# Machine Learning Engineer Nanodegree

# Capstone Proposal

Gizem Burcu Çınar Pala
April, 2021

## Gender Recognition Based on Voice

## 1 Definition

### 1.1 Overview

The gender of a speaker, which is the most distinctive characteristics of a speech, can easily be recognized by a person who hears it. Automatic identification of gender information from speech signal is substantially important for many applications. With the identification of gender, gender-dependent systems are defined and accuracy and robustness of systems can be increased.

In this study, a system identifying the gender of a speaker independent from a voice is developed. The proposed system is based on the classification of MFCC coefficients obtained from speech signals with GMM. In the study, the effect of Gaussian mixture number and MFCC coefficients to the system success is investigated. Also K-means unsupervised classification method is compared with GMM.

### 1.2 Problem Statement

The applications of automatic gender detection 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.

Automatic gender detection system based on different types of feature and classifier with varying accuracies are reported in the literature. The acoustic characteristics of humans are based on gender due to physiological changes in glottis, vocal tract thickness, and length. Therefore, researchers are trying to find out the most discriminative features for gender detection. For example, two acoustic

features, pitch and first formant, are extracted by linear predictive analysis to construct a gender detection system. The first feature relates to voice source and the second to the vocal tract. The pitch and the formant frequencies of females are higher as compared to those of males.

In this project creating a gender recognition model that works with a high success rate is the targeted solution.

## 1.2 Metrics

**MFCC :** Mel Frequency Cepstral Coefficents (MFCCs) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. Prior to the introduction of MFCCs, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) and were the main feature type for automatic speech recognition (ASR).

The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear.

The formula for converting from frequency to Mel scale is:

$$M(f) = 1125 \ln(1 + f/700)$$

**Deltas and Delta-Deltas**: Also known as differential and acceleration coefficients. The MFCC feature vector describes only the power spectral envelope of a single frame, but it seems like speech would also have information in the dynamics i.e. what are the trajectories of the MFCC coefficients over time. It turns out that calculating the MFCC trajectories and appending them to the original feature vector increases ASR performance by quite a bit (if we have 12 MFCC coefficients, we would also get 12 delta coefficients, which would combine to give a feature vector of length 24).

To calculate the delta coefficients, the following formula is used:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

where d(t) is a delta coefficient, from frame t computed in terms of the static coefficients c(t)+N to c(t)-N . A typical value for N is 2. Delta-Delta (Acceleration) coefficients are calculated in the same way, but they are calculated from the deltas, not the static coefficients.

**The computation of the scores**:

- H(f) : speaker is a Female
- H(m) : speaker is a Male

$$\frac{p(Y|H_f)}{p(Y|H_m)} = \begin{cases} \geq 1 & \text{accept } H_f \\ < 1 & \text{reject } H_m \end{cases}$$

Figure: Y is a speaker

where p(Y|H(i)), is the probability density function for the hypothesis H(i) evaluated for the observed speech segment Y, also called the likelihood of the hypothesis H(i) given the speech segment Y

# 2 Analysis

## 2.1 Data Exploration and Visiualization

While developing the voice recognition system based on gender, **The Free ST American English Corpus dataset (SLR45) which** can be found on SLR45 (http://www.openslr.org/45/) is used. It is a free American English corpus by Surfingtech, containing utterances from 10 speakers (5 females and 5 males). Totally 3842(2186 female/1656 male) voice sample is invatigated. 1/3 of dataset is used for testing. 2/3 of dataset is used for training.

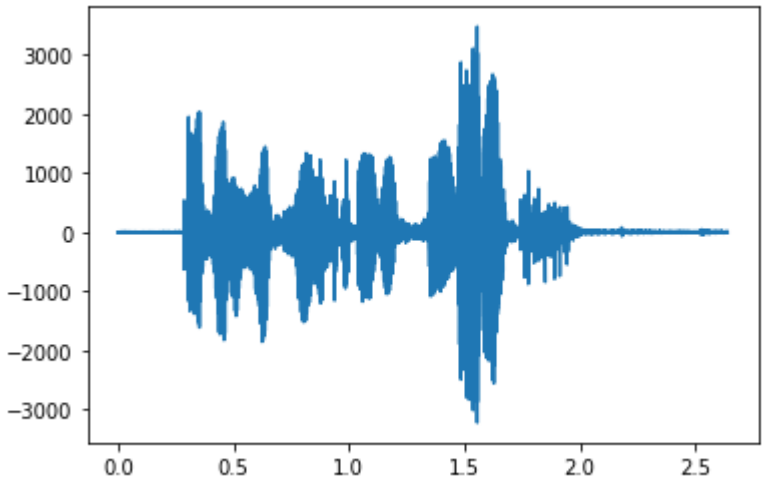



Fig: Voice sample

## 2.2 Algorithms and Techniques

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

In this project for extracting features of voice sample MFCC method is used.

**Steps of Extracting Features by MFCC**

Implementation steps of MFCC are given below

**1)** Frame the signal into short frames.

**2)** For each frame calculate the periodogram estimate of the power spectrum.

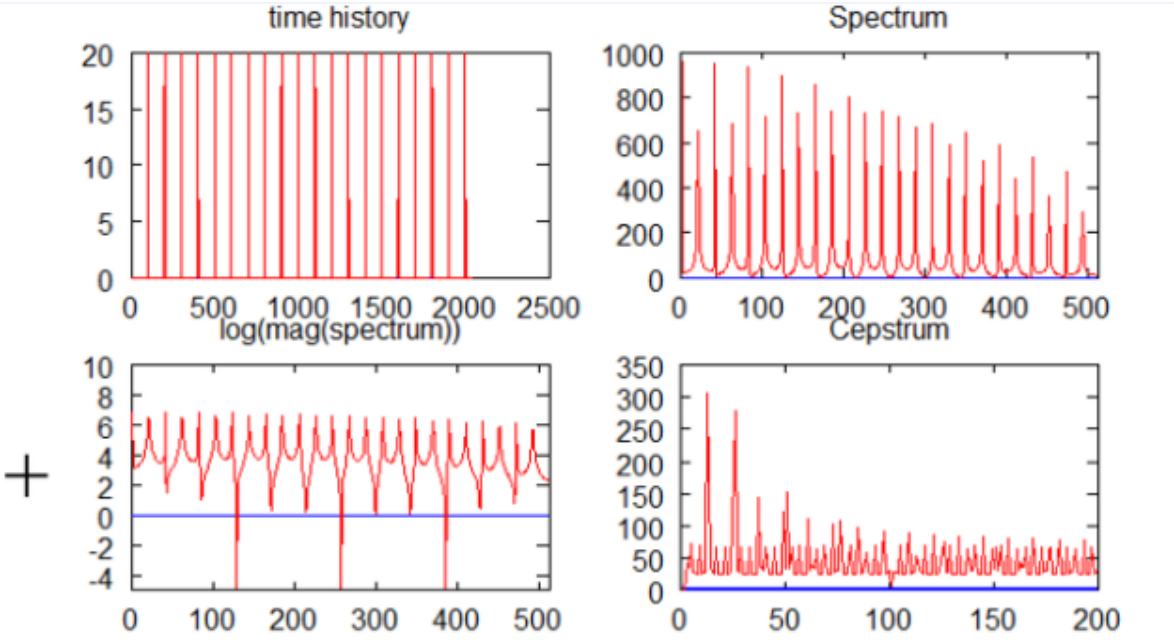
**3)** Apply the mel filterbank to the power spectra, sum the energy in each filter.

**4)** Take the logarithm of all filterbank energies.

**5)** Take the DCT of the log filterbank energies.

**6)** Keep DCT coefficients 2-13, discard the rest.

**7)** There are a few more things commonly done, sometimes the frame energy is appended to each feature vector. Delta and Delta-Delta features are usually also appended.




**Why do we do these things?**

We will now go a little more slowly through the steps and explain why each of the steps is necessary.

- An audio signal is constantly changing, so to simplify things we assume that on short time scales the audio signal doesn't change much. This is why we frame the signal into 20-40ms frames. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

- The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain

frequencies are present. Our periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

- The periodogram spectral estimate still contains a lot of information not required for Automatic Speech Recognition (ASR). In particular the cochlea can not discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by our Mel filterbank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations. We are only interested in roughly how much energy occurs at each spot. The Mel scale tells us exactly how to space our filterbanks and how wide to make them. See below for how to calculate the spacing.

- Once we have the filterbank energies, we take the logarithm of them. This is also motivated by human hearing: we don't hear loudness on a linear scale. Generally to double the percieved volume of a sound we need to put 8 times as much energy into it. This means that large variations in energy may not sound all that different if the sound is loud to begin with. This compression operation makes our features match more closely what humans actually hear. Why the logarithm and not a cube root? The logarithm allows us to use cepstral mean subtraction, which is a channel normalisation technique.

- The final step is to compute the DCT of the log filterbank energies. There are 2 main reasons this is performed. Because our filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features in e.g. a HMM classifier. But notice that only 12 of the 26 DCT coefficients are kept. This is because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade ASR performance, so we get a small improvement by dropping them.

Second step is creating a model and classifing them with an appropriate algorithm. This is an unsupervised problem that's why GMM algorithm is chosen to solve this problem.

**Gaussian Mixture Model**

According to D. Reynolds in Gaussian_Mixture_Models:

*<< A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori(MAP) estimation from a well-trained prior model. >>*

In a some way, you can consider a Gaussian mixture model as a probabilistic clustering representing a certain data distribution as a sum of Gaussian density functions. These densities forming a GMM are also called the components of the GMM. The likelihood of data points (feature vectors) for a model is given by following equation

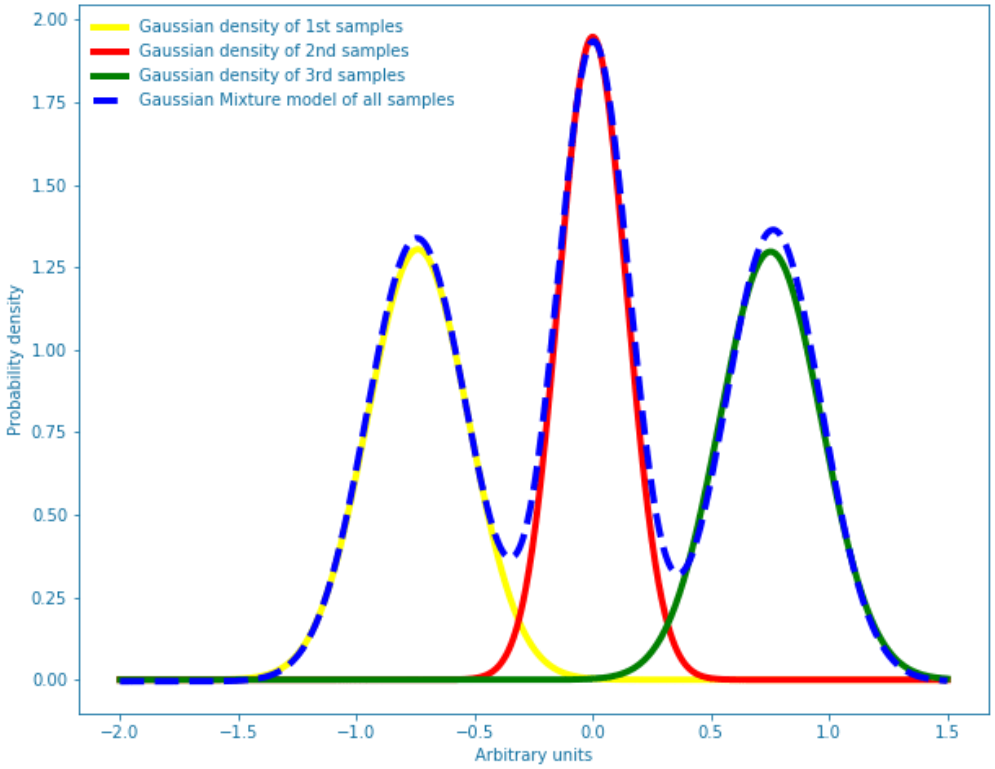
$$P(X|\lambda) = \sum_{k=1}^{K} w_k P_k\left(X|\mu_k, \Sigma_k\right),$$

where ;

$$P_k\left(X|\mu_k, \Sigma_k\right) = \frac{1}{\sqrt{2\pi|\Sigma_k|}} e^{\frac{1}{2}(X-\mu_k)^T \Sigma^{-1}(X-\mu_k)}$$

is the Gaussian distribution, with:

- Λ represents the training data.
- μ is the mean.
- Σ is co-variance matrices.
- wk represent the weights.
- K refers the index of the GMM components.




# 3 Implementation

The steps below are followed in order to develope a robust voice recognition model. Also ***sklearn, scipy, IPython, python_speech_features libraries*** are used.

**Training Part :**

**1)** Extract All Voice Sample

**2)** Seperate them based on gender with shuffling

**3)** Train Female file (2/3 of dataset)

- Feature Extraction: Extract Mel Frequency Cepstral Coefficient
- Create a Model: Train GMM models basen on MFCC
- Create a Model: Train Kmeans models basen on MFCC

**4)** Save models for two genders as Females and Males

**Testing Part:**

**1)** Feature Extraction: Extract Mel Frequency Cepstral Coefficient for testing sample

**2)** Compute likelihoods score by using Males GMM model

**3)** Compute likelihoods score by using Females GMM model

**4)** Compare scores

**5)** Decission male or female according to comparing scores. Biggest one is winner

**6)** Repeate all steps for Kmeans models

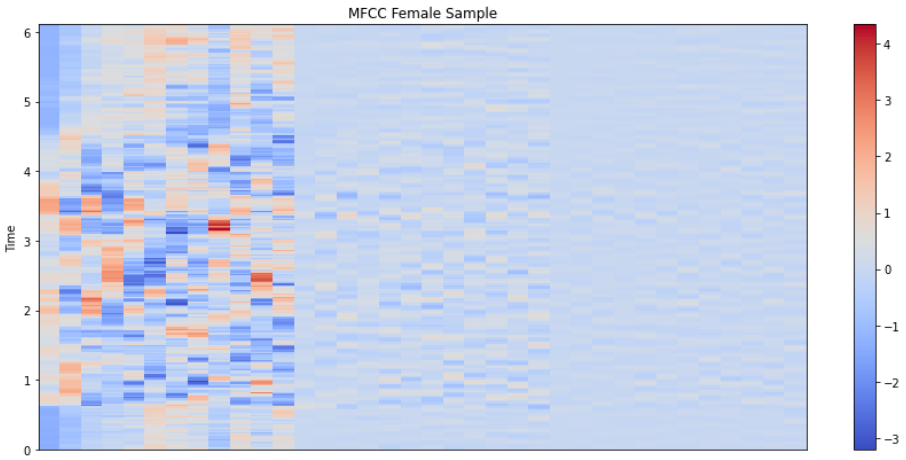
**Step 1 & Step2:** All voice samples are shuffled and splited to the training and testing sets for each gender. Training data is 2/3 and test data is 1/3 of dataset. Dataset consists of 3842(2186 female/1656 male) voice sample.

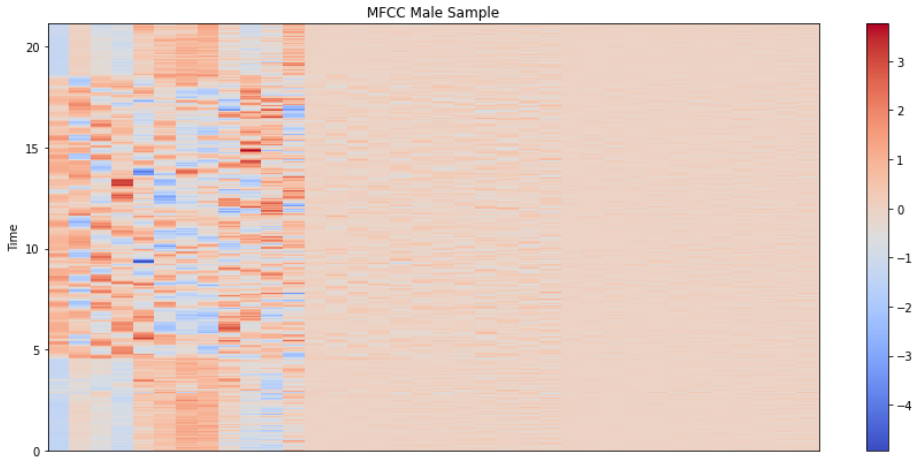
```
#Shuffle voice records and split the training and test sets by gender
female_train, female_test = train_test_split( shuffle(np.array(dataset_dict['f'])), test_size=0.33)
male_train, male_test = train_test_split( shuffle(np.array(dataset_dict['m'])), test_size=0.33)
```

**Step3:** For extracting feature of voice samples ***python_speech_features*** library is used. As seen below all input variables are given in detailed.

For instance *f0001_us_f0001_00003.wav* file is splited with 25 ms windowing size and 10 ms windowing step to farmes which is totally 263 frames, then for each frame 12 mfcc is calculated. because of delta function which we used for twice, number of Mfcc are 36.

```
# Extracting MFCC Function by using python_speech_features library
def extract_Mfcc(wave_path):
    """
    Extract Mel Frequency Cepstral Coefficient (MFCC) of a voice sample
    Args:
        audio_path (str) : path to wav file
    Returns:
        combined (array) : Extracted MFCC features
    """
    fs, wave  = read(wave_path)
    mfcc_feature = mfcc(wave, fs, winlen= 0.025, winstep=0.01, numcep=12, nfilt=30, nfft=512,appendEnergy=True)

    #wave :    The audio signal to extract features.
    #fs :      The sample rate of the signal
    #winlen :  The length of the analysis window in seconds.(Default is 0.025s)
    #winstep : The step between successive windows in seconds.(Default is 0.01s)
    #numcep  : The number of cepstrum to return. (Default is 13)
    #nfilt   : The number of filters in the filterbank. (Default is 26)
    #nfft    : The FFT size.(Default is 512)
    #appendEnergy : If true, the zeroth cepstral coefficient is replaced with the log of the total frame energy.

    mfcc_feature  = preprocessing.scale(mfcc_feature)  #scaling mfcc
    deltas        = delta(mfcc_feature, 2)  #calculating differential and acceleration coefficients
    double_deltas = delta(deltas, 2) #calculating differential and acceleration coefficients
    combined_mfcc      = np.hstack((mfcc_feature, deltas, double_deltas))
    return combined_mfcc
```




If a cepstral coefficient has a positive value, the majority of the spectral energy is concentrated in the low-frequency regions. On the other hand, if a cepstral coefficient has a negative value, it represents that most of the spectral energy is concentrated at high frequencies.

**Step 4:** For comparing two different unsupervised calssification method that are GMM and Kmeans are used to crate a model. For each classification technique, a model is created for each gender.

```python
def create_GMM_model(training_path, model_name):
    """
    Collect voice features from given path
    Args:
        training_path (str) : voice samples file path.
        model_name (str) : created model name
    Returns:
        None
    """

    files = os.listdir(training_path)
    features = np.asarray(())

    print("Processing.....! ")
    for file in files:  # collect voice features
        # extract MFCC & delta MFCC features from audio
        vector = extract_Mfcc( training_path + '\\' + file)

        # stack the features
        if features.size == 0:
            features = vector
        else:
            features = np.vstack((features, vector))


    # generate gaussian mixture models
    gmm_model = GMM(n_components = 16,  covariance_type='diag', n_init = 3)

    # fit features to models
    gmm_model.fit(features)

    # save models
    filename = pwd + "\\" + model_name + ".gmm"
    with open(filename, 'wb') as gmm_file:
        pickle.dump(gmm_model, gmm_file)
    print ("%5s %10s" % ("SAVING", filename))
```

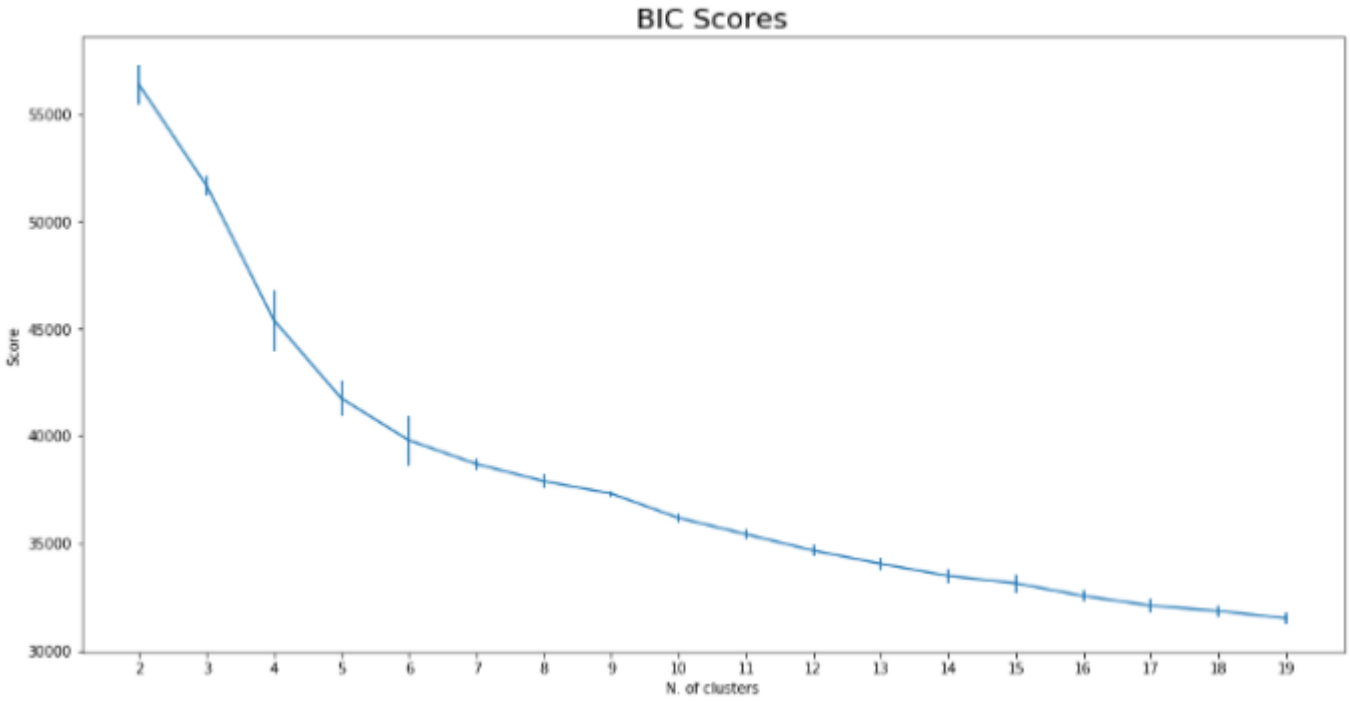
```python
def create_Kmeans_model(training_path, model_name):
    """
    Collect voice features from given path
    Args:
        training_path (str) : voice samples file path.
        model_name (str) : created model name
    Returns:
        None
    """
    files = os.listdir(training_path)
    features = np.asarray(())
    print("Processing.....! ")

    for file in files:  # collect voice features
        # extract MFCC & delta MFCC features from audio
        vector = extract_Mfcc( training_path + '\\' + file)

        # stack the features
        if features.size == 0:
            features = vector
        else:
            features = np.vstack((features, vector))

    # generate gaussian mixture models
    kmeans_model = KMeans(n_clusters = 7).fit(features)

    # save models
    filename = pwd + "\\" + model_name + ".km"
    with open(filename, 'wb') as km_file:
        pickle.dump(kmeans_model, km_file)
    print ("%5s %10s" % ("SAVING", filename))
```

While creating a GMM model selecting number of componenet of GMM model is the most important step. That's why BIC algorithm is used to specify n_components value.

The BIC criterion can be used to select the number of components in a Gaussian Mixture in an efficient way. In theory, it recovers the true number of components only in the asymptotic regime (i.e. if much data is available and assuming that the data was actually generated i.i.d. from a mixture of Gaussian distribution).

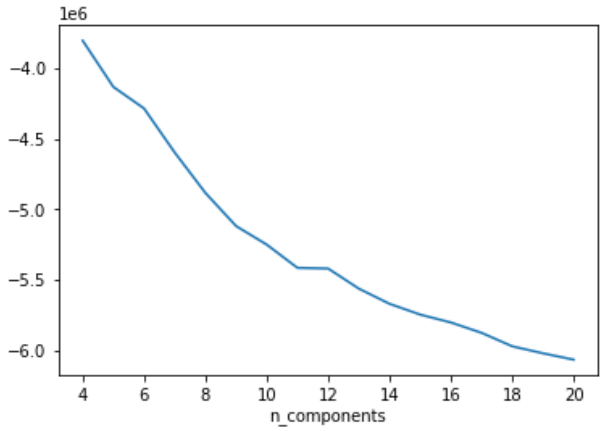
**Bayesian information criterion (BIC)**

This criterion gives us an estimation on how much is good the GMM in terms of predicting the data we actually have. The lower is the BIC, the better is the model to actually predict the data we have, and by extension, the true, unknown, distribution. In order to avoid overfitting, this technique penalizes models with big number of clusters.




Following this criterion, the bigger the number of clusters, the better should be the model. Which means that the penalty BIC criteria gives to complex models do not save us from overfit. Or, in more prosaic terms, this score sucks. At least in this basic form.

```
n_components = np.arange(4, 21)
gmm_models = [GMM(n, covariance_type='diag').fit(male_gmm_model_features)
              for n in n_components]

plt.plot(n_components, [m.bic(male_gmm_model_features) for m in gmm_models], label='BIC')
plt.xlabel('n_components');
```




As can be seen above, the BIC calculation is made with different componenet values. 16 will be most suitable for our model to handle high accuracy

**Step5:** To idenfy gender, a function is developed. Mainly this function determines a score for both the GMM model and the Kmeans model using the score method. Score method computes per-sample average log-likelihood of the given data for GMM model, for Kmeans model score method opposites of the value of given data on the K-means objective.

```python
def identify_gender(vector,model_name):
    """
    Identify gender by using created gmm model

    Args:
        vector (array) : Extracted MFCC features
        model_name (str) : Extracted MFCC features
    Returns:
        winner(str) : Female or Male
    """

    females_model = pickle.load(open(pwd + '\\female_' + model_name, 'rb'))
    males_model   = pickle.load(open(pwd + '\\male_' + model_name, 'rb'))

    # female hypothesis scoring
    female_score = np.array(females_model.score(vector))
    female_score_sum = female_score.sum()


    # male hypothesis scoring
    male_score          = np.array(males_model.score(vector))
    male_score_sum = male_score.sum()

    print("%10s %5s %1s" % ("* Female_Score",":", str(round(female_score_sum, 3))))
    print("%10s %7s %1s" % ("* Male_Score", ":", str(round(male_score_sum,3))))

    if male_score_sum > female_score_sum:
        winner = "male"
    else:
        winner = "female"
    return winner    # return winner
```

**Step6)** To test the male and female models we created before, we will use the test samples we splitted in the dataset before. We will perform feature extraction for each sample and calculate the score with the model we created. The higher the score value will be the model's prediction.

```
def testing(testing_path,model_name):
    """
    Testing  created GMM model with test voice sample

    Args:
        testing_path (str) : test voice samples file path.
        model_name (str) : km_model.km or gmm_model.gmm which are created model
    Returns:
        None
    """

    files = os.listdir(testing_path)
    # read the test directory and get the list of test audio files
    total_sample = 0
    error = 0

    for file in files:
        total_sample += 1
        print("%10s %8s %1s" % ("--> TESTING", ":", os.path.basename(file)))

        vector = extract_Mfcc( testing_path + '\\' + file) #extract MFCC feature of voice sample
        winner = identify_gender(vector,model_name)  #get winner form identify_gender function
        expected_gender = testing_path.split("\\")[-1][:-1]

        print("%10s %6s %1s" % ("* Expected_Gender",":", expected_gender))
        print("%10s %3s %1s" %  ("* Prediction", ":", winner))

        if winner != expected_gender: error += 1
        print("----------------------------------------------------")

    accuracy     = ( float(total_sample - error) / float(total_sample) ) * 100
    accuracy_msg = "*** Accuracy = " + str(round(accuracy, 3)) + "% ***"
    print(accuracy_msg)
```

# 4 Result

In this study a system which determines the gender of the speaker automatically according to the speech signal is proposed. The system composed of two stages. In the first stage the system is trained by using the sentences that are vocalized by known gender speakers. In the second stage the system is tested by using the sentences that are vocalized by unknown gender speakers. The speech records used in this study are taken from **The Free ST American English Corpus dataset (SLR45)**. Some of the speakers of database are used for the training of the system and the rest is used for testing. In the training stage by using the features vectors derived from the speech records determined for the training, GMM and K-means models are formed for male and female speakers each. MFCC coefficients are used as features vector in this study. MFCC coefficients are calculated by speech signals passing by 30 triangle filters that are placed linearly in melscale.

In the training stage, after forming the male and female models we pass to the test stage. In the test stage, MFCC features vectors derived from the test speakers are compared with the gender models of training stage. At the end, the gender of the speaker is determined by the models representing the larger score value.

The effects of the number of mixture, MFCC features dimensions and K-means performance are analyzed. The are listed in table.

| | MFCC # | # Mixture | # Kmeans | K-means Success rate % | GMM Success rate % |
|---|---|---|---|---|---|
| Male | 6 | 4 | 4 | 81,90% | 98,19% |
| | 6 | 16 | 16 | 92,87% | 99,03% |
| | 12 | 4 | 4 | 96,89% | 95,97% |
| | 12 | 16 | 16 | 99,00% | 99,45% |
| Female | 6 | 4 | 4 | 80,26% | 96,40% |
| | 6 | 16 | 16 | 89,58% | 97,09% |
| | 12 | 4 | 4 | 98,89% | 97,50% |
| | 12 | 16 | 16 | 100,00% | 99,72% |

***The effect of number of mixture and features dimention to the success:***

From the results of table, it is seen that the success rate increased from 81,90% to 92,87% with 6 MFCC coefficients. Especially in the k-menas model, the increase of the number of MFCC coefficients greatly influenced the success rate for both male and female.

Although the increase in the number of MFCCs affects the GMM model, it is not possible to see as the effect in the K-means model. It is also seen in the table that the increase in the number of components also affects the success rate.

Finally, it would be correct to say that the GMM model is a more suitable classification method for voice recognition.

# 5 Conclusions

In this study the gender of the speaker is determined by modeling the MFCC coefficient derived from the vocalized sentence of the speaker with GMM and K-means. It is seen that the increase of the classification components and MFCC coefficients are also increasing the success of the system. By analyzing the data amount used in the training of the system, it is evaluated that the reliability and accuracy of the system will be increased.