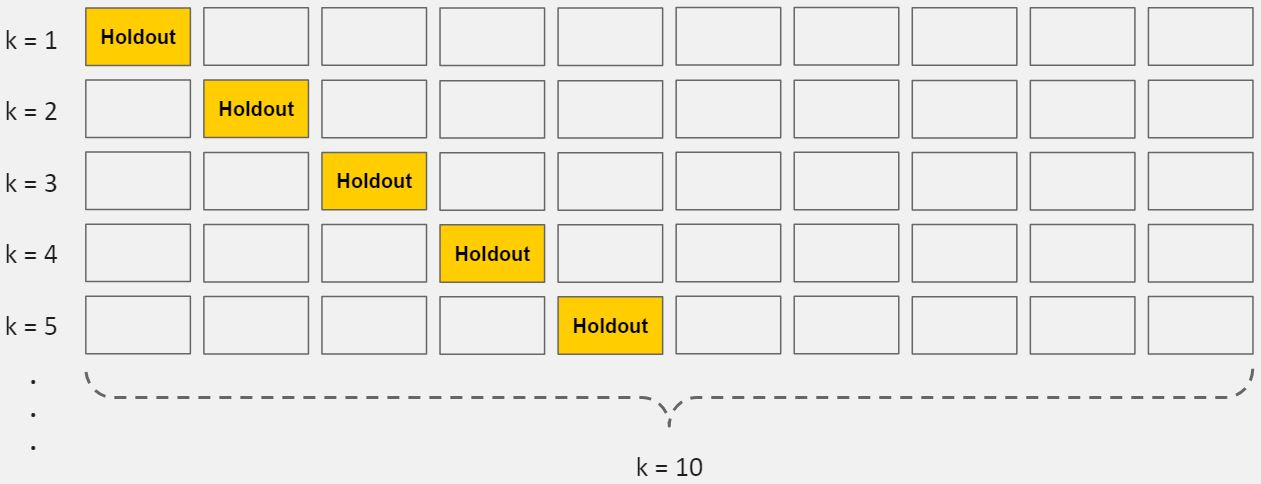
A model suffers from **Underfitting** when it has not learned enough from the training data, and does not perform well in practice as a result. As a direct contrast to the previous idea, this issue is caused by not letting the model learn enough from training data.

The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting.

**Cross-validation**

Cross-validation is a powerful preventative measure against overfitting.The most popular resampling technique is k-fold cross validation. It allows you to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data.

A validation dataset is simply a subset of your training data that you hold back from your machine learning algorithms until the very end of your project. After you have selected and tuned your machine learning algorithms on your training dataset you can evaluate the learned models on the validation dataset to get a final objective idea of how the models might perform on unseen data.

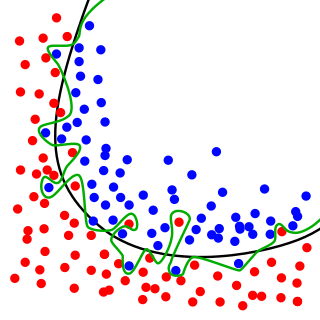


Using cross validation is a gold standard in applied machine learning for estimating model accuracy on unseen data. If you have the data, using a validation dataset is also an excellent practice

## **Goodness of Fit**

In statistics, *goodness of fit* refers to how closely a model’s predicted values match the observed (true) values.

A model that has learned the noise instead of the signal is considered “overfit” because it fits the training dataset but has poor fit with new datasets.



*While the black line fits the data well, the green line is overfit.*

**Data Conversion Methods:**

## **Bag-of-Words Model**

Text data cannot directly be fed when using machine learning algorithms. Text need to be converted to numbers.

For document classification, each document is an “input” and a class label is the “output” for our predictive algorithm. Algorithms take vectors of numbers as input, therefore we need to convert documents to fixed-length vectors of numbers.

A simple and effective model for thinking about text documents in machine learning is called the Bag-of-Words Model, or BoW.

The model is simple in that it throws away all of the order information in the words and focuses on the occurrence of words in a document.

This can be done by assigning each word a unique number. Then any document we see can be encoded as a fixed-length vector with the length of the vocabulary of known words. The value in each position in the vector could be filled with a count or frequency of each word in the encoded document.

This is the bag of words model, where we are only concerned with encoding schemes that represent what words are present or the degree to which they are present in encoded documents without any information about order.

There are many ways to extend this simple method, both by better clarifying what a “word” is and in defining what to encode about each word in the vector.

## **Word Counts with CountVectorizer**

The [CountVectorizer](http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html) provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

Because these vectors will contain a lot of zeros, we call them sparse. Python provides an efficient way of handling sparse vectors in the [scipy.sparse](https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html) package.

The vectors returned from a call to transform() will be sparse vectors, and can be transformed back to numpy arrays to look and better understand what is going on by calling the fit transform function.

**Categorical Data:**

**Needed/////////////////////////////////////////////////**

**Feature Importance:**

It is equally important to not only have an accurate, but also an interpretable model. Oftentimes, apart from wanting to know what our model’s prediction is, we might need why it is working this way and which features are most important in determining the results. Identifying which variables are important can help us in detection and even improving the product/service.

Knowing feature importance indicated by machine learning models can help us in multiple ways:

* By getting a better understanding of the model’s logic, the model can be verified being correct but also work on improving the model by focusing only on the important variables
* The above can be used for variable selection — we can remove thevariables that are not that significant and have similar or better performance in much shorter training time.

**Chapter III: Methodology/Statement of work**

# **3.1 Introduction:**

# In this thesis, the process of predicting English names using Decision Tree Classifier have been done at first. Next, Bengali names are collected to form a dataset, features are extracted and model is made using different classifiers. Then the model is trained for each classifier and tested later with unlabeled or unseen data for its performance.

**3.2 Objective:**

Accurate prediction of an unknown individual’s gender is desirable for use in marketing, social science, and many other applications in academia and industry. Perhaps the most obvious and telling indicator of a person’s gender is their first name. Names of males and females exhibit very subtle differences. These features are mostly due to the morphological and phonological structure of the name. The linguistic and phonological analysis of American names enlists a number of such features, a subset of those has been chosen by us for understanding the typical characteristics which distinguish between male & female Bangladeshi names.

**3.3 Scope**

In this research, we’re getting started with machine learning. We’ll be building a classifier able to distinguish between boy and girl names. Knowing how many female or male are participating in a program, for labs and classrooms if they need different set up or arrangement, having the information can be a plus point for making arrangements. For different websites or online services, it would certainly help you take greater ownership of your marketing and send more relevant messages to your target audience. You can use the predictive gender feature to target gender-specific products and content in more relevant ways to your audience.

**3.4 hypothesis**

**3.5 Constraints**

As there was a plan of predicting gender by Bangladeshi names to get the number of male and females for purposes, it was needed to go through on some previous work on this field.Some research is done on these issues for which a lot of models have been built.But some of them were not adequate as information.Besides,we needed to go through collecting people’s name on which our research was based.It was needed to get a qualitative data on Bangladeshi names.In this research we have identified various features based upon morphological analysis that can be useful for such classification and evaluate them.

**3.6 Conclusion**

In this section we have discussed about the gender prediction to predict gender and our objective,scope,data collection process.This research is based to predict the gender to improve the marketing and other purposes and adopting some predictive approach and results by analysing the names.

**Diagrams to be added**

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Chapter IV: Process Model /Overview of Application**

**Research Platform & Tools:**

For analysis & mining Jupyter Notebook has been used as the programming platform using Python as the programming language.

**Features:**

Names of males and females exhibit very subtle differences.These features are mostly due to the morphological and phonological structure of the name. The linguistic and phonological analysis of North American names enlists a number of such features, a subset of those has been chosen for understanding the typical characteristics which distinguish between male & female Bangladeshi name.

For the analysis of American & Bangladeshi names dataset features are,

**Vowel Ending:** More female names end with vowels than of males comprising a, e, i, o, u the set of vowels.

**Consonant Ending: More** male names than female ends with a consonant.

**Vowel Start:** Mostly Female names ending with consonants start with vowels.

**Consonant start:** Mostly male names ending with vowel starts with a consonant.

**Length of the word:** Male names are longer than of females.

**First letter, First two letters, First three letters, last letter, Last two letters, Last three letters:** Names ending in -a,- e and -i are likely to be female, while names ending in -k, -o, -r, -s and -t are likely to be male. Names ending in -yn appear to be predominantly female, despite the fact that names ending in -n tend to be male; and names ending in -ch are usually male, even though names that end in -h tend to be female.

**Classifiers:**

Classifiers used for the American dataset are

* Decision Tree
* Multinomial Naive Bayes

Classifiers used for the Bangladeshi dataset are

* Multinomial Naive Bayes
* Decision Tree
* Logistic Regression
* K Nearest Neighbor
* Support Vector Machine
* Linear Discriminant Analysis
* Gaussian Naive Bayes

**Limitations:**

**Existing Research/work reviews:**

SVM based classification has previously been used for language identification of names and has performed better than other language models. Reference [4] has used the n-grams of words and word length as features and has shown that the classification accuracy increases with n. However, they do not use any other morphological information of words. Gender identification of Chinese email documents used format features, linguistic features and structural features of emails in SVM for classification. It concentrates more upon the overall document structure and less on the individual named entity. SVM has been used for Gender identification from many other media such as images , gait recognition and speech signals . To the best of our knowledge, much work has not been done in using SVM classifiers for gender identification of names represented in text.Gender predicting of indonesian names used a new way to predict genders from names using character-level LongShort Term Memory (char-LSTM). They compared their method with some conventional machine learning methods, namely Naıve Bayes, logistic regression, and XGBoost with n-grams as features.

**Chapter V: Predicting Names with Classifiers/Experimental Procedure**

We formed a list of names containing 1586 Bangladeshi names & collected a list of 1048575 American names of which are tagged as “male” & “female”.

Our compiled data contained 622482 female & 426093 male American names & 812 female, 774 male Bangladeshi names. We split the total dataset into separate train & test data, so the testing was performed on unseen or unlabeled data. The training data contained 80% and the remaining 20% were set aside before training for testing purposes.

Different Methods, features & classifiers have been used. We have tried making a model with just a classifier without any feature to see how it goes. Then all the models have been made with different classifiers using different features both with the American and Bangladeshi name datasets. The process is explained in flowcharts-

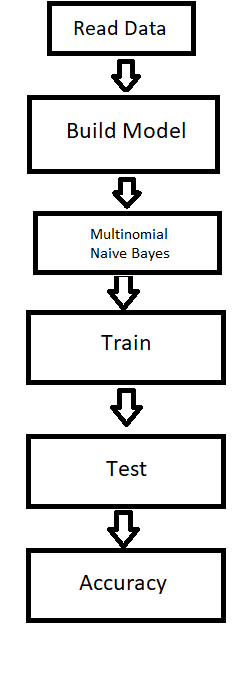
****

Fig: Process 1(Initial process for both types)

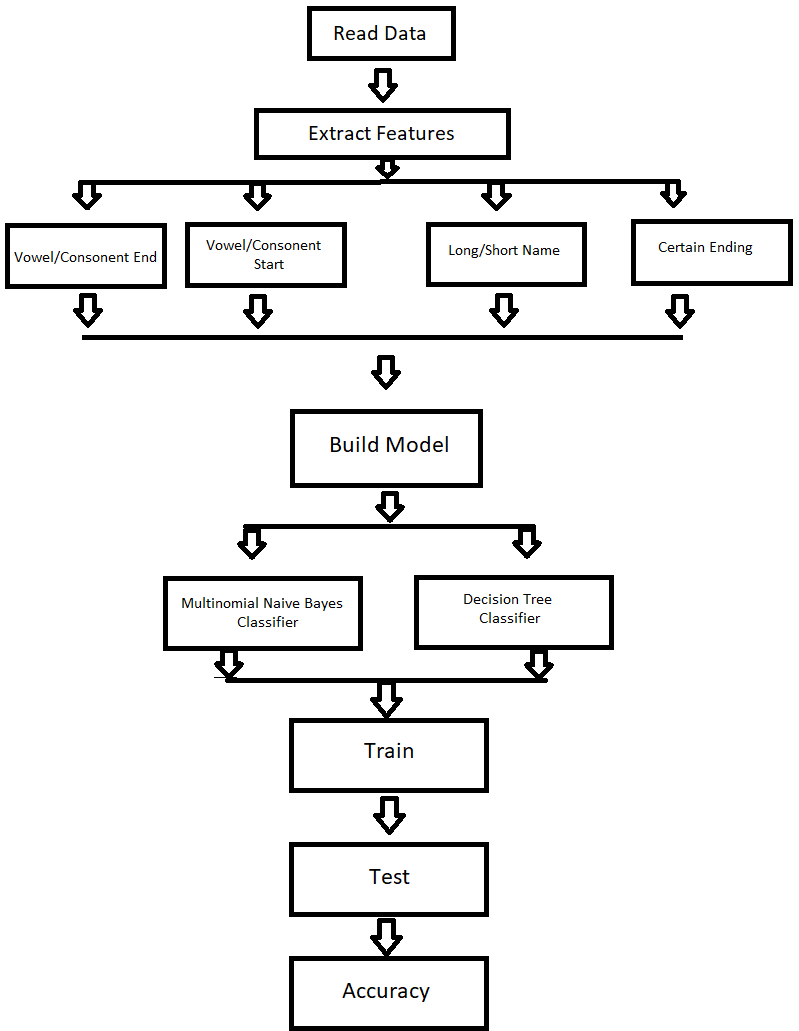
****

Fig: Process 2 (For American Names Dataset)

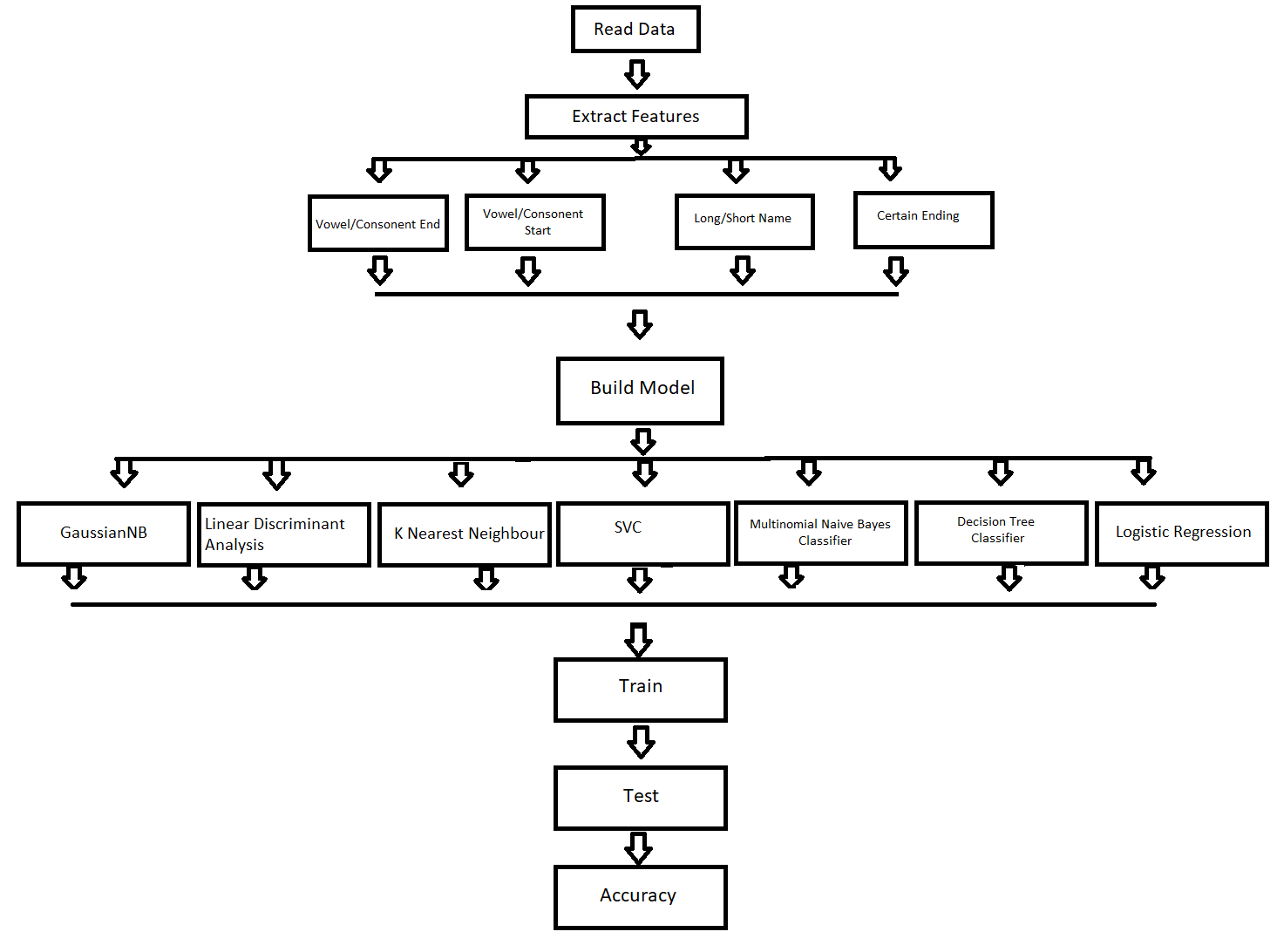
****

Fig: Process 2 (For Bangladeshi Names)

**5.1 Data Collection:**

The American names dataset was collected from kaggle`s website, which contained 1048575 labelled names. The Bangladeshi names come from friends and family members.

**5.2 Conversion of Data:**

Our datasets contained no null values since we collected and checked it before making the final sets. We have checked for null values in the codes as well .Before feeding the data to machine we have applied vectorization for the initial step & later to categorical data as features were added to build models.

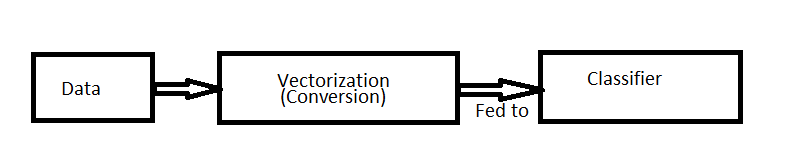


Fig: Conversion for process 1

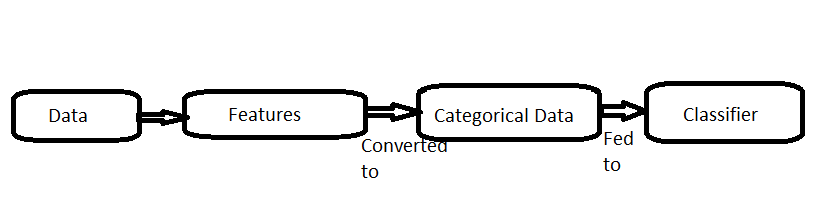


Fig: Conversion for process 2

**5.3 Feature Extraction:**

**5.3.1 For American Name Dataset:**

**5.3.1.1 Vowel/Consonant End:**

Even though there is no rule of naming ,but female names tend to end with vowels than males and more male names tend to end with consonants than of females.We tried using this feature and the result came out -

* Female
* Vowel: 4,73,746
* Consonant: 1,95,035
* Male
* Vowel: 87,440
* Consonant: 2,91,503

**5.3.1.2 Long/Short Name:**

Name length has nothing to do with gender.For research purposes we made a feature that male names are longer than female names.we tried names having the length >7 are long and others are short.The result came out -

* Female
* Long: 3,60,709
* Short: 3,08,281
* Male
* Long: 2,49,946
* Short: 2,29,637

**5.3.1.3 Certain Ending:**

We used a feature that female names tend to end with A, E or has the sound of A.The results are-

* Female
* 3,95,312
* Male
* 69,206

**5.3.1.4 Vowel/Consonant Start:**

* **Female**
* **Vowel:**
* **Consonant:**
* **Male**
* **Vowel:**
* **Consonant:**

**5.3.2 Bangladeshi Name Dataset:**

**5.3.2.1 Vowel/Consonant End:**

We tried using this feature on Bangladeshi name dataset to see how it performs and the result came out -

* Female
* Vowel: 448
* Consonant: 215
* Male
* Vowel: 69
* Consonant: 436

**5.3.2.2 Long/Short Name:**

This feature is applied on Bangladeshi name dataset as well.The result came out -

* Female
* Long: 404
* Short: 260
* Male
* Long: 261
* Short: 243

**5.3.2.3 Certain Ending:**

We used a feature that female names tend to end with A, E or has the sound of A.The results are-

* Female
* 479
* Male
* 32

**5.3.2.4 Vowel/Consonant Start:**

* **Female**
* **Vowel:** 112
* **Consonant:** 552
* **Male**
* **Vowel:** 122
* **Consonant:** 383

**5.4 Model Building:**

We have used different classifiers to build models. Have applied the same classifiers to the American name and Bangladeshi name dataset, have applied more classifiers to the Bangladeshi one and will choose the best model from them.

**5.4.1 Classifiers:**

Classifiers have been used to build models. Each classifiers has its own method of working. We define them to instruct the machine how the machine is supposed to train.

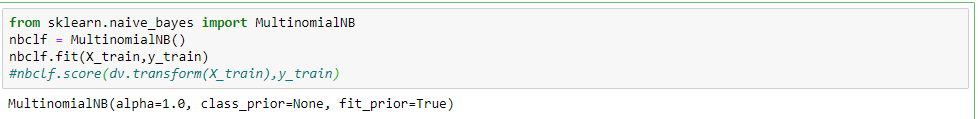
**5.4.1.1 English Name Dataset:**

**5.4.1.1.1 Multinomial Naive Bayes:**

A model was built without using any features to see how it performs. We have applied Multinomial Naive Bayes on the American names set and the accuracy was around 63.89% while the training score was 63.95%.

Later, with applying four features and then building the model, Multinomial Naïve Bayes model gave an accuracy of almost 71.17% & the training accuracy was almost similar of 73.44%.

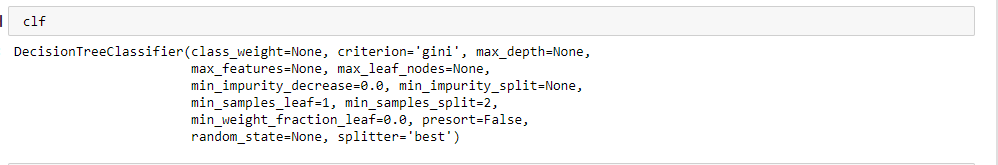
Here is features used in this model:



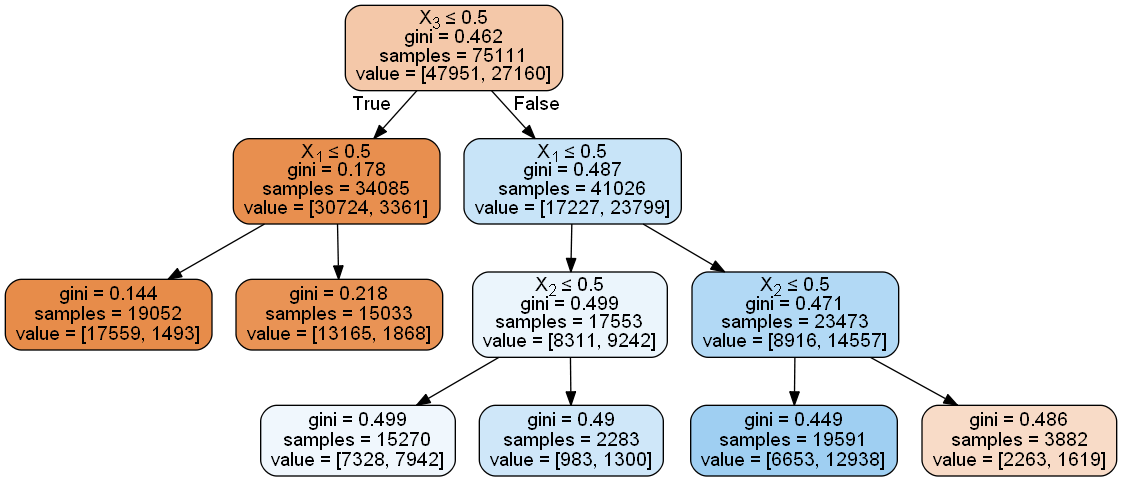
**5.4.1.1.2 Decision Tree:**

Decision tree classifier was added with the existing features and the model test accuracy was around 73.52% with the training score of 73.01%.

Here is the features used in this model:



A visual representation of the generated Decision Tree Model:



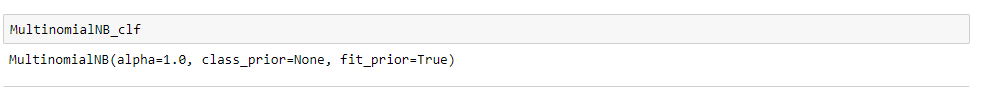
**5.4.1.2 Bangladeshi Name Dataset:**

**5.4.1.2.1 Multinomial Naive Bayes:**

The initial model without any feature gave accuracy of almost 65.88%.

Also, this classifier was used with the four features to build model. This model with the features gave almost 78.12% accuracy with the training accuracy of 74.97%.

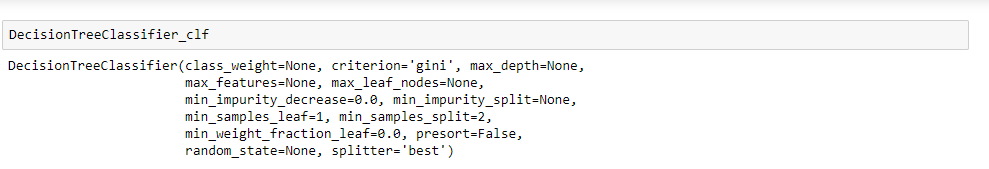
The features used in this model:



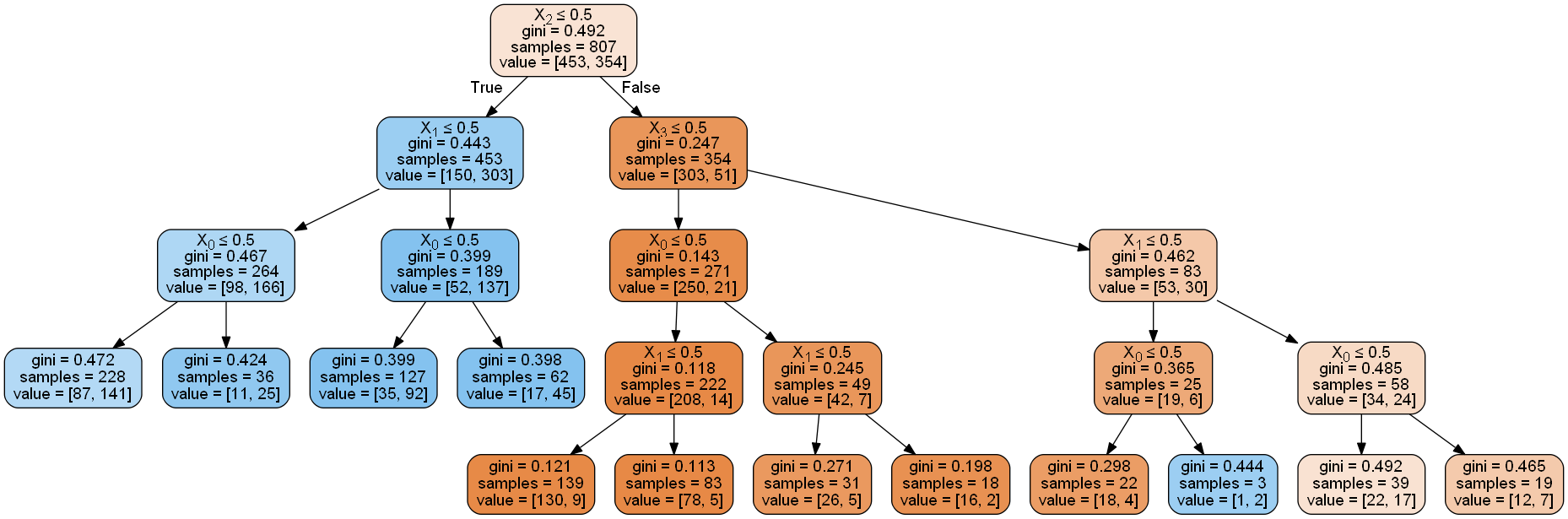
**5.4.1.2.2 Decision Tree:**

Decision tree classifier was used with the four features to build model. This model gave accuracy of around 84.73% while the training accuracy was 81.09%

.



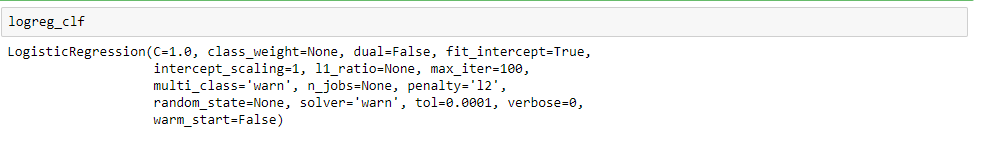
Visual representation of the generated tree:



**5.4.1.2.3 Logistic Regression:**

The Logistic Regression model gave an accuracy of almost 80.19% with the training accuracy of 76.97%.

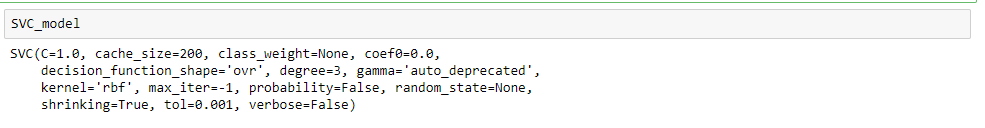
The features of this model:



**5.4.1.2.4 Support Vector Machine:**

The support vector machine model gave the accuracy of around 80.19% with the training accuracy of 76.97%.

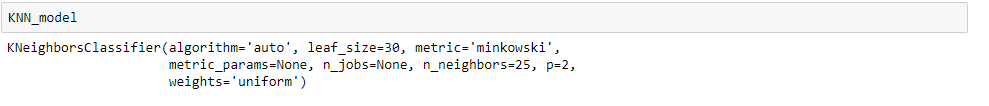
The features of this model are:



**5.4.1.2.5 K-Nearest Neighbor:**

We have taken k = 25 for the model. K-Nearest Neighbor model gave an accuracy score of almost 78.71% with the training accuracy of almost 78.09%.

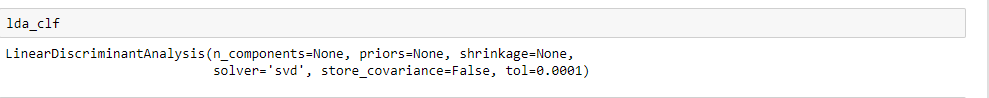
The features used in this model are:



**5.4.1.2.6 Linear Discriminant:**

The Linear Discriminant model gave the accuracy of around 80.19% with the training accuracy of 76.97%.

The features of this model:



**5.4.1.2.7 Gaussian Naive Bayes:**

The Gaussian Naïve Bayes model gave an accuracy of almost 80.19% with the training accuracy of 76.97%.

The features of this model are:



**Training:**

As the ways of how the data will be trained is instructed, next in the training phase data will be trained on the splitted 80% data .In this phase, the machine analyses the patterns of texts and categorizes or classifies it.Training accuracy defines how well a model has been trained.

**Testing:**

The 20% remaining data kept for testing is used in this phase. An initial testing usually is done to see how a trained model works with seen data or with data it has already been trained with known as training accuracy.

Final testing is done on the 20% data, that’s unseen data for the machine. This testing indicates how well the prediction is done known as test accuracy.

**Accuracy:**

Accuracy of each model can differ from each other as each has its own method or way of working. We have discussed the results & how well each model have worked in the next chapter.

**Chapter VI: Results & Discussion**

**5.4.1.1 English Name Dataset:**

**5.4.1.1.1 Multinomial Naive Bayes:**

The Multinomial Naive Bayes model on the American names dataset gave accuracy around 63.89%.



Fig: Multinomial NB without features

Later, with applying four features and then building the model, Multinomial Naive Bayes model gave an accuracy of almost 73.17%.

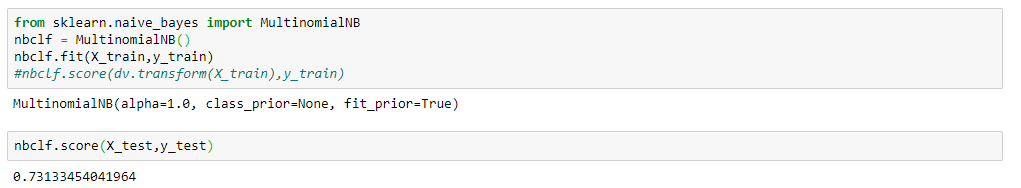


Fig: Multinomial NB with features

**5.4.1.1.2 Decision Tree:**

Decision tree classifier was added with the existing features and the model test accuracy was around 73.52%.

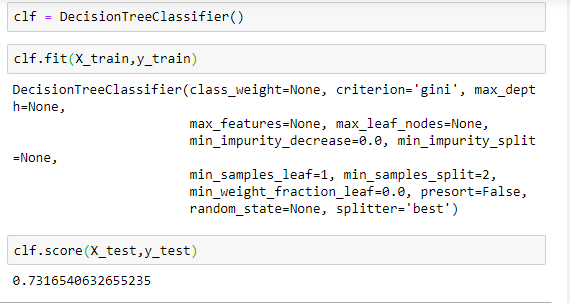


Fig: Decision Tree Classifier

**5.4.1.2 Bangladeshi Name Dataset:**

**5.4.1.2.1 Multinomial Naive Bayes:**

The initial model without any feature gave accuracy of almost 65.88%.



Fig: Multinomial NB without features

Also, this classifier was used with the four features to build model. This model with the features gave almost 78.12% accuracy.



Fig: Multinomial NB with features

**5.4.1.2.2 Decision Tree:**

Decision tree classifier was used with the four features to build model. This model gave accuracy of around 84.73%.

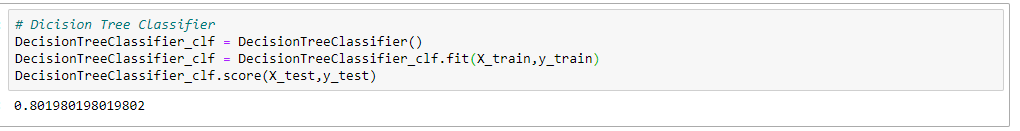


Fig: Decision Tree Classifier

**5.4.1.2.3 Logistic Regression:**

The Logistic Regression model gave an accuracy of almost 80.19%.

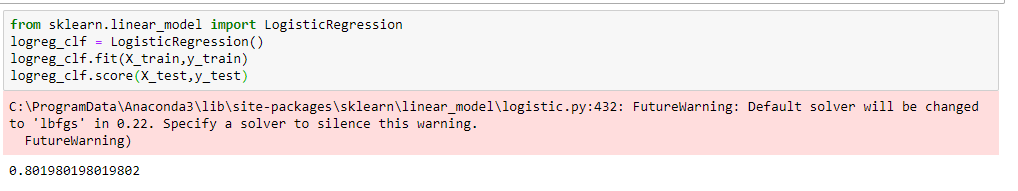


Fig: Logistic Regression Classifier

**5.4.1.2.4 Support Vector Machine:**

The support vector machine model gave the accuracy of around 80.19%.



Fig: Support Vector Classifier

**5.4.1.2.5 K-Nearest Neighbor:**

We have taken k = 25 for the model. K-Nearest Neighbor model gave an accuracy score of almost 78.71%.



Fig: K-Nearest Neighbor Classifier

Later we graphed the scores of k from 1 to 26 to see how it performs.

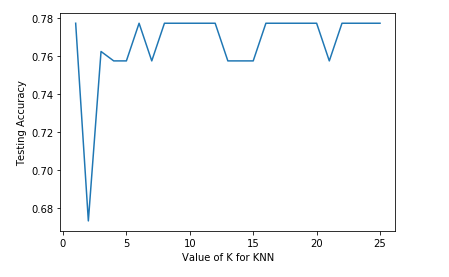


Fig: Accuracy of k value from 1 to 26

**5.4.1.2.6 Linear Discriminant:**

The Linear Discriminant model gave the accuracy of around 80.19%.

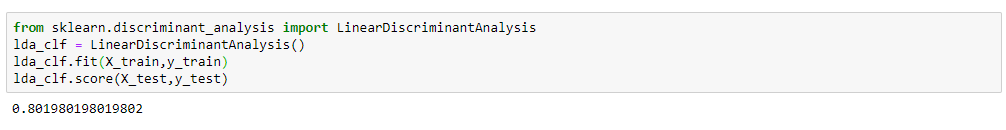


Fig: Linear Discriminant Classifier

**5.4.1.2.7 Gaussian Naive Bayes:**

The Gaussian Naïve Bayes model gave an accuracy of almost 80.19%.



Fig: Gaussian Naive Bayes

The accuracy results in tabular format:

American Name Dataset: (Without Features):

|  |  |  |
| --- | --- | --- |
| **Model** | **Training Score** | **Testing Score** |
| Multinomial Naïve Bayes | 63.95% | 63.89% |

Bangladeshi Name Dataset: (Without Features):

|  |  |  |
| --- | --- | --- |
| **Model** | **Training Score** | **Testing Score** |
| Multinomial Naïve Bayes | 66.79% | 65.88% |

American Name Dataset: (With Features):

|  |  |  |
| --- | --- | --- |
| **Model** | **Training Score** | **Testing Score** |
| Multinomial NB | 73.44% | 71.17% |
| Decision Tree | 73.01% | 73.52% |

Bangladeshi Name Dataset: (With Features):

|  |  |  |
| --- | --- | --- |
| **Model** | **Training Score** | **Testing Score** |
| Multinomial NB | 74.97% | 78.12% |
| Decision Tree | 81.09% | 84.73% |
| Logistic Regression | 76.97% | 80.19% |
| SVM | 76.97% | 80.19% |
| KNN | 78.09% | 78.71% |
| Linear Discriminant | 76.97% | 80.19% |
| Gaussian NB | 76.97% | 80.19% |

As we can see,

1. The training and testing scores are quite okay as the testing score and training scores of each models have compatible scores.
2. Between American & Bangladeshi names, Bangladeshi names model of Multinomial Naïve Bayes (without features) have a bit better score than the American one.
3. Even with the features, Bangladeshi names` model testing scores are better than the American one`s.
4. Among all the seven models of Bangladeshi names, Decision Tree worked the best having the testing score of almost 84.73%.

So, the final decision is:

|  |  |
| --- | --- |
| **Algorithms & Features worked best on** | **Best Classifier**  **for**  **Bangladeshi Names** |
| Bangladeshi Names | Decision Tree |

**Chapter VII: Conclusion**

In this research we have analyzed American names and Bangladeshi names and compared them. Between both, the same classifiers and features have worked best on Bangladeshi names and Decision Tree classifier has proved the best among all.

**Reference:**

1. S. Slater and S. Feinman, “Gender and the phonology of north American first names,” Sex Roles, vol. 13, pp. 429–440, 1985, 10.1007/BF00287953.
2. “Grammatical Gender.” *Wikipedia*, Wikimedia Foundation, 21 Dec. 2017, en.wikipedia.org/wiki/Grammatical gender.
3. S. Slater and S. Feinman, “Gender and the phonology of north American first names,” Sex Roles, vol. 13, pp. 429–440, 1985, 10.1007/BF00287953.
4. Bhargava and G. Kondrak, “Language identification of names with svms,” in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Los Angeles, California: Association for Computational Linguistics, June 2010, pp. 693–696.
5. G.-F. Teng, W.-Q. Dong, J. Yang, and J.-B. Ma, “Gender identification for Chinese email documents,” in Proceedings of the Second International Conference on Innovative Computing, Information and Control, ser. ICICI ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 36–.
6. H. cheng Lian, B. Liang Lu, and S. Hosoi, “L.: Gender recognition using a min-max modular support vector machine,” in In: Proc. ICNC05- FSKD05, LNCS 3611. Springer-Verlag, 2005, pp. 433–436
7. J. Yoo, D. Hwang, and M. S. Nixon, “Gender classification in human gait using support vector machine,” Lecture notes in computer science, vol. 3708, p. 138, 2005
8. K.-H. LEE, S.-I. KANG, D.-H. KIM, and J.-H. CHANG, “A support vector machine-based gender identification using speech signal,” 2008

<https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/>

1. <https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e>
2. Xia, B. S., & Gong, P. (2015). Review of business intelligence through data analysis. Benchmarking, 21(2), 300-311. doi:10.1108/BIJ-08-2012-0050
3. “Neighbourhood Components Analysis”, J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Advances in Neural Information Processing Systems, Vol. 17, May 2005, pp. 513-520.
4. Fan, Rong-En, et al., “LIBLINEAR: A library for large linear classification'', Journal of machine learning research 9.Aug (2008): 1871-1874.
5. Wu, Lin and Weng, “Probability estimates for multi-class classification by pairwise coupling”, JMLR 5:975-1005, 2004.
6. Platt “Probabilistic outputs for SVMs and comparisons to regularized likelihood methods”.
7. “Automatic Capacity Tuning of Very Large VC-dimension Classifiers”, I. Guyon, B. Boser, V. Vapnik - Advances in neural information processing 1993.
8. “Support-vector networks”, C. Cortes, V. Vapnik - Machine Learning, 20, 273-297 (1995).
9. https://en.wikipedia.org/wiki/Decision\_tree\_learning
10. https://en.wikipedia.org/wiki/predictive\_analytics
11. J.R. Quinlan. C4. 5: programs for machine learning. Morgan Kaufmann, 1993.
12. T. Hastie, R. Tibshirani and J. Friedman. Elements of Statistical Learning, Springer, 2009.
13. H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.
14. Mccallum, A. "A comparison of event models for naive bayes text classification." Proc. AAAI-98 Workshop on Learning for Text Categorization (1998).pp.41--48.
15. “The Elements of Statistical Learning”, Hastie T., Tibshirani R., Friedman J., Section 4.3, p.106-119, 2008.
16. Ledoit O, Wolf M. Honey, I Shrunk the Sample Covariance Matrix. The Journal of Portfolio Management 30(4), 110-119, 2004.
17. An Introduction to Statistical Learning: with Applications in R, Chapter 4, Page 138.
18. Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning, chapter 8
19. Bird, S., Klein, E., and Loper, E. “6.1.1 Gender Identification.” Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, O’Reilly, 2009, [www.nltk.org/book/ch06.html](http://www.nltk.org/book/ch06.html).
20. <https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63>
21. <https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623>
22. American name dataset, <https://www.kaggle.com/springboardroger/naive-bayes-name-gender-classifier/data>
23. <https://data-flair.training/blogs/train-test-set-in-python-ml/>
24. <https://nlpforhackers.io/introduction-machine-learning/>

**Appendix:**

American name dataset :

from pandas import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction import DictVectorizer

df = pd.read\_csv('NationalNames.csv')

print ("%d names in dataset" %len(df))

df = df.drop\_duplicates(subset="Name")

df.head(10)

print (df[df.Gender == 'F'].count())

print (df[df.Gender == 'M'].count())

Xfeatures =df['Name']

cv = CountVectorizer()

X = cv.fit\_transform(Xfeatures)

# Labels

y = df.Gender

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

# Naive Bayes Classifier

from sklearn.naive\_bayes import MultinomialNB

clf = MultinomialNB()

clf.fit(X\_train,y\_train)

clf.score(X\_train,y\_train)

clf.score(X\_test,y\_test)

# Check if the name ends in vowel

def checkVowelEnd(name):

if name[-1] in "aeiou":

return "Vowel End"

return "Consonant End"

df["Vowel/Consonant End"] = df["Name"].apply(checkVowelEnd)

df.head()

def checkGender(gender):

if gender == "F":

return 0

else:

return 1

df["Gender Value"] = df["Gender"].apply(checkGender)

df.head(-20)

#df.count

def compare(group):

return df.groupby([group])["Gender Value"].sum()\*100/df.groupby([group])["Gender Value"].count()

# df.groupby(["Vowel/Consonant End"])['Gender Value'].count()

# df.head()

g=df.groupby(['Vowel/Consonant End','Gender Value'])

g.head()

print (len(df))

# > 93889 = 43635 + 50254

#female\_names = sum(df.groupby(["Vowel/Consonant End"])["Gender Value"].sum())

female\_names =( df.groupby(['Vowel/Consonant End','Gender Value'])).size()

all\_names = df.groupby(["Gender"])["Gender Value"].count()

print (female\_names)

#print (all\_names)

# print ("\nBoth are equal? %s" % str(female\_names == all\_names["F"]))

print(df.groupby(["Vowel/Consonant End"])["Gender Value"].sum()\*100/df.groupby(["Vowel/Consonant End"])["Gender Value"].count())

print(compare("Vowel/Consonant End"))

def vowelConsonantstart(name):

if name[0] in "aeiou":

return "vowel start"

else:

return "consonant Start"

df["Vowel/Consonant Start"] = df["Name"].apply(vowelConsonantstart)

names =( df.groupby(['Vowel/Consonant Start','Gender Value'])).size()

print(names)

#print("\n Comparison => %s", compare("Vowel/Consonant Start"))

df.head()

def shortLongName(name):

if len(name) < 7:

return "Short"

else:

return "Long"

df["Short/Long Name"] = df["Name"].apply(shortLongName)

longshort\_name =( df.groupby(['Short/Long Name','Gender Value'])).size()

print(longshort\_name)

#print(compare("Short/Long Name"))

df.head()

# 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:],

}

df["features"] = df["Name"].apply(features)

df.head()

def checkfeature(name):

if name[-1] in "a,e":

return "a"

return "b"

df["Feature End"] = df["Name"].apply(checkfeature)

df.head(-20)

g=df.groupby(['Feature End','Gender'])

g.head()

names =(df.groupby(['Feature End','Gender'])).size()

#female\_names =(df.groupby(['features','Gender Value'])).count()

#all\_names = df.groupby(["Gender"])["Gender Value"].count()

print (names)

#df.info()

# training\_data = df[['Gender Value', 'Vowel/Consonant Start', 'Short/Long Name', 'Vowel/Consonant End']]

# training\_data.head()

X= df[['Vowel/Consonant Start', 'Short/Long Name', 'Vowel/Consonant End','Feature End']]

y= df ['Gender Value']

#X.head()

#y.head()

def reprCategory(column):

column = column.astype("category")

return column.cat.codes

# training\_data[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start"]] = training\_data[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start"]].apply(reprCategory)

# training\_data.info()

#len(training\_data)

X[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"]] = X[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"]].apply(reprCategory)

#training\_data.head()

X.head()

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size = 0.20)

len(X\_train)

len(y\_train)

len(X\_test)

len(y\_test)

clf = DecisionTreeClassifier()

clf.fit(X\_train,y\_train)

clf.score(X\_train,y\_train)

clf.score(X\_test,y\_test)

clf.predict(X\_test)

y\_test

from sklearn.externals.six import StringIO

from IPython.display import Image

from sklearn.tree import export\_graphviz

import pydotplus

dot\_data = StringIO()

export\_graphviz(clf, out\_file=dot\_data,

filled=True, rounded=True,

special\_characters=True)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

Image(graph.create\_png())

Clf

clf.feature\_importances\_

from sklearn.naive\_bayes import MultinomialNB

nbclf = MultinomialNB()

nbclf.fit(X\_train,y\_train)

#nbclf.score(dv.transform(X\_train),y\_train)

Nbclf

nbclf.score(X\_train,y\_train)

nbclf.score(X\_test,y\_test)

with open("decidenamesE.dot", "w") as dot\_file:

dot\_file = export\_graphviz(clf,

feature\_names=["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"], out\_file=dot\_file)

**For Bangladeshi Name**

from pandas import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction import DictVectorizer

df = pd.read\_csv('BanglaNames.csv')

print ("%d names in dataset" %len(df))

df = df.drop\_duplicates(subset="Name")

df.head(20)

print (df[df.Gender == 'female'].count())

print (df[df.Gender == 'male'].count())

Xfeatures =df['Name']

cv = CountVectorizer()

X = cv.fit\_transform(Xfeatures)

# Labels

y = df.Gender

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

# Naive Bayes Classifier

from sklearn.naive\_bayes import MultinomialNB

clfNB = MultinomialNB()

clfNB.fit(X\_train,y\_train)

clfNB.score(X\_train,y\_train)

clfNB.score(X\_test,y\_test)

clfNB

# Check if the name ends in vowel

def checkVowelEnd(name):

if name[-1] in "aeiou":

return "Vowel End"

else:

return "Consonant End"

df["Vowel/Consonant End"] = df["Name"].apply(checkVowelEnd)

df.head()

def checkGender(gender):

if gender == "female":

return 0

else:

return 1

df["Gender Value"] = df["Gender"].apply(checkGender)

df.head()

def compare(group):

return df.groupby([group])["Gender Value"].sum()\*100/df.groupby([group])["Gender Value"].count()

df.groupby(["Vowel/Consonant End"])['Gender Value'].count()

df.groupby(['Vowel/Consonant End','Gender Value']).size()

print (len(df))

female\_names = sum(df.groupby(["Vowel/Consonant End"])["Gender Value"].sum())

all\_names = df.groupby(["Gender"])["Gender Value"].count()

print (all\_names)

print ("\nBoth are equal? %s" % str(female\_names == all\_names["female"]))

print(df.groupby(["Vowel/Consonant End"])["Gender Value"].sum()\*100/df.groupby(["Vowel/Consonant End"])["Gender Value"].count())

print(compare("Vowel/Consonant End"))

def vowelConsonantStart(name):

if name[0] in "aeiou":

return "Vowel Start"

else:

return "Consonant Start"

df["Vowel/Consonant Start"] = df["Name"].apply(vowelConsonantStart)

df.groupby(['Vowel/Consonant Start','Gender Value']).size()

#print("\n Comparison => %s", compare("Vowel/Consonant Start"))

#df.head()

def shortLongName(name):

if len(name) < 6:

return "Short"

else:

return "Long"

df["Short/Long Name"] = df["Name"].apply(shortLongName)

df.groupby(['Short/Long Name','Gender Value']).size()

#print(compare("Short/Long Name"))

#df.head(20)

# 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:],

}

df["features"] = df["Name"].apply(features)

df.head()

def checkfeature(name):

if name[-1] in "a,e":

return "a"

return "b"

df["Feature End"] = df["Name"].apply(checkfeature)

df.head(-20)

g=df.groupby(['Feature End','Gender'])

g.head()

names =(df.groupby(['Feature End','Gender'])).size()

#female\_names =(df.groupby(['features','Gender Value'])).count()

#all\_names = df.groupby(["Gender"])["Gender Value"].count()

print (names)

training\_data = df[["Gender Value", "Vowel/Consonant Start", "Short/Long Name", "Vowel/Consonant End","Feature End"]]

# training\_data.head()

X= df[['Vowel/Consonant Start', 'Short/Long Name', 'Vowel/Consonant End','Feature End']]

y= df ['Gender Value']

def reprCategory(column):

column = column.astype("category")

return column.cat.codes

#training\_data[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"]] = training\_data[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"]].apply(reprCategory)

X[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"]] = X[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start","Feature End"]].apply(reprCategory)

#training\_data.head()

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size = 0.20)

#train,test = train\_test\_split(training\_data, test\_size = 0.20)

len(X\_train)

len(X\_test)

len(y\_train)

len(y\_test)

from sklearn.naive\_bayes import MultinomialNB

MultinomialNB\_clf = MultinomialNB()

MultinomialNB\_clf.fit(X\_train,y\_train)

MultinomialNB\_clf.predict(X\_test)

df= MultinomialNB\_clf.predict(X\_test)

print (str(df))

y\_test

MultinomialNB\_clf.score(X\_train,y\_train)

MultinomialNB\_clf.score(X\_test,y\_test)

MultinomialNB\_clf

# Dicision Tree Classifier

DecisionTreeClassifier\_clf = DecisionTreeClassifier()

DecisionTreeClassifier\_clf = DecisionTreeClassifier\_clf.fit(X\_train,y\_train)

DecisionTreeClassifier\_clf.score(X\_train,y\_train)

DecisionTreeClassifier\_clf.score(X\_test,y\_test)

DecisionTreeClassifier\_clf

from sklearn.linear\_model import LogisticRegression

logreg\_clf = LogisticRegression()

logreg\_clf.fit(X\_train,y\_train)

logreg\_clf.score(X\_train,y\_train)

logreg\_clf.score(X\_test,y\_test)

logreg\_clf

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

SVC\_model = SVC()

SVC\_model.fit(X\_train,y\_train)

#prediction = SVC\_model.predict(test[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start"]])

#accuracy\_score(test["Gender Value"], prediction)

SVC\_model.score(X\_train,y\_train)

SVC\_model.score(X\_test,y\_test)

SVC\_model

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report

KNN\_model = KNeighborsClassifier(n\_neighbors=25)

KNN\_model.fit(X\_train,y\_train)

#predictions = KNN\_model.predict(test[["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start"]])

#accuracy\_score(test["Gender Value"], predictions)

KNN\_model.score(X\_train,y\_train)

KNN\_model.score(X\_test,y\_test)

KNN\_model

# try K=1 through K=25 and record testing accuracy

from sklearn import metrics

k\_range = range(1, 26)

# We can create Python dictionary using [] or dict()

scores = []

# We use a loop through the range 1 to 26

# We append the scores in the dictionary

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(X\_train,y\_train)

y\_pred = knn.predict(X\_test)

scores.append(metrics.accuracy\_score(y\_test, y\_pred))

print(scores)

# import Matplotlib (scientific plotting library)

import matplotlib.pyplot as plt

# allow plots to appear within the notebook

%matplotlib inline

# plot the relationship between K and testing accuracy

# plt.plot(x\_axis, y\_axis)

plt.plot(k\_range, scores)

plt.xlabel('Value of K for KNN')

plt.ylabel('Testing Accuracy')

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

lda\_clf = LinearDiscriminantAnalysis()

lda\_clf.fit(X\_train,y\_train)

lda\_clf.score(X\_train,y\_train)

lda\_clf.score(X\_test,y\_test)

lda\_clf

from sklearn.naive\_bayes import GaussianNB

gnb\_clf = GaussianNB()

gnb\_clf.fit(X\_train,y\_train)

gnb\_clf.score(X\_train,y\_train)

gnb\_clf.score(X\_test,y\_test)

gnb\_clf

with open("decidenamesB.dot", "w") as dot\_file:

dot\_file = export\_graphviz(DecisionTreeClassifier\_clf,

feature\_names=["Vowel/Consonant End", "Short/Long Name", "Vowel/Consonant Start",""], out\_file=dot\_file)

/////////////////////////////////////

As stated in the previous section female names differ from that of males in terms of the numSyll, lenWord, isVowel and isSonorant. On one hand, we have features like numSyll & lenWord which do not differ a lot for the two categories and on the other hand, the percentage of words showing isVowel and isSonorant features vary largely across the two categories. This gives us an idea that isVowel and isSonorant are the two features which may primarily help in classifying a name. As suggested by [2] we include n-gram features as well for our analysis. Including n-gram features would try to identify the set of alphabets which occur together frequently as prefixes, postfixes or in between the word in male and female names. Since all names in our training data set had length ≥ 4 we chose 1-gram, 2-gram & 3-gram features in our experiments. We do not include 4-gram feature as it may lead to overfitting on the training data and processing it is computationally much more expensive than n-grams of lower degree

////////////////////////////////////////