**ABSTRACT**

**Abstract:**

We have various machine algorithms for gender classification but choosing best one is important task. For selecting best algorithm we conducted experimental study on machine learning algorithms for gender classification. In this experimental study of machine learning algorithms, we analyzed performance of various algorithms for gender classification using voice dataset. From this study we concluded that SVM and ANN are giving best results. After tuning parameters ANN outperforms SVM giving accuracy 99.87% on test data**.**

**INTRODUCTION**

**1.Introduction**

**1.1 Overview:**

Gender prediction is important in applications like targeted advertisements, interactive systems and mobile based health care systems. Based on the gender of a person interactive systems respond accordingly. If marketing firms know the the gender of the person then they can target respective people who potentially buy the products. Classifying the gender of a person accurately based on their voice is a challenging problem in machine learning. Deep learning models are more suitable for unstructured data like audio, video and images. Deep learning models perform better results when the data is large. In this paper we used the voice dataset consists of 3168 male and female voice acoustic features to train different machine learning algorithms. From this research we compared the accuracy of different algorithms.

**1.2 Company Profile**

**Sanguine Global**

**About Sanguine Global:**

Sanguine is a specialists recruitment agency, now perfectly positioned to offer you with the best career opportunities nationwide, leveraging our extensive client relationships across important global business hubs. At Sanguine we are pep-up to provide our clients and applicants with solutions that offer them a competitive edge in today's FierceMarket place. It work with a Performance-Based Hiring methodology, resulting in a dramatically reducing the risk of a wrong hire.

Sanguine have access to the best executive talent pool. It use personalized, dynamic, multichannel approach that gets the jobs in front of the right candidates with the right message at the right time.

It aim to be a progressive, modern company where employees have the power of shared values and aspirations and also the kind of organisation that puts self-management at front of every decision. After all, resisting change is a great way to fail.

The well-deserved success of Sanguine Global Services lies in the philosophy "that job satisfaction for working people can only come if there is a right person for the right job".

The company experts in recruitment; its what we do efficiently, confidentially and with the utmost level of professionalism. It finds you the most talented and capable individuals within the market, without just picking from those candidates who are ‘on the market’.

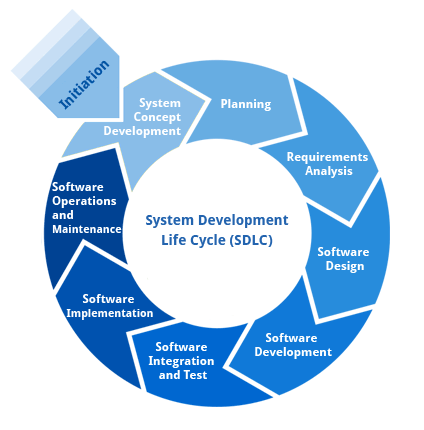
It work with a Performance-Based Hiring methodology, resulting in a dramatically reducing the risk of a wrong hire. It has access to the best executive talent pool. It use personalized, dynamic, multichannel approach that gets the jobs in front of the right candidates with the right message at the right time.

Sanguine position ourselves as a premium provider of excellence within our industry; it measure ourselves closely on our successes and actively seek feedback from our clients to ensure we get it right every time.

**2. Literature Overview**

**2.1 SOFTWARE DEVELOPMENT LIFE CYCLE:**

The Systems Development Life Cycle (SDLC), or Software Development Life Cyclein systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



**2.1.1 Requirement Analysis and Design**

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

**2.1.2 Implementation**

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like PYTHON 3.6, Anaconda Cloud are used for coding. With respect to the type of application, the right programming language is chosen.

**2.1.3 Testing**

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

**2.1.4 Maintenance**

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

**Software developing process model**

**Waterfall model:**

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

## Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** −The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**Waterfall Model – Application**

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with required expertise are available to support the product.
* The project is short.

**Waterfall Model - Advantages**

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows -

* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented.

**Waterfall Model - Disadvantages**

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow
* identifying any technological or business bottleneck or challenges early.

**2.3 Architecture:**

* Client/server architecture is a producer/consumer computing architecture where the server acts as the producer and the client as a consumer. The server houses and provides high-end, computing-intensive services to the client on demand. These services can include application access, storage, file sharing, printer access and/or direct access to the server’s raw computing power.
* Client/server architecture works when the client computer sends a resource or process request to the server over the network connection, which is then processed and delivered to the client. A server computer can manage several clients simultaneously, whereas one client can be connected to several servers at a time, each providing a different set of
* services. In its simplest form, the internet is also based on client/server architecture where web servers serve many simultaneous users with website data.

****

**Figure 2.Architecture forSentiment Analysis to Predict Election Results.**

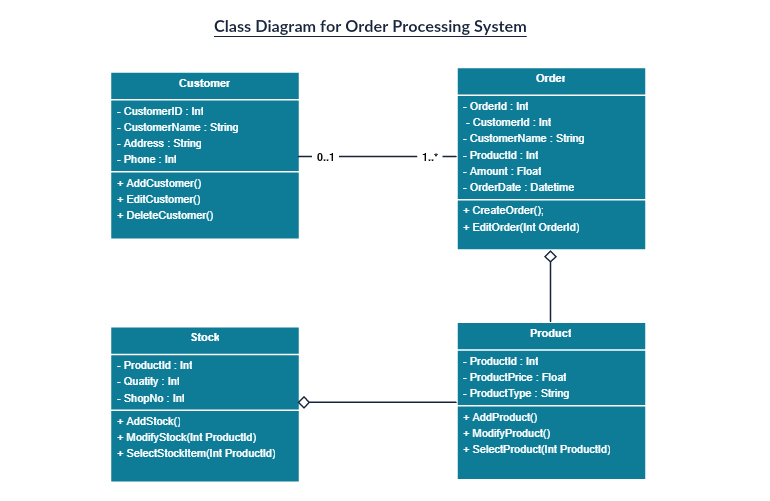
**2.4 Object Oriented System Development:**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* components(logical)
* activities
* programming language statements
* database schemas, and
* Reusable software components.

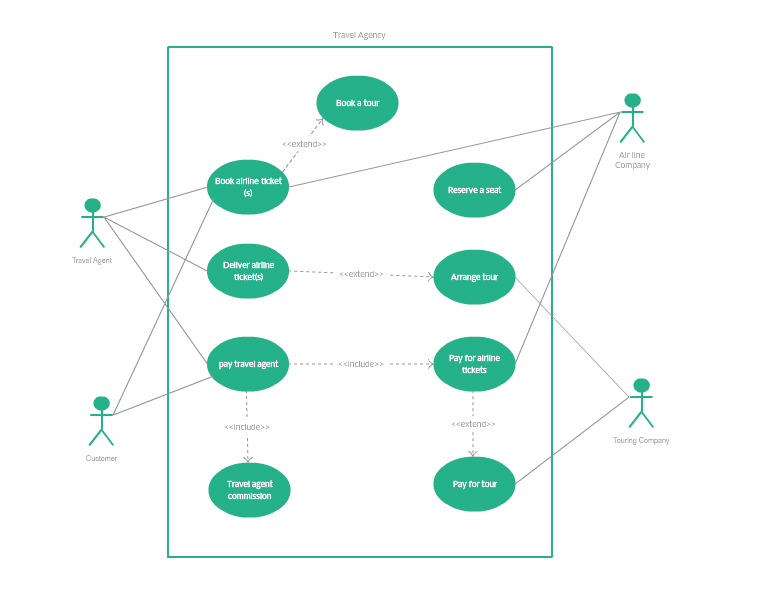
**2.4.1 Class diagram:**

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Active class is used in a class diagram to represent the concurrency of the system.



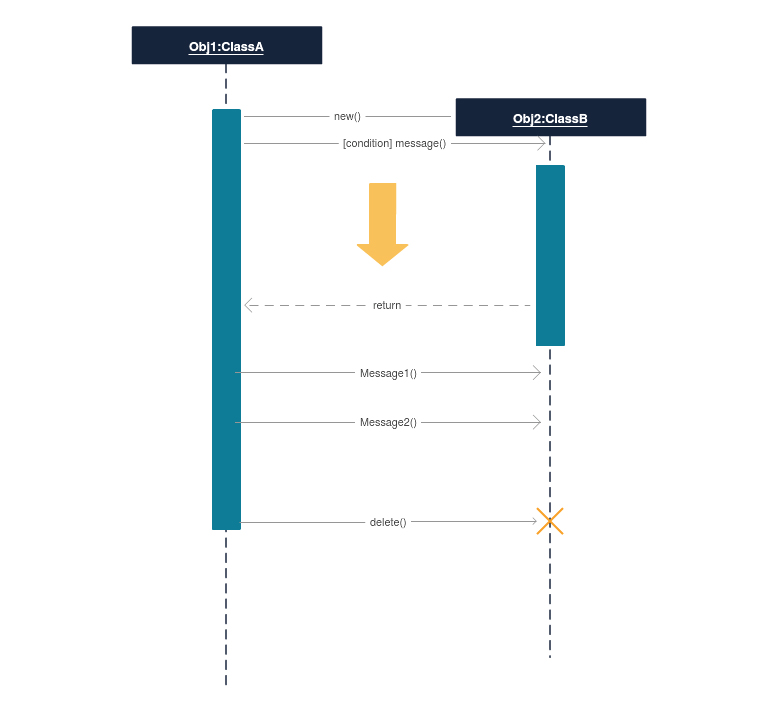
**2.4.2Usecase Diagram**:

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as **actors**.



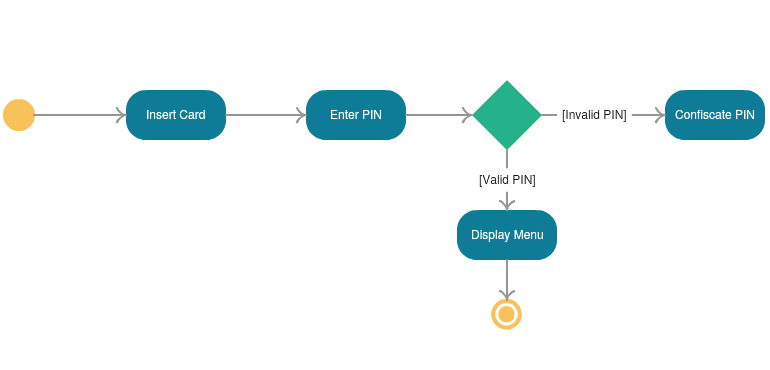
**2.4.3 Sequence Diagram:**

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.



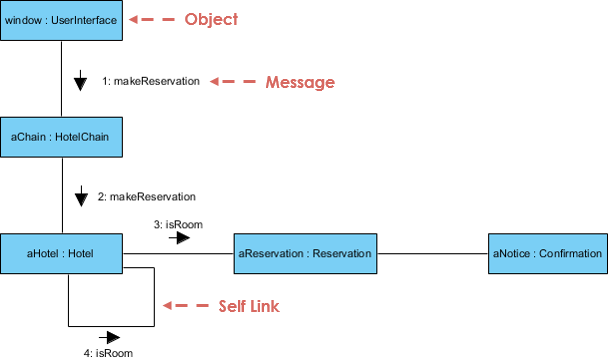
**2.4.4 Activity Diagram:**

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.



**2.4.5 Collaboration Diagram**

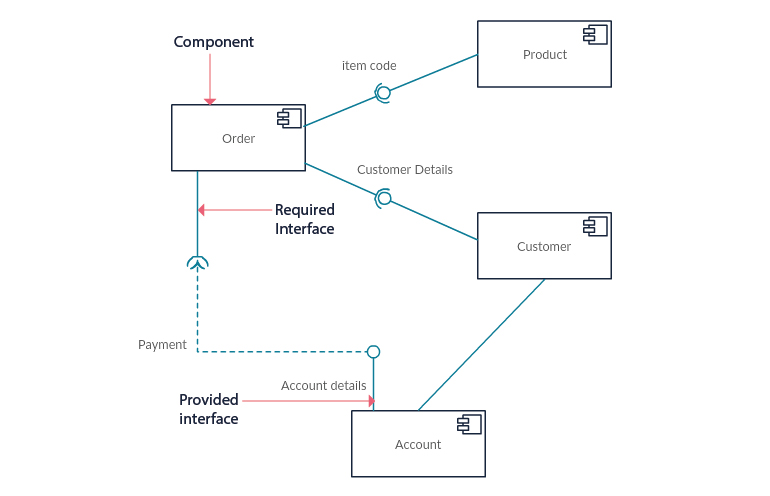
A Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task. A Collaboration defines a set of participants and relationships that are meaningful for a given set of purposes.



**2.4.6 Component diagram:**

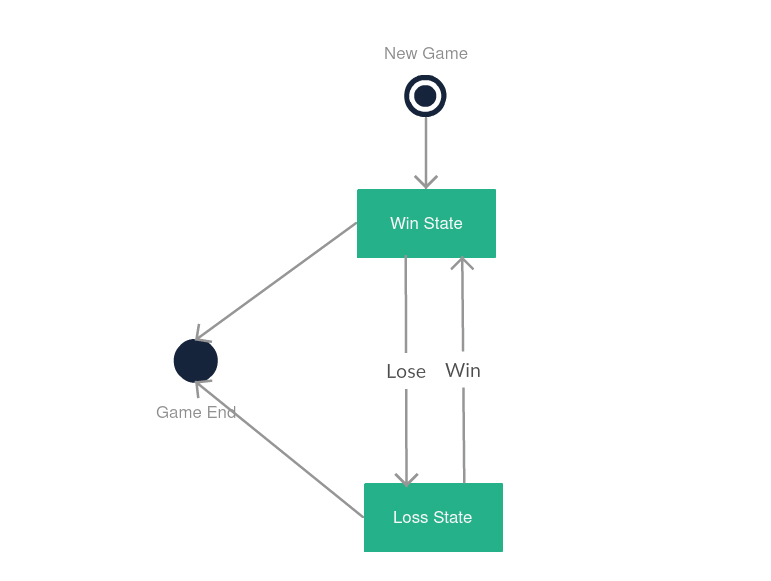
Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagram represent the implementation view of a system.



**2.4.7 State chart Diagram**

Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. State chart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc.



**2.5 literature survey**

**Performance Analysis of Machine Learning Algorithms for Gender Classification**

Gender identification based on the voice of a speaker consists of detecting a speech signal uttered by a male or a female. Automatically detecting the gender of a speaker has several potential applications. In the context of Automatic Speech Recognition, gender dependent models are more accurate than gender independent ones. Hence, gender recognition is needed prior to the application of one gender dependent model. In the context of speaker recognition, gender detection can improve the performance by limiting the search space to speakers from the same gender. Also, in the context of content based multimedia indexing the speaker’s gender is a cue used in annotation. Therefore, automatic gender detection can be a tool in a content-based multimedia indexing system. This paper describes an approach for voicebased gender identification for audio-visual content-based indexing. Several acoustic conditions exist in audio-visual data such as compressed speech, telephone quality speech, noisy speech, speech over background music, studioquality speech, different languages, and so on. Gender identification system must be able to process this variety of speech conditions with acceptable performance. Gender identification is an important step in speaker and speech recognition systems. In these systems, the gender identification step transforms the gender independent problem into a gender dependent one, thus it can reduce the size and complexity of the problem.For speech signal based on gender identification, the most commonly used features are pitch period and Mel-Frequency Cepstral Coefficients (MFCC). The main intuition for using the pitch period comes from the fact that the average fundamental frequency (reciprocal of pitch period) for men is typically in the range of 100-146 Hz, whereas for women it is 188-221 Hz. However, there are several challenges while using pitch period as the feature for gender identification.

The need of gender identification from speech arises several situation such as sorting telephonic call. Many methods of gender identification have been proposed in literature. We implemented the gender classification method and gender dependant feature such as pitch, roll of and energy in combination with MFCC. The clustered approach of above said parameter is implemented using SVM. We also present the experimental result of the proposed approach .It is observed that the accuracy of gender identification system is improved on the basis of size of codebook .The high accuracy is got at 25 codebook size with greater time slice. The accuracy of system tested with respective to gender and age .The efficient recognition rate of 95% is achieved.

A Support Vector Machine (SVM) performs classification by constructing an N-dimensional hyper plane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptions neural network. Support Vector Machine (SVM) models are a close cousin to classical multilayer perception neural networks. Using a kernel function, SVM’s are an alternative training method for polynomial, radial basis function and multi-layer perception classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training. In the parlance of SVM literature, a predictor variable is called an attribute, and a transformed attribute that is used to define the hyper plane is called a feature. The task of choosing the most suitable representation is known as feature selection. A set of features that describes one case (i.e., a row of predictor values) is called a vector. So the goal of SVM modeling is to find the optimal hyper plane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyper plane are the support vectors

An automatic gender detection may be useful in some cases of a mobile healthcare system. For example, there are some pathologies, such as vocal fold cyst, which mainly occur in female patients. If there is an automatic method for gender detection embedded into the system, it is easy for a healthcare professional to assess and prescribe appropriate medication to the patient. In human voice production system, contribution of the vocal folds is very vital. The length of the vocal folds is gender dependent; a male speaker has longer vocal folds than a female speaker. Due to longer vocal folds, the voice of a male becomes heavy and, therefore, contains more voice intensity. Based on this idea, a new type of time domain acoustic feature for automatic gender detection system is proposed in this paper. The proposed feature measures the voice intensity by calculating the area under the modified voice contour to make the differentiation between males and females. Two different databases are used to show that the proposed feature is independent of text, spoken language, dialect region, recording system, and environment. The obtained results for clean and noisy speech are 98.27% and 96.55%, respectively

The applications of automatic gender detection (AGD) system have increased significantly due to the recent developments in speech/speaker recognition, human-computer interaction, and biometric security systems including authentication to access data, surveillance, and security. Gender detection systems limit the search of an imposter to half of the space in many recognition and security systems, where the ultimate goal is the identification of a person. Considering different feature extraction and modeling techniques, an AGD for recognition and security systems should be implemented in such a way that it should not increase the complexity of the whole system. Moreover, a gender detection system can be used for automatic transfer of a phone call of a male/female to the relevant person or department. Furthermore, the accuracy of gender dependent models is higher than gender independent models,In a mobile healthcare system, automatic gender detection can play a significant role. There are some vocal folds pathologies , which are biased to a particular gender; for example, vocal folds cyst can be seen particularly in female patients. If there is a mechanism to automatically detect the gender of the patient, it is easier for a care giver or a healthcare professional to prescribe the appropriate treatment. In this system, the voice or speech of the patient is recorded via a smart device, which is connected to the Internet. The voice or speech is then transmitted to a cloud, where a cloud manager authenticates the patient. The manager distributes the task of feature extraction and classification to various servers, where a decision of gender is made. The decision along with medical data is transmitted to registered healthcare professionals for proper treatment. In most of the studies, the acoustic features used for the gender detection depend on the accurate estimation of the fundamental frequency. The accurate estimation of the fundamental frequency is itself a challenging task.

In this study, an automatic gender detection system by using the proposed feature is developed. The proposed feature determines the voice intensity of a speech signal by using the MVC. To implement the feature, Simpson’s rule is used to calculate the area under the MVC. The MVC is obtained after adding a factor in a polynomial of degree three that is fitted through the peaks. The peaks are found from each frame when a speech utterance is blocked into frames. At the end, the calculated area is fed to SVM to make the decision about the type of gender.

Performance Improvement in Automatic Gender Identification Using Hierarchical Clustering

In this paper a hierarchical structure is proposed for automatic gender identification (AGI). In this structure two clustering techniques are used. The first technique is divisive clustering for dividing speakers from each gender to some classes of speakers. The second clustering technique is agglomerative clustering for creating a hierarchical structure. Feature reduction is done by SOAP feature selection to reduce clustering and classification run time. Another idea used in this paper is to choose and use the best classifier in each level of the proposed hierarchical gender classifier. Three classifiers including GMM, SVM and MLP are used. 96.19% gender classification accuracy was obtained on OGI Multilanguage Telephony Corpus which is 3.5%, 2.5%, and 1% better than the performance obtained by traditional classifiers using GMM, SVM and MLP respectively.

### Automatic Gender Identification Using Fusion of Generative and Discriminative Classifiers and Clustering of Speakers from the Same Gender

a two layer classifier fusion technique using clustering of training data from speakers of the same gender for automatic gender identification (AGI). The first layer is an acoustic classification layer for mapping MFCC and pitch acoustic feature space to score space. In this layer, a divisive clustering is proposed for dividing the speakers from each gender to some classes, where speakers in each class have similar vocal articulatory characteristics. Finally, the best structure could map 22 feature coefficients to 5 likelihood scores as new features. The second layer is a back-end classifier that receives the vectors of fused likelihood scores from the first layer. This means that the new feature coefficients are used in the second layer. GMM, SVM and MLP classifiers were evaluated in the middle and back-end layers. 96.53% gender classification accuracy was obtained on OGI multilingual corpus which is much better than the performance obtained by traditional AGI methods.

Automatic Gender Identification (AGI) is a technique to determine the user sex of a voice processing system through speech signal analysis. Automatically detecting the gender of a speaker has several advantages. In speech recognition systems, gender dependent models are more accurate than gender independent ones. For example, the performance of SPHINX-II, an ASR system developed by Carnegie Mellon University, improved when gender dependent parameters were used.In the context of speaker recognition, gender detection can improve the performance by limiting the search space to speakers from the same gender. Also, in the context of content-based multimedia indexing, the speaker’s gender is a cue used in the annotation. In addition, gender dependent speech coders are more accurate than gender independent ones Gender identification has become gradually a matter of concern in recent years. Harb and Chen (2005) used pitch and spectral features with multi layer perceptron classifier and reported 93% of classification accuracy. Some other classifiers have also been used. For example Lee and Lang (2008) used SVM (Support Vector Machine)[4] and Silvosky and Nouza (2006) used G-MM[5]. In this paper, a fusion technique proposed for AGI. Scores obtained from an acoustic classification layer are fused and used as input feature vector for a back-end classification layer.

In order to achieve the fusion of scores in a two class (male and female) classification task, speakers from each gender are clustered in several classes. This clustering helps to implement our fusion technique detailed in section IV and V. Different discriminative and generative classifiers are evaluated to be used in both acoustic and back-end layers. Experimental results approve the appropriateness of the proposed technique.

The first release of Oregon Graduate Institute (OGI) Multilanguage Telephone Corpus collected by Muthusamy [6] was used in our experiments to evaluate the proposed techniques. This database consists of spontaneous factual speech utterances of ten languages uttered over telephone lines. This database contains 90 calls from each language, with a 50-20-20 division into training, development and final-test sets. All three divisions of OGI are employed in our experiments.

**3. Analysis**

**3.1 System Requirement Specification**

**3.1.1Introduction:**

We have various machine algorithms for gender classification but choosing best one is important task. For selecting best algorithm we conducted experimental study on machine learning algorithms for gender classification. In this experimental study of machine learning algorithms, we analyzedvperformance of various algorithms for gender classification using voice dataset. From this study we concluded that SVM and ANN are giving best results. After tuning parameters ANN outperforms SVM giving accuracy 99.87% on test data**.**

.Gender identification is one of the major problem speech analysis today. Tracing the gender from acoustic data i.e., pitch, median, frequency etc. Machine learning gives promising results for classification problem in all the research domains. There are several performance metrics to evaluate algorithms of an area. Our Comparative model algorithm for evaluating 5 different machine learning algorithms based on eight different metrics in gender classification from acoustic data. Agenda is to identify gender, with five different algorithms: Linear Discriminant Analysis (LDA), K-Nearest Neighbour (KNN), Classification and Regression Trees (CART), Random Forest (RF), and Support Vector Machine (SVM) on basis of eight different metrics. The main parameter in evaluating any algorithms is its performance. Misclassification rate must be less in classification problems, which says that the accuracy rate must be high. Location and gender of the person have become very crucial in economic markets in the form of AdSense. Here with this comparative model algorithm, we are trying to assess the different ML algorithms and find the best fit for gender classification of acoustic data.

**3.1.2 Existing system:**

Gender prediction is important in applications like targeted advertisements, interactive systems and mobile based health care systems. Based on the gender of a person interactive systems respond accordingly. If marketing firms know the the gender of the person then they can target respective people who potentially buy the products

**Disadvantage:**

Classifying the gender of a person accurately based on their voice is a challenging problem in machine learning

* + 1. **Proposed System:**

Parameter tuning is used to find the best hyper parameters. GridSearch technique is used to find best hyper parameters. GridSearch will test several combinations of hyper parameters and returns the best selection that gives best accuracy.

We created dictionary with hyper parameters and applied on GridSearchCV of keras library. GridSeachCV will train

Artificial Neural Networks using k-fold cross validation to get relevant accuracy with different combinations of the dictionary of hyper parameters and returns best accuracy with best selection of these values.

**Advantages:**

Both the SVM and ANN are giving better results compared with other machine learning algorithms. SVM is giving 97% accuracy on both train, test sets with linear kernel. Artificial Neural Network with three hidden dense layer of each contains 1000 nodes and relu as activation function, one input layer with 20 features and one output layer consists two nodes. In the output layer softmax is used as activation function and adam optimizer used then ANN is giving 98% accuracy.

* + 1. **Scope:**
  1. **SOFTWARE/ HARDWARE REQUIREMENTS**

**3.2.1 Software Requirements:**

* OS: Windows or Linux
* Python IDE : python 2.7.x and above
* Pycharm IDE Required, jupyter notebook
* Setup tools and pip to be installed for 3.6 and above
* Language : Python Scripting

**3.2.2 Hardware Requirements:**

• RAM: 4GB and Higher

• Processor: Intel i3 and above

• Hard Disk: 500GB: Minimum

* 1. **MODULES**

**3.3.1Supervised Classification (Training Dataset)**

The data has been divided into two parts i.e., training and testing data in the 70:30 ratios. Learning algorithms have been applied on the training data and based on the learning, predictions are made on the test data set.

A **voice frequency** (VF) is one of the frequency.

Gender Recognition by Voice and Speech Analysis

This database was created to identify a voice as male or female, based upon acoustic properties of the voice and speech. The dataset consists of 3,168 recorded voice samples, collected from male and female speakers. The voice samples are pre-processed by acoustic analysis in using the see wave and tune

## The Dataset

The following acoustic properties of each voice are measured and included within the CSV:

* **meanfreq**: mean frequency (in kHz)
* **sd**: standard deviation of frequency
* **median**: median frequency (in kHz)
* **Q25**: first quantile (in kHz)
* **Q75**: third quantile (in kHz)
* **IQR**: interquantile range (in kHz)
* **skew**: skewness (see note in specprop description)
* **kurt**: kurtosis (see note in specprop description)
* **sp.ent**: spectral entropy
* **sfm**: spectral flatness
* **mode**: mode frequency
* **centroid**: frequency centroid (see specprop)
* **peakf**: peak frequency (frequency with highest energy)
* **meanfun**: average of fundamental frequency measured across acoustic signal
* **minfun**: minimum fundamental frequency measured across acoustic signal
* **maxfun**: maximum fundamental frequency measured across acoustic signal
* **meandom**: average of dominant frequency measured across acoustic signal
* **mindom**: minimum of dominant frequency measured across acoustic signal
* **maxdom**: maximum of dominant frequency measured across acoustic signal
* **dfrange**: range of dominant frequency measured across acoustic signal
* **modindx**: modulation index.Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
* **label**: male or female

The Bayes point machine algorithm achieved an 89.5% average classification accuracy. After hyperparameter model tuning, the best performing multiclass logistic regression algorithm, random decision forest algorithm, fully-connected neural network, and support vector machine achieved an average predictive accuracy of 90.2%, 90.0%, 88.3%, and 90.6%, respectively.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Accuracy | | | | | |
|  | Training Set | | Testing Set | | |
|  |  | Accuracy | | F-Measure |  |
| 1 | The Bayes point machine algorithm | 89.532 | | 89.394 |  |
| 2 | logistic regression algorithm | 97.264 | | 97.727 |  |
| 3 | random decision forest algorithm | 99.832 | | 97.607 |  |
| 4 | Artificial neural network | 100 | | 98.385 |  |
| 5 | support vector machine | 97.42 | | 97.97 |  |

**3.3.2Supervised Classification (Test Dataset)**

The test dataset is 30% of the total data. Supervised learning algorithms have been applied on the test data and the output obtained is compared with the actual output.

A well trained algorithm can predict well for new test data.

**3.3.3 Artificial neural network module:**

Several hyper parameter on ANN algorithm with different batch sizes 10,20,32, different number of epochs 50,100,200 and with different optimizers are adam, rmsprop. Best parameter values are batch size 32, epochs 100 and optimizer rmsprop.

1. **Batch Size:-** It is indicative of number of patterns shown to the network before the weight matrix is updated. If batch size is less, patterns would be less repeating and hence the weights would be all over the place and convergence would become difficult. If batch size is high learning would become slow as only after many iterations will the batch size change. It is recommend to try out batch sizes in powers of 2 (for better memory optimization) based on the data-size.
2. **Number of Epochs**:- The number of epochs is the number of times the entire training data is shown to the model. It plays an important role in how well the model fit on the train data does. High number of epochs may over-fit to the data and may have generalization problems on the test and validation set, also they could cause vanishing and exploding gradient problems. Lower number of epochs may limit the potential of the model. Try different values based on the time and computational resources you have.

**RMSprop optimizer**

here are a myriad of hyper parameters that you could tune to improve the performance of your neural network. But, not all of them significantly affect the performance of the network. One parameter that could make the difference between your algorithm converging or exploding is the optimizer you choose. There are a considerable number of optimizers you could choose

**Adam optimization**

Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

The Adam is derived from adaptive moment estimation*.*

Model optimization is one of the toughest challenges in the implementation of machine learning solutions. Entire branches of machine learning and deep learning theory have been dedicated to the optimization of models. Typically, we think about model optimization as a process of regularly modifying the code of the model in order to minimize the testing error. However, deep learning optimization often entails fine tuning elements that live outside the model but that can heavily influence its behavior. Deep learning often refers to those hidden elements as hyper parameters as they are one of the most critical components of any machine learning application.Hyper parameters are settings that can be tuned to control the behavior of a machine learning algorithm. Conceptually, hyper parameters can be considered orthogonal to the learning model itself in the sense that, although they live outside the model, there is a direct relationship between them.

The criteria of what defines a hyperparameter is incredibly abstract and flexible. Sure, there are well established hyperparameters such as the number of hidden units or the learning rate of a model but there are also an arbitrarily number of settings that can play the role of hyperparameters for specific models. In general, hyperparameters are very specific to the type of machine learning model you are trying to optimize. Sometimes, a setting is modeled as a hyperparameter because is not appropriate to learn it from the training set. A classic example are settings that control the capacity of a model (the spectrum of functions that the model can represent). If a deep learning algorithm learns those settings directly from the training set, then it is likely to try to optimize for that dataset which will cause the model to overfit( poor generalization).

**3.3.4 SVM(Support vector machine):**

Applied parameter tuning on SVM using GridSeachCV with different kernels linear, rbf, poly, different gamma and C vales. Best parameter values are C 0.6, gamma 0.04 and kernel rbf. After applying parameter tuning SVM, ANN are giving improved results. Applied 0.1 dropout between hidden layers to avoid over fitting machine learning model.

Supervised learning is the machine learning task for deducing functions from a labeled training data that can be occupied for both classification and regression. Support vector machines are a binary classification algorithm. Support vectors are the data points adjacent to the hyper planesif the dataset isremoved it will change the position ofthe dividinghyper plane. In SVM, each example set is a pair having an input object and a preferred output value. Supervised learning algorithm analyses the data and result out the inferred function, which result in mapping new outcomes.

We can say, SVM is a machine learning tool used for classification, approximation, etc.., it can be used in a generalized manner that leads to the success in many fields. Metrics of SVM is to minimize and generalization of upper bound error upon maximizing the margin which is separating hyperplane and data set. Advantages of model selection by means of both optimal number as well as the location of functions are obtained automatically during training.

* 1. **FUNCTIONAL Requirements**

**3.4.1 Supervised Classification (Training Dataset)**

* The data has been divided into two parts
* training and testing data in the 70:30 ratios.
* Learning algorithms have been applied on the training data and based on the learning, predictions are made on the test data set.
* This database was created to identify a voice as male or female, based upon acoustic properties of the voice and speech.
* The dataset consists of 3,168 recorded voice samples, collected from male and female speakers.
* The voice samples are pre-processed by acoustic analysis in using the see wave and tune.
* The data set takes different acoustic properties of each voice are measured and included in voice frequency.

**3.4.2 Supervised Classification (Test Dataset)**

* The test dataset is 30% of the total data.
* Supervised learning algorithms have been applied on the test data and the output obtained is compared with the actual output.
* A well trained algorithm can predict well for new test data.
* The test data should be very clear and shoud not contain any errors then we can take test data into the supervised classification.
* This data is tested several times and taken to do analysis process.
* Then we will get a clear output when the test data is so clear.

**3.4.3 Artificial neural network module:**

* Several hyper parameter on ANN algorithm with different batch sizes 10,20,32, different number of epochs 50,100,200 and with different optimizers are adam, rmsprop.
* Best parameter values are batch size 32, epochs 100 and optimizer rmsprop.
* Batch Size:- It is indicative of number of patterns shown to the network before the weight matrix is updated.
* If batch size is less, patterns would be less repeating and hence the weights would be all over the place and convergence would become difficult.
* If batch size is high learning would become slow as only after many iterations will the batch size change.
* It is recommend to try out batch sizes in powers of 2 (for better memory optimization) based on the data-size.
* Number of Epochs:- The number of epochs is the number of times the entire training data is shown to the model.
* It plays an important role in how well does the model fit on the train data
* Hyperparameters are settings that can be tuned to control the behavior of a machine learning algorithm.
* hyperparameters can be considered orthogonal to the learning model itself in the sense that, although they live outside the model, there is a direct relationship between them.

**3.4.4 SVM (Support vector machine):**

* Applied parameter tuning on SVM using GridSeachCV with different kernels linear, rbf, poly,different gamma and C waves.
* Best parameter values are C 0.6, gamma 0.04 and kernal rbf. After applying parameter tuning SVM, ANN are giving improved results.
* Applied 0.1 drop out between hidden layer to avoid over fitting machine learning model
* Support vector machines are a binary classification algorithm.
* Support vectors are the data points adjacent to the hyperplanes if the dataset is removed.
* it will change the position of the dividing hyperplane.
* Supervised learning algorithm analyses the data and result out the inferred function, which result in mapping new outcomes.
* SVM is a machine learning tool used for classification, approximation, etc..,
* SVM is to minimize and generalization of upper bound error upon maximizing the margin which is separating hyperplaneand data set.

**NON Functional Requirements**

The major non-functional Requirements of the system are as follows:

**Usability:**

The system is designed with completely automated process hence there is no or less user intervention

**Reliability:**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

**Performance:**

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

**Supportability:**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

**Implementation:**

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows XP professional is used as the platform. Interface the user interface is based on Struts provides HTML Tag.

**3.5 FEASIBILITY STUDY:**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time.

There are aspects in the feasibility study portion of the preliminary investigation:

* Economical Feasibility
* Technical Feasibility
* Operational Feasibility

.

**3.5.1 Economic feasibility:**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**3.5.2 Technical feasibility:**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

To manage unstructured big data that does not fit into any database, special tolls are needed. To examine this type of big dataset, the IT sector uses the Hadoop platform for a wide variety of methods that have been developed to record, organize, and, analyze this type of data. More efficient tools are needed to extract meaningful output from big data. Most of the tools are implemented in the Apache Hadoop architecture including Map Reduce, Mahout, Hive, and others.

**3.5.3 Operational feasibility:**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation.

Some of the important issues raised are to test the operational feasibility of a

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Modern big data technologies make it possible in a short time to analyze a large collection of data from thousands of patients, identify clusters and correlations, and develop predictive models using statistical or machine-learning modeling techniques. In this new context it would be feasible to take all the data collected in all past epidemiology studies - for example, those used to develop FRAX – and continue to enrich them with new studies where not only new patients are added, but different types of information are collected.

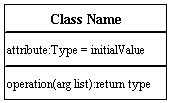
**4.Design**

**4.2.1Class diagram**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

**Basic Class Diagram Symbols and Notations:**

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and  
Write operations into the third.



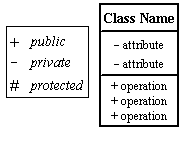
**Active Class:**

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



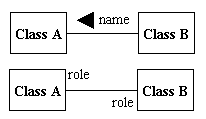
**Visibility:**

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class. [.](http://www.smartdraw.com/resources/tutorials/Text-and-Tables)



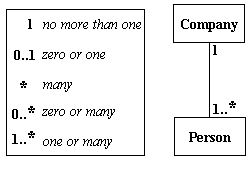
**Associations:**

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.  
***Note:***It's uncommon to name both the association and the class roles.



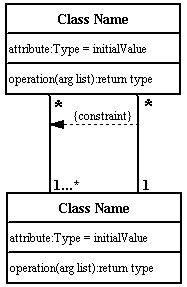
**Multiplicity (Cardinality):**

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.



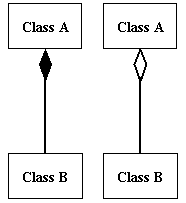
**Constraint:**

Place constraints inside curly braces {}.

http://wc1.smartdraw.com/resources/tutorials/images/uml_constraint.gif*Simple Constraint*

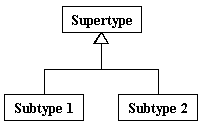
**Composition and Aggregation:**

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustratecomposition with a filled diamond. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



**Generalization:**

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

**Figure: Class Diagram for Sentiment Analysis to predict gender classification**

**4.2.2 Use case Diagram:**

The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system. Create a professional diagram for nearly any use case using our UML diagram tool.

Top of Form

Bottom of Form

**What is a use case diagram**

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. Scenarios in which your system or application interacts with people, organizations, or external systems

* Goals that your system or application helps those entities (known as actors) achieve
* The scope of your system

**When to apply use case diagrams**

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

* Representing the goals of system-user interactions
* Defining and organizing functional requirements in a system
* Specifying the context and requirements of a system
* Modeling the basic flow of events in a use case

Diagramming is quick and easy with Lucid chart. Start a free trial today to start creating and collaborating.

Top of Form

Bottom of Form

**Use case diagram components:**

Common components include:

* **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
* **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
* **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

**Use case diagram symbols and notation:**

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams. Here are all the shapes you will be able to find in Lucid chart:

* **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
* **Actors:** Stick figures that represent the people actually employing the use cases.
* **Primary Actors:** The Actor(s) using the system to achieve a goal. The Use Case documents the interactions between the system and the actors to achieve the goal of the primary actor.
* **Secondary Actors:** Actors that the system needs assistance from to achieve the primary actor’s goal. Secondary actors may or may not have goals that they expect to be satisfied by the use case, the primary actor always has a goal, and the use case exists to satisfy the primary actor.
* **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.

**System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below

****

**Figure:Use Case Diagram Sentiment Analysis to Predict gender classification**

**4.2.3 Sequence Diagram:**

UML Sequence diagrams are interaction diagrams that detail how operations are carried out. As sequence diagrams can be used to capture the interaction between objects in the context of collaboration, one of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

**Sequence Diagrams captures interaction in different level of granularity:**

* High-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams).
* The interaction that takes place in collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams).
* Represent objects interact in (Model, View / Controller) MVC pattern of software framework.

## Sequence Diagram Notations:

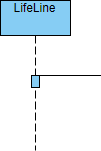
### Lifeline:

A lifeline represents an individual participant in the Interaction.



### Activation:

An activation is represented by a thin rectangle on a lifeline) represents the period during which an element is performing an operation. The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively



### Messages:

#### Call Message:

A call message defines a particular communication between lifelines of an interaction, which represents an invocation of operation of target lifeline.

UML Sequence Diagram: Call message example

#### Return Message:

A return message defines a particular communication between lifelines of an interaction, which represents the pass of information back to the caller of a corresponded former message.

UML Sequence Diagram: Return message example

#### Self Message:

A self message defines a particular communication between lifelines of an interaction, which represents the invocation of message of the same lifeline.

UML Sequence Diagram: Self message example

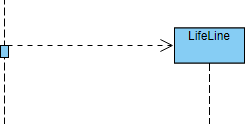
#### Recursive Message:

A recursive message defines a particular communication between lifelines of an interaction, which represents the invocation of message of the same lifeline. Its target points to an activation on top of the activation where the message was invoked from.

UML Sequence Diagram: Recursive message example

#### Create Message:

A create message defines a particular communication between lifelines of an interaction, which represents the instantiation of (target) lifeline.



## When to Draw Sequence Diagram:

* Model high-level interaction between active objects in a system
* Model the interaction between object instances within a collaboration that realizes a use case
* Model the interaction between objects within a collaboration that realizes an operation
* Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

**How to Draw a Sequence Diagram:**

1. Identify a set of objects that will participate in the general collaboration (or use case scenario)
   * If you derive the sequence diagram based on a scenario of a use case, select the normal scenarios first
   * You should know the primary actor(s) who activates the use case
2. Consider the first point of the scenario (or if you get it from the first point of the flow of event of a use case)
3. Consider what the system need to be done in order to response to the actor, when the actor send the message to the system
   * What the system need to be handled before the return message response back from the system?
   * E.g. A customer inserted an ATM card to the machine, the system will display "input pin number" in the normal scenario, right?
   * Guess, what will to be handled inside the ADM by a set of objects at the "back" of the system? Something like, read and verify the ATM card (card reader), read the card information of the card holder (by the bank) and ask for the pin, or, return "invalid card type, insert another card", and etc.
   * By this way, you will identify the candidate objects and operations of the target application for that particular scenario and you can also use this information as a basis to derive the class diagram incrementally.
4. Repeat each of the point of the scenario (or flow of event) and until you complete all the points in the scenario.



**Figure: Sequence Diagram Sentiment Analysis to predict gender classification**

**4.2.4 Activity Diagram:**

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

**Purpose of Activity Diagrams:**

The basic purpose of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

**The purpose of an activity diagram can be described as** −

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.

**How to Draw an Activity Diagram:**

Activity diagrams are mainly used as a flowchart that consists of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlanes, etc. Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

Before drawing an activity diagram, we should identify the following elements −

* Activities
* Association
* Conditions
* Constraints

Once the above-mentioned parameters are identified, we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram.

### Activity Diagram Notations:

**(1).Initial State –** The starting state before an activity takes place is depicted using the initial state.

****

A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

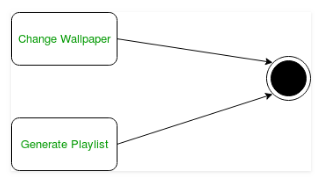
**(2).Action or Activity State –** An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

UML-Activity-Diagram

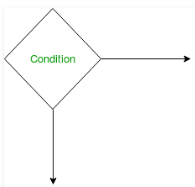
**(3).Action Flow or Control flows –** Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.



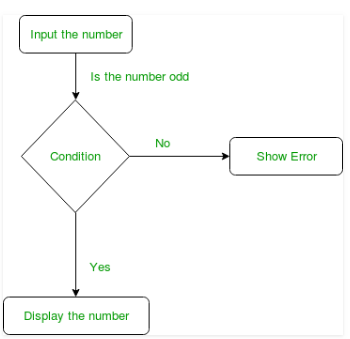
An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow. Consider the example – Here both the states transit into one final state using action flow symbols i.e. arrows.



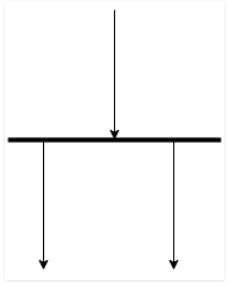
**(4).Decision node and Branching –** When we need to make a decision before deciding the flow of control, we use the decision node.



The outgoing arrows from the decision node can be labelled with conditions or guard expressions.It always includes two or more output arrows.

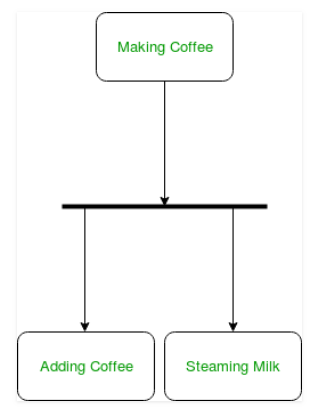


**(5).Fork –** Fork nodes are used to support concurrent activities.

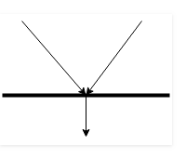


**Figure –** fork notation when we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards then newly created activities.

**For example:** In the example below, the activity of making coffee can be split into two concurrent activities and hence we use the fork notation.

  
 **Figure –** a diagram using fork

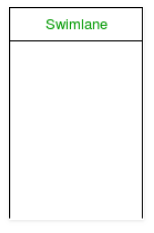
**(6).Join –** Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.



**For example** – When both activities i.e. steaming the milk and adding coffee get completed, we converge them into one final activity.

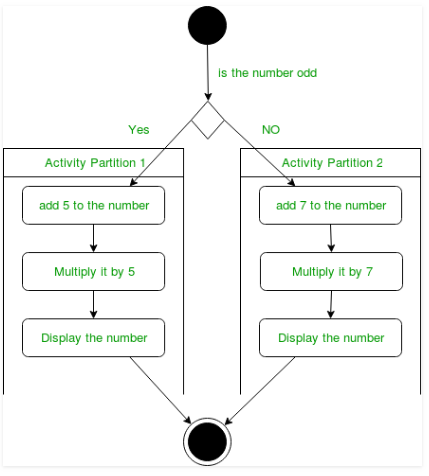


**Swimlanes –** We use swimlanes for grouping related activities in one column. Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add modularity to the activity diagram. It is not mandatory to use swimlanes. They usually give more clarity to the activity diagram. It’s similar to creating a function in a program. It’s not mandatory to do so, but, it is a recommended practice.



**Figure–** swimlanes notation We use a rectangular column to represent a swimlane as shown in the figure above.

**For example** – Here different set of activities are executed based on if the number is odd or even. These activities are grouped into a swimlane.

  
 **Figure –** an activity diagram making use of swimlanes

**Final State or End State –** The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.





**Figure:Activity Diagram Sentiment Analysis to predictgender classification**

**4.2.5 Collaboration Diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**Basic Collaboration Diagram Symbols and Notations**

**Class roles**

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Class roles

**Association roles**

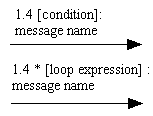
Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.

Association roles

**Messages**

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The a condition for a message is usually placed in square brackets immediately following the sequence number. Use a \* after the sequence number to indicate a loop.

Learn how to add arrows to your lines.



****

**Figure: Collaboration Diagram Sentiment Analysis to predict gender classification**

**4.2.6 Component diagram**

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.

****

**Figure: Component Diagram Sentiment Analysis to predict gender classification**

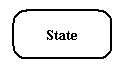
**4.2.7 State chart Diagram**

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

**Basic State chart Diagram Symbols and Notations**

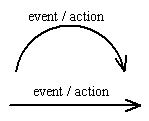
**States**

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



**Transition**

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.



**Initial State**

A filled circle followed by an arrow represents the object's initial state.

Initial State

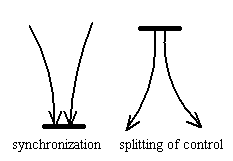
**Final State**

An arrow pointing to a filled circle nested inside another circle represents the object's final state.

Final State

**Synchronization and Splitting of Control**

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



****

**Figure: State chart Diagram Sentiment Analysis to Predict Election Result**

**CODING**

**Sample coding:**

import numpy as np

# In[7]:

import pandas as pd

# In[8]:

import warnings

warnings.filterwarnings('ignore')

# In[9]:

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import Imputer

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import OneHotEncoder

from sklearn.model\_selection import GridSearchCV

from sklearn.pipeline import Pipeline

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

from sklearn.svm import SVC

from matplotlib import pyplot as plt

get\_ipython().run\_line\_magic('matplotlib', 'inline')

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import cross\_val\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble.forest import RandomForestClassifier

# In[10]:

import keras

from keras.models import Sequential

from keras.layers import Dense

# In[ ]:

# In[11]:

from sklearn.model\_selection import StratifiedKFold, cross\_val\_score

from sklearn.metrics import classification\_report

import multiprocessing

from keras.utils import np\_utils

from keras.wrappers.scikit\_learn import KerasClassifier

# In[12]:

from numpy import array

from numpy import argmax

# In[13]:

voice\_ds=pd.read\_csv("voice.csv")

# In[14]:

voice\_ds.head()

# # converting categorical to integers

# In[15]:

dv=array(voice\_ds.label.values)

label\_encoder=LabelEncoder()

int\_val=label\_encoder.fit\_transform(dv)

onehot\_encoder = OneHotEncoder(sparse=False)

integer\_encoded = int\_val.reshape(len(int\_val), 1)

onehot\_encoded = onehot\_encoder.fit\_transform(integer\_encoded)

voice\_ds.label=onehot\_encoded

voice\_ds.head()

# # treating with missing values

# In[16]:

columns=list(voice\_ds.columns.values)

columns.remove('label')

# In[17]:

(voice\_ds[columns]==0).sum()

# In[18]:

voice\_ds[columns]=voice\_ds[columns].replace(0,np.NaN)

# In[19]:

voice\_ds.fillna(voice\_ds.mean(),inplace=True)

# In[20]:

voice\_ds.mean()

# In[21]:

(voice\_ds[columns]==0).sum()

# In[22]:

y=voice\_ds.loc[:,'label']

X = voice\_ds.loc[:,'meanfreq':'modindx']

# In[23]:

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.25,random\_state=0)

sc=StandardScaler()

X\_train=sc.fit\_transform(X\_train)

X\_test=sc.transform(X\_test)

# In[24]:

voice\_ds.describe()

# In[25]:

voice\_ds.corr()

# In[26]:

from sklearn import linear\_model

logreg = linear\_model.LogisticRegression()

logistic = logreg.fit(X\_train,y\_train)

print("Logistic Regression Accuracy : {}".format(logreg.score(X\_test,y\_test)))

# In[27]:

dt = DecisionTreeClassifier()

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

print ("Using Decision tree the Accuracy score is ",dt.score(X\_test, y\_test))

# In[28]:

rf = RandomForestClassifier(n\_estimators=100)

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

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

print("Accuracy:",rf.score(X\_test, y\_test))

# In[29]:

svm = SVC()

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

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

print("Accuracy:",rf.score(X\_test, y\_test))

# In[30]:

classifier = Sequential()

# Adding the input layer and the first hidden layer

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu', input\_dim = 20))

# Adding the second hidden layer

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu'))

# Adding the output layer

classifier.add(Dense(units = 1, kernel\_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Fitting the ANN to the Training set

#classifier.fit(X\_train, y\_train, batch\_size = 10, epochs = 100)

classifier.fit(X\_train, y\_train, batch\_size = 10, epochs = 20)

# Part 3 - Making predictions and evaluating the model

# Predicting the Test set results

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

y\_pred = (y\_pred > 0.5)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

# In[31]:

cm

# # below steps took too much time don't execute

#

# In[ ]:

svm = SVC()

parameters = {'kernel':('linear', 'rbf','poly'), 'C':(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1),'gamma': (0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,1,10,1000)}

clf = GridSearchCV(svm, parameters,cv=10)

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

print("accuracy:"+str(cross\_val\_score(clf, X\_train, y\_train, scoring='accuracy')))

print(clf.best\_params\_)

print("Test Accuracy:",clf.score(X\_test,y\_test))

# In[ ]:

from sklearn.metrics import confusion\_matrix

classifier = Sequential()

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'reel', input\_dim = 20))

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu'))

classifier.add(Dense(units = 1, kernel\_initializer = 'uniform', activation = 'sigmoid'))

seed = 7

np.random.seed(seed)

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

batch\_size = [32]

epochs = [200]

cm=[]

score=[]

for bs,epochs in zip(batch\_size,epochs):

classifier.fit(X\_train, y\_train, batch\_size = bs, epochs = epochs)

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

y\_pred = (y\_pred > 0.5)

cm.append(confusion\_matrix(y\_test, y\_pred))

score.append(accuracy\_score(y\_test, y\_pred))

# In[ ]:

print(score)

# In[ ]:

# Use scikit-learn to grid search the batch size and epochs

import numpy

from sklearn.model\_selection import GridSearchCV

from keras.models import Sequential

from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasClassifier

# Function to create model, required for KerasClassifier

def create\_model():

# create model

model = Sequential()

model.add(Dense(12, input\_dim=20, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# Compile model

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

model = KerasClassifier(build\_fn=create\_model, verbose=0)

# define the grid search parameters

batch\_size = [10, 20, 32]

epochs = [50, 100,200]

param\_grid = dict(batch\_size=batch\_size, epochs=epochs)

grid = GridSearchCV(estimator=model, param\_grid=param\_grid, n\_jobs=-1)

grid\_result = grid.fit(X, y)

# summarize results

print("Best: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))

means = grid\_result.cv\_results\_['mean\_test\_score']

stds = grid\_result.cv\_results\_['std\_test\_score']

params = grid\_result.cv\_results\_['params']

for mean, stdev, param in zip(means, stds, params):

print("%f (%f) with: %r" % (mean, stdev, param))

print("%f (%f) with: %r" % (mean, stdev, param))

# In[6]:

def create\_model():

activation='relu'

dropout\_rate=0.0

init\_mode='uniform'

weight\_constraint=0

optimizer='adam'

learn\_rate=0.01

momemntum=0

model = Sequential()

model.add(Dense(12, input\_dim=20,kernal\_initializer= init\_mode, activation=activation,kernal\_contraint=maxnorm(weight\_constraint)))

model.add(Dropout(dropout\_rate))

model.add(Dense(1,kernal\_initializer= init\_mode, activation='sigmoid'))

# Compile model

model.compile(loss='binary\_crossentropy', optimizer=optimizer, metrics=['accuracy'])

return model

# In[7]:

model = KerasClassifier(build\_fn=create\_model, batch\_size=10,epochs=10)

# In[ ]:

optimizer=['Adam']

dropout\_rate=[0.2]

epochs=[100]

batch\_size=[32]

param\_grid=dict(epochs=epochs,batch\_size=batch\_size)

grid = GridSearchCV(estimator=model, param\_grid=param\_grid, n\_jobs=-1)

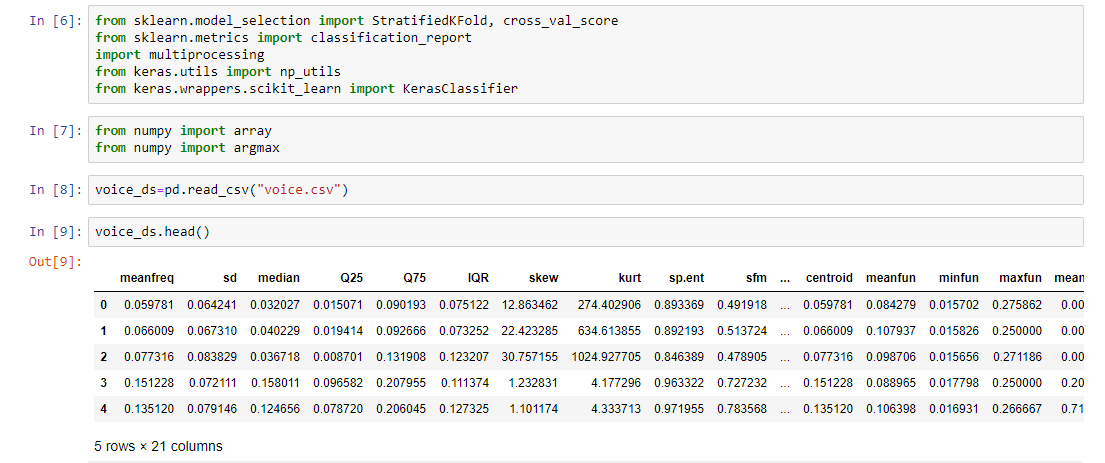
grid\_result = grid.fit(X, y)

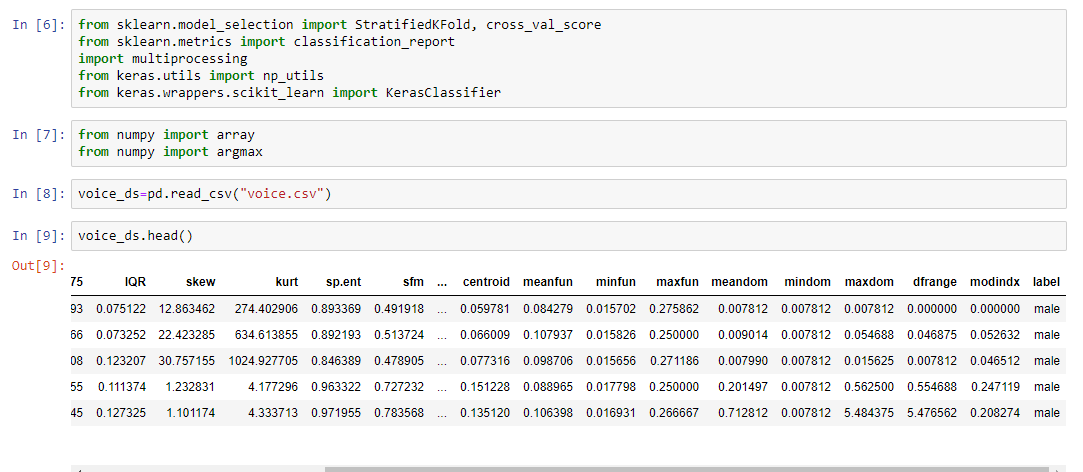
# In[ ]:

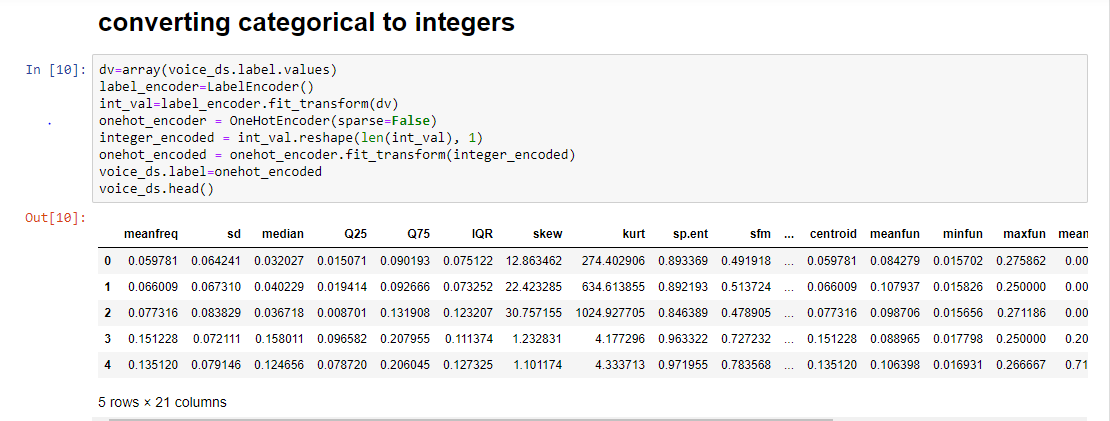
# In[ ]:

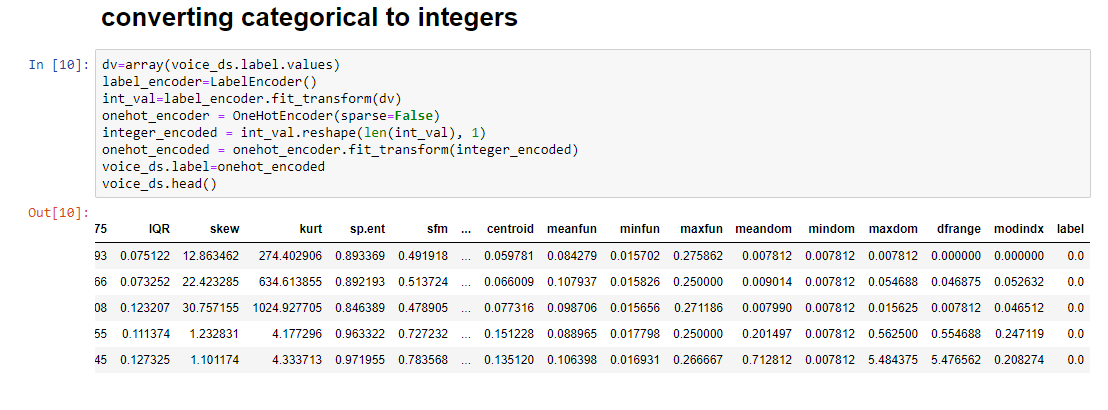
# In[ ]:

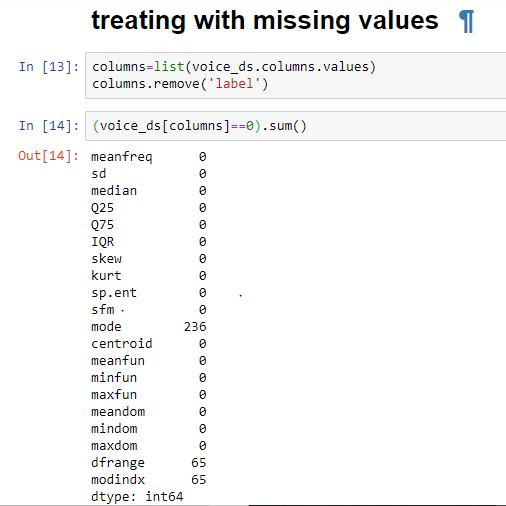
**SCREEN SHOTS**

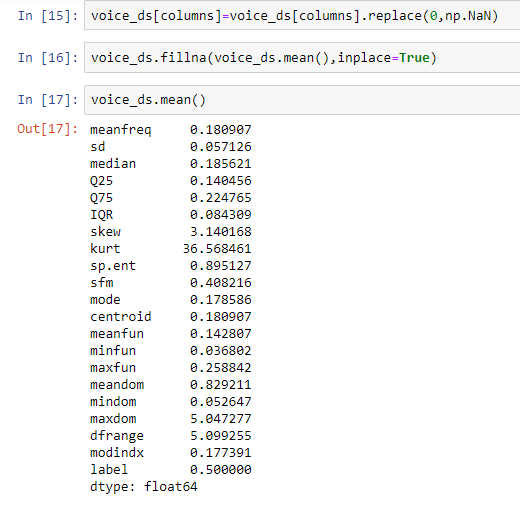


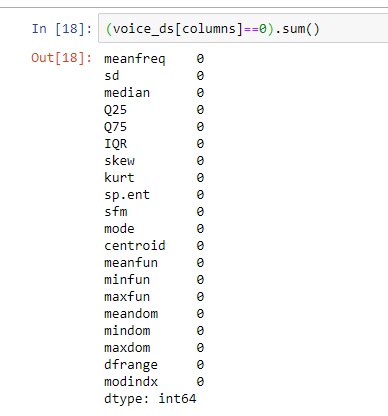


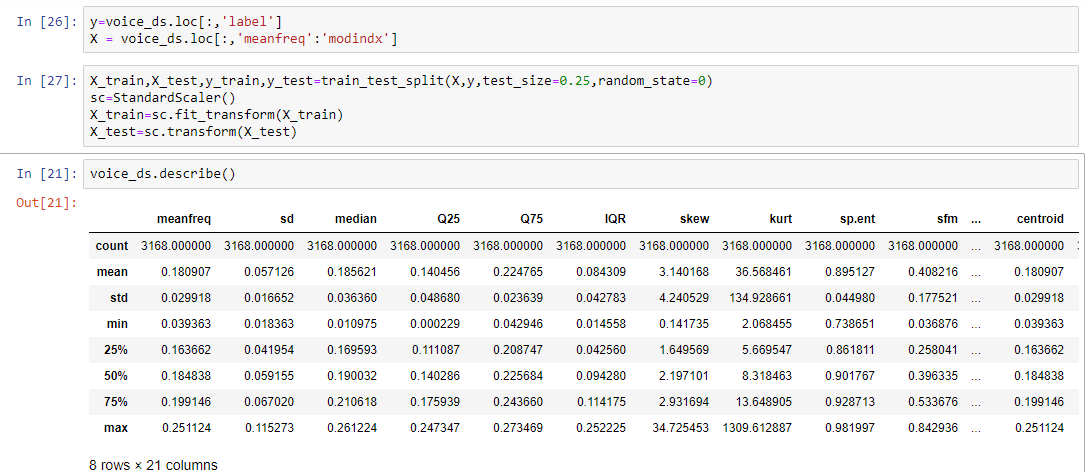




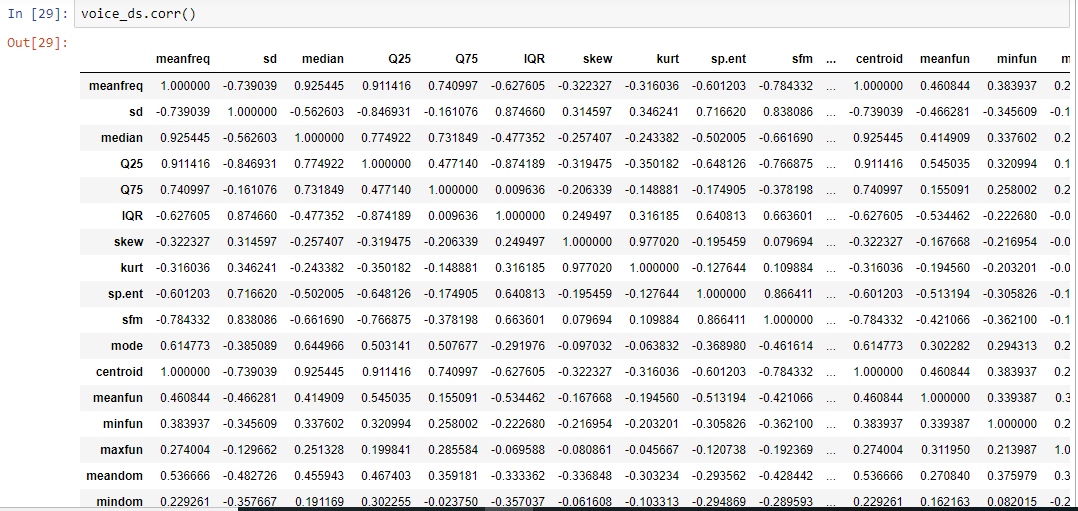


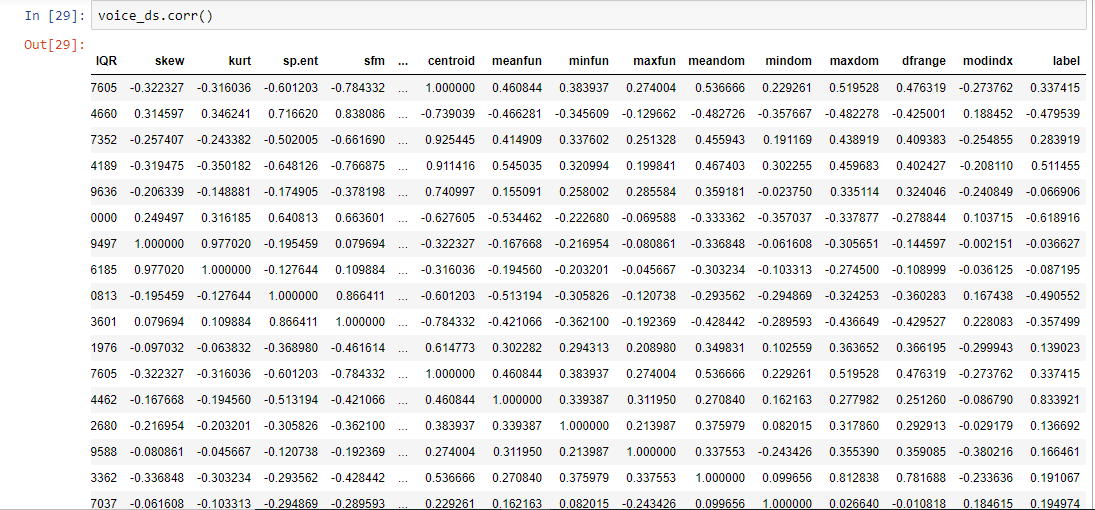


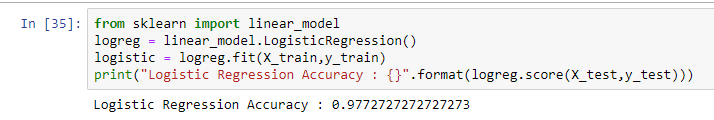


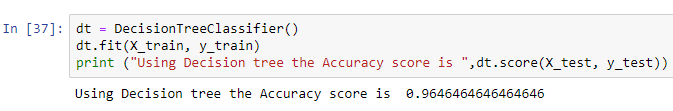


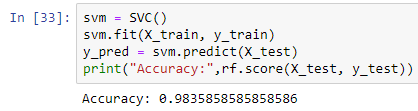


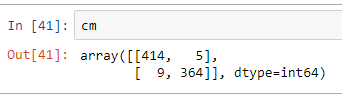


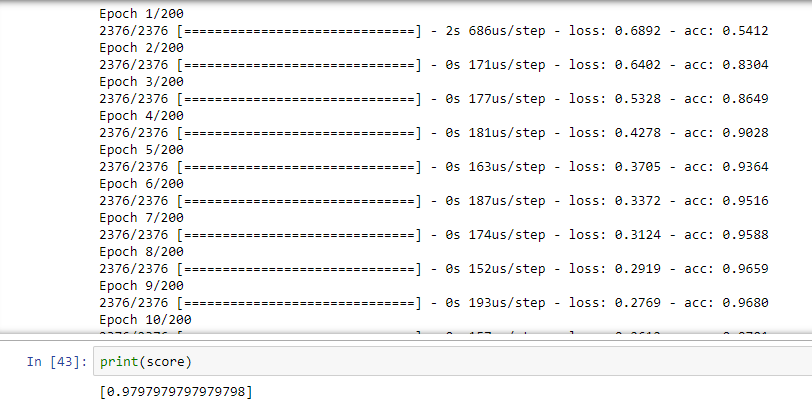












**TESTING**

**INTRODUCTION**

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior implemented system. From modeling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

The process of executing a program with intent of finding errors is called testing. During testing , the program to be tested is executed with a set of test cases , and the output of the program for the test cases is evaluated to determine if the program is performing as expected . Testing forms the first step in determining the errors in the program. The success of testing in revealing errors in program depends critically on test cases.

**STRATEGIC APPROACH TO SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirements analysis where the information domain , functions, behavior, performance ,constraints and validation criteria for software are established. moving inward along the spiral , we come to design and finally to coding . To develop computer software we spiral in along streamlines that decreases the level of abstraction on each item.

A Strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing will progress by moving outward along the spiral to integration testing , where the focus on the design and the concentration of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed . Finally we arrive at system testing , where the software and other system elements are tested as a whole .

UNUNI

UNIT TESTING

MODULE

SUB-SYSTEM

**Component**

SYSTEM TESTING

**Integration Testing**

ACCEPTANCE

**User Testing**

**Different Levels of Testing**

Client Needs Acceptance Testing

Requirements System Testing

Design Integration Testing

Code Unit Testing

Testing is the process of finding difference between the expected behavior specified by system models and the observed behavior of the implemented system.

**Testing Activities**

Different levels of testing are used in the testing process , each level of testing aims to test different aspects of the system. the basic levels are:

Unit testing

Integration testing

System testing

Acceptance testing

**Unit Testing**

Unit testing focuses on the building blocks of the software system, that is, objects and sub system . There are three motivations behind focusing on components. First, unit testing reduces the complexity of the overall tests activities, allowing us to focus on smaller units of the system. Second , unit testing makes it easier to pinpoint and correct faults given that few components are involved in this test . Third , Unit testing allows parallelism in the testing activities , that is each component can be tested independently of one another . Hence the goal is to test the internal logic of the module.

**Integration Testing**

In the integration testing, many test modules are combined into sub systems , which are then tested . The goal here is to see if the modules can be integrated properly, the emphasis being on testing module interaction.

After structural testing and functional testing we get error free modules. These modules are to be integrated to get the required results of the system. After checking a module, another module is tested and is integrated with the previous module. After the integration, the test cases are generated and the results are tested.

**System Testing**

In system testing the entire software is tested . The reference document for this process is the requirement document and the goal is to see whether the software meets its requirements. The system was tested for various test cases with various inputs.

**Acceptance Testing**

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactory. Testing here focus on the external behavior of the system , the internal logic of the program is not emphasized . In acceptance testing the system is tested for various inputs.

**Types of Testing**

1. Black box or functional testing
2. White box testing or structural testing

**Black box testing**

This method is used when knowledge of the specified function that a product has been designed to perform is known . The concept of black box is used to represent a system whose inside workings are not available to inspection . In a black box the test item is a "Black" , since its logic is unknown , all that is known is what goes in and what comes out , or the input and output.

Black box testing attempts to find errors in the following categories:

Incorrect or missing functions

Interface errors

Errors in data structure

Performance errors

Initialization and termination errors

As shown in the following figure of Black box testing , we are not thinking of the internal workings , just we think about

What is the output to our system?

What is the output for given input to our system?

**?**

Input Output

The Black box is an imaginary box that hides its internal workings

**White box testing**

White box testing is concerned with testing the implementation of the program. the intent of structural is not to exercise all the inputs or outputs but to exercise the different programming and data structure used in the program. Thus structural testing aims to achieve test cases that will force the desire coverage of different structures . Two types of path testing are statement testing coverage and branch testing coverage.

**INTERNAL WORKING**

Input Output

The White Box testing strategy , the internal workings

**Test Plan**

Testing process starts with a test plan. This plan identifies all the testing related activities that must be performed and specifies the schedules , allocates the resources , and specified guidelines for testing . During the testing of the unit the specified test cases are executed and the actual result compared with expected output. The final output of the testing phase is the test report and the error report.

**Test Data:**

Here all test cases that are used for the system testing are specified. The goal is to test the different functional requirements specified in Software Requirements Specifications (SRS) document.

**Unit Testing:**

Each individual module has been tested against the requirement with some test data.

**Test Report:**

The module is working properly provided the user has to enter information. All data entry forms have tested with specified test cases and all data entry forms are working properly.

**Error Report:**

If the user does not enter data in specified order then the user will be prompted with error messages. Error handling was done to handle the expected and unexpected errors.

**SCOPE**

**Scope**

Our project performance comparison gender classification machine learning approach is aimed at classifying the male female from speech recognition dataset. For this we have used machine learning algorithms and checked theperformance comparison between the algorithms. For the enhancement of this project we felt there is a scope to try it with neural network approaches artificial neural networks which can increase the performance of the model further mode. It can give the best results when we have a large dataset.

**CONCLUSION**

**Conclusion**

Support-vector machines and Deep neural networks are performing better on voice dataset. Parameter Tuning is giving the 98.6% accuracy with SVM and 99.87% with ANN. From the above results we can conclude that deep neural networks are performing better compared with all machine learning algorithms to classify gender of a person using acoustic properties of voice.

**BIBLIOGRAPHY**

**BIBLIOGRAPHY**

**For software installation:**

<https://www.anaconda.com/download/>

<https://www.python.org/downloads/release/python-360/>

**Modules:**

* Install numpy
* Install pandas
* Install matplotlib
* Install scikit – learn

**References:**

S. Gaikwad, B. Gawali, and S. C. Mehrotra, “Gender identification using SVM with combination of MFCC,” Advances in Computational Research, vol. 4, no. 1, pp. 69–73, 2012.

[2] S. M. R. Azghadi, M. R. Bonyadi, and H. Sliahhosseini, “Gender classification based on feedforward backpropagation neural network,” in Artificial Intelligence and Innovations 2007: From Theory to Applications: Proceedings of the 4th IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI 2007), C. Boukis, L. Pnevmatikakis, and L. Polymenakos, Eds., vol. 247 of IFIP The International Federation for Information Processing, pp. 299–304, Springer, Berlin, Germany, 2007.

[3] Cortes, Corinna & Vapnik, VN. (1995). Support Vector Networks. Machine Learning. 20. 273-297. 10.1023/A:1022627411411.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, CRC Press, 1984.

[5] K.-H. Lee, S.-I. Kang, D.-H. Kim et al., “A Support Vector Machine- Based Gender Identification Using Speech Signal,” IEICE Transactions on Communications, pp. 3326-3329, 2008.

[6] J. Silovsky, and J. Nouza, “Speech, Speaker and Speaker's Gender Identification in Automatically Processed Broadcast Stream,”

[7] Büyükyılmaz, Mücahit & Çıbıkdiken, Ali. (2016). Voice Gender Recognition Using Deep Learning. 10.2991/msota-16.2016.90.

[8] <https://github.com/primaryobjects/voice-gender>

[9] M. Araya-Salas, G. Smith-Vidaurre, warbleR: an R package to streamline analysis of animal acoustic signals. Methods Ecol Evolution, 2016, doi:10.1111/2041-210X.12624.

[10] dataset, https://github.com/primaryobjects/voice- gender/blob/master/voice.csv

[11] http://neupy.com/2016/12/17/hyperparameter\_optimization\_for\_neural\_ networks.html

[12] G. S. Archana and M. Malleswari, "Gender identification and performance analysis of speech signals," 2015 Global Conference on Communication Technologies (GCCT), Thuckalay, 2015, pp. 483-489.

[13] M. A. Keyvanrad and M. M. Homayounpour, "Improvement on automatic speaker gender identification using classifier fusion," 2010 18th Iranian Conference on Electrical Engineering, Isfahan, Iran, 2010, pp. 538-541.