AUTOMATED DETECTION OF ARTIFICIAL VOICES

Student: Vivek Sharath
Supervisor: Alex Liu
Lab: SISL

Image from: http://www.clipartpanda.com/categories/voice-clip-art-free

Outline

Voice Conversion

Sprocket

Authentic vs Converted

MFCC

Logistic Regression

Conclusion

Voice Conversion

Voice Conversion is the transformation of one speaker's voice (the source) to sound like another speakers voice (the target)

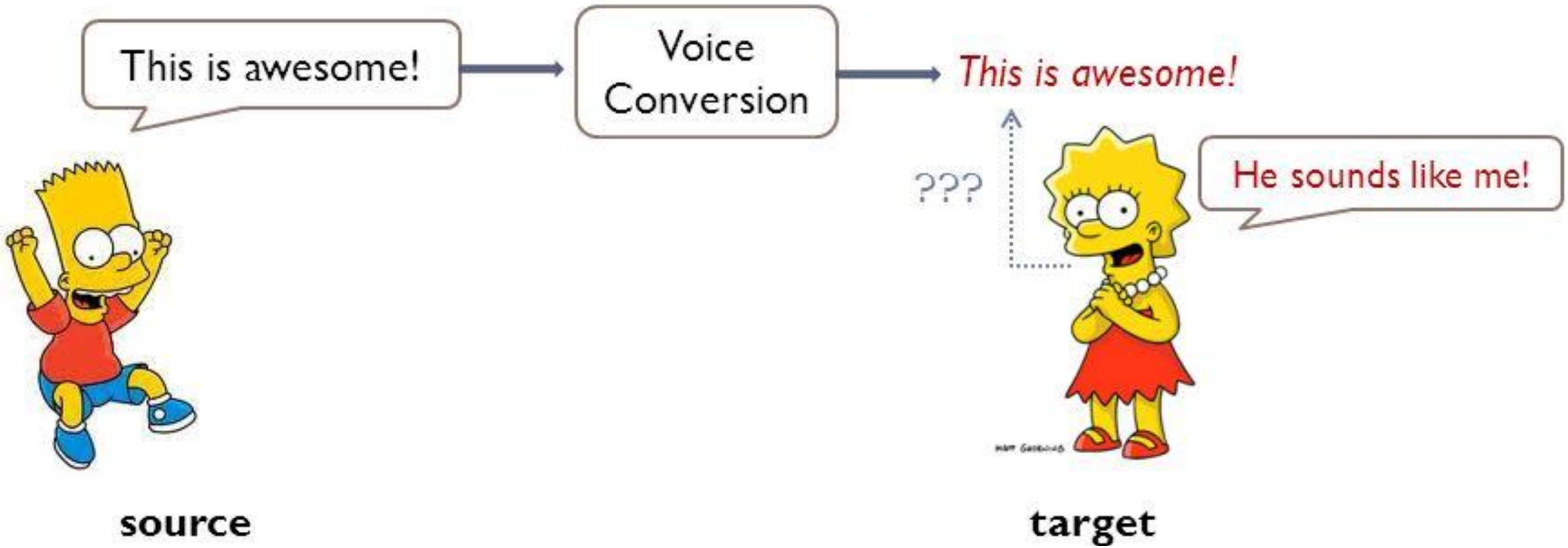


Image from https://slideplayer.com/slide/7589055/

Sprocket

Open-Source Voice Conversion Tool

1. Preparation of the parallel speech dataset
   a. Same Linguistic Info with different individual speakers
2. Acoustic feature extraction
   a. F0, MCC
3. Calculation of acoustic feature statistics
   a. Mean, SD of log(f0)
4. Time alignment between the source and target signals
5. f0is linearly transformed frame by frame using the speaker-dependent statistics of the source and target speakers in the logarithmic space using GMM

Kazuhiro KOBAYASHI
k2kobayashi

Ph.D. in Engineering. Research interests: Voice conversion.

Follow

Image from: https://github.com/k2kobayashi

*K. Kobayashi, T. Toda, "sprocket: Open-Source Voice Conversion Software," Proc. Odyssey, pp. 203-210, June 2018.

Sprocket

Open-Source Voice Conversion Tool

SF1	8/8/2018 3:58 PM	File folder
SF2	8/8/2018 3:58 PM	File folder
SF3	8/8/2018 3:58 PM	File folder
SM1	8/8/2018 3:58 PM	File folder
SM2	8/8/2018 3:59 PM	File folder
TF1	8/8/2018 3:59 PM	File folder
TF2	8/8/2018 3:59 PM	File folder
TM1	8/8/2018 3:59 PM	File folder
TM2	8/8/2018 3:59 PM	File folder
TM3	8/8/2018 3:59 PM	File folder

- Uses F0 transformation
- Parallel Data Set
- Each folder has 216 of the same spoken sentences
- Sentences from The Call of the Wild by Jack London

Kazuhiro KOBAYASHI
k2kobayashi

Ph.D. in Engineering. Research interests: Voice conversion.

Follow

Image from: https://github.com/k2kobayashi

*K. Kobayashi, T. Toda, "sprocket: Open-Source Voice Conversion Software," Proc. Odyssey, pp. 203-210, June 2018.

Sprocket

Open-Source Conversion Tool

Source

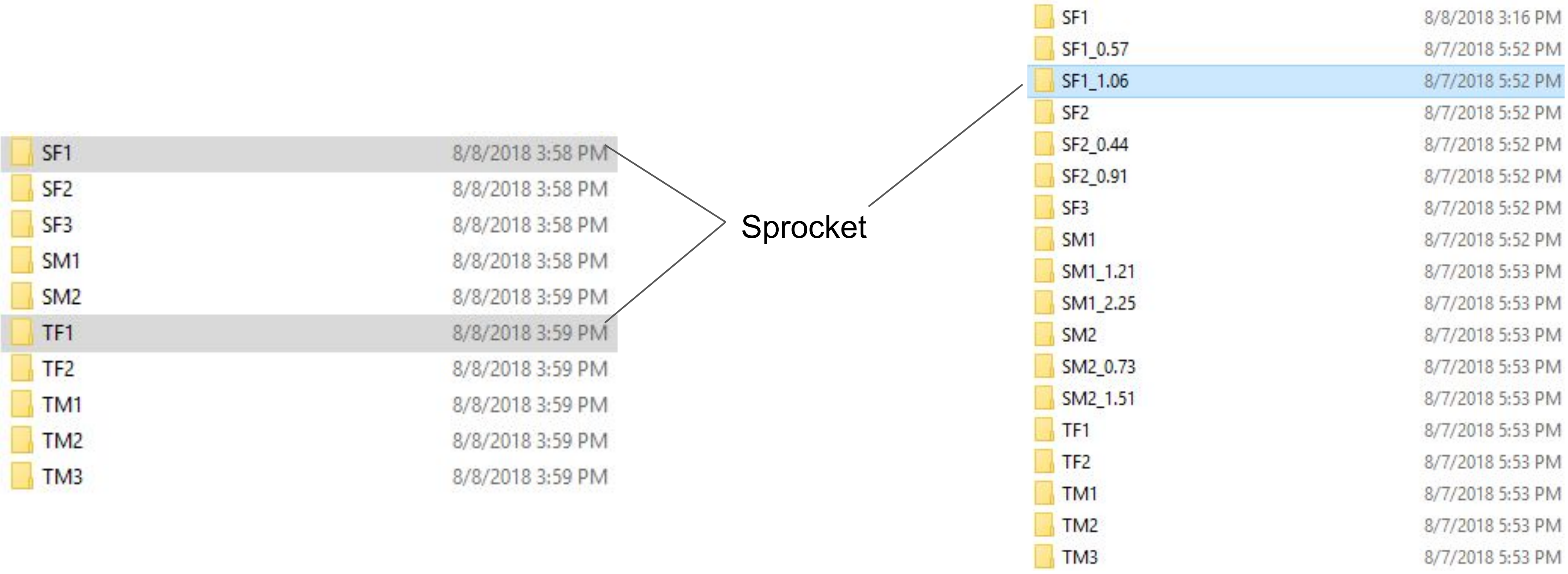
SF1	8/8/2018 3:58 PM	File folder
SF2	8/8/2018 3:58 PM	File folder
SF3	8/8/2018 3:58 PM	File folder
SM1	8/8/2018 3:58 PM	File folder
SM2	8/8/2018 3:59 PM	File folder
TF1	8/8/2018 3:59 PM	File folder
TF2	8/8/2018 3:59 PM	File folder
TM1	8/8/2018 3:59 PM	File folder
TM2	8/8/2018 3:59 PM	File folder
TM3	8/8/2018 3:59 PM	File folder

Target

Name	#	Title
100001		
100002		
100003		
100004		
100005		
100006		
100007		
100008		
100009		
100010		
100011		
100012		
100013		
100014		
100015		
100016		
100017		
100018		
100019		
100020		
100021		
100022		
100023		
100024		
100025		

Sprocket

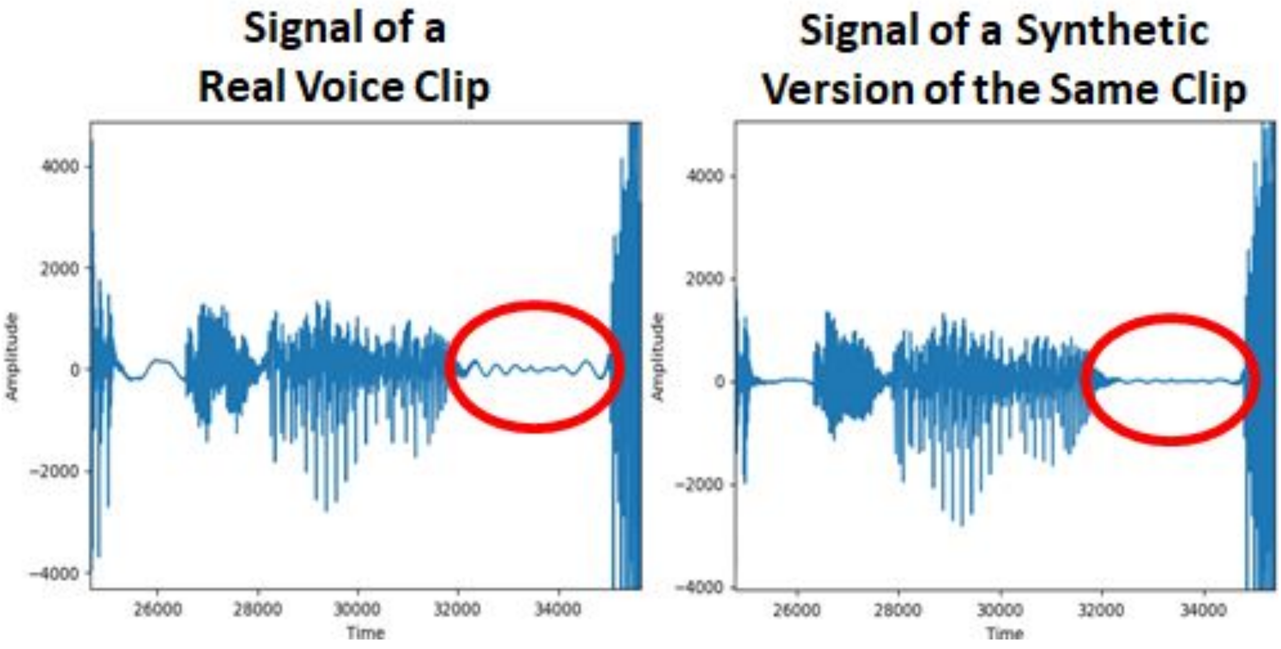
Open-Source Conversion Tool



Voice Conversion 101

Image From: https://kids.nationalgeographic.com/videos/real-or-fake/#real_or_fake__ep_1.mp4

Voice Conversion 101




```python
# Import Necessary Libraries
import wave
from scipy.io import wavfile
import matplotlib.pyplot as plt

# Define a Signal Reading Function
def signal(file):

    # Open Wav Files
    y= wave.open(file, 'r')

    # Retrieve and Plot Signal Information
    samplerate, data = wavfile.read(file)
    plt.plot(data)

    # Add Axis-Labels and a Title
    plt.xlabel("Time")
    plt.ylabel("Amplitude")
    plt.title("Signal For "+str(file))

    # Display Signal Visualization
    plt.show()

# Pass Sample Wav Files for Visualization
signal('sample_real.wav')
signal('sample_fake.wav')
```

MFCC

Mel-Frequency Cepstrum Coefficients: Quantitative Representation of a Sound

Step 1: Take the Fourier transform of a signal.

Fourier Transform:

- Mathematical Transformation
- Sound Visualization
- Takes in a signal, outputs the frequencies of the signal (decomposition)

Image From: https://ibmathsresources.com/2014/08/14/fourier-transforms-the-most-important-tool-in-mathematics/

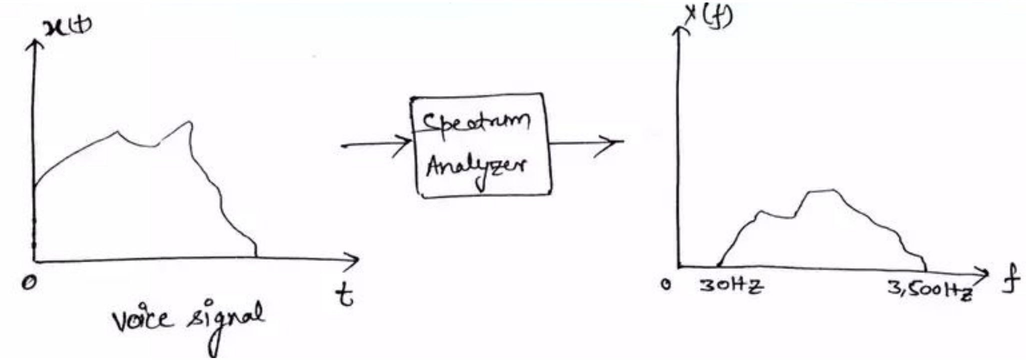
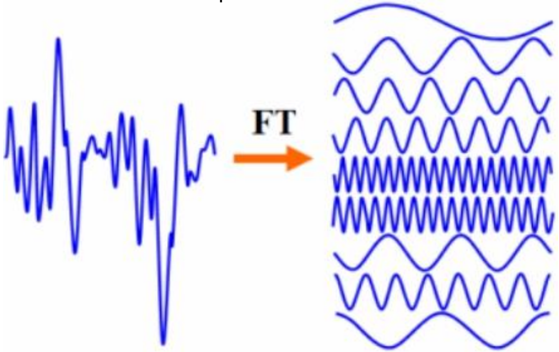


Image From: https://unacademy.com/lesson/fourier-transform/V8XBCM8H



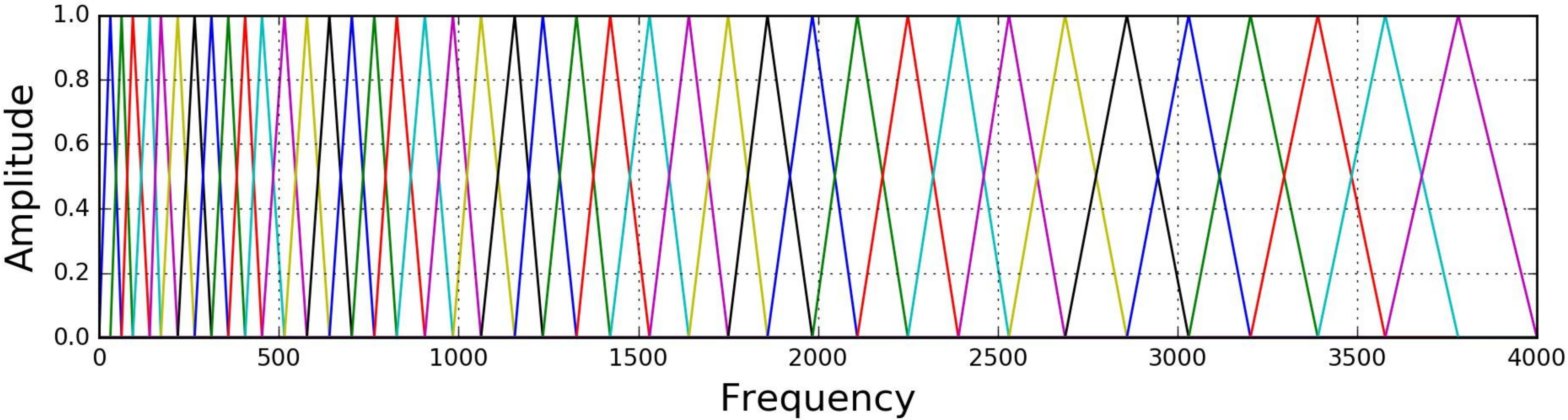
MFCC

Mel-Frequency Cepstrum Coefficients: Quantitative Representation of a Sound

Step 2: Place each frequency in a bin of the Mel-spaced filterbank (usually 40 bins)

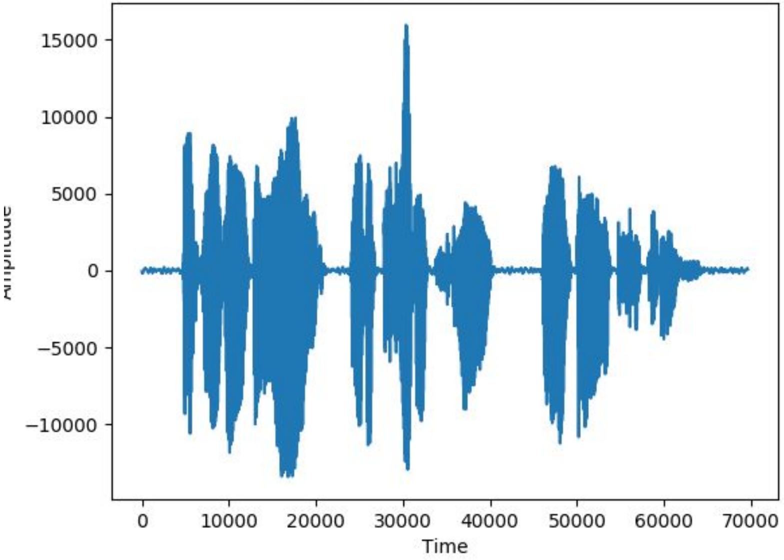
Step 3: For each bin, sum the total weighted FFT energy

Step 4: cosine_transform(log(bin amplitude))



https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

MFCC




```
C:\Users\sharath\Desktop\New folder\4Example\SF1>test.py
[[ 4.98309782  4.48906841  3.08090581  2.70519673  3.04931724  2.08110305
   2.65866915  2.27108082  2.32199479  1.6289689   1.51515088  0.68061874
   0.91996503  0.68847051 -0.23562629  0.2801078   0.69131802  1.08069291
   0.83015588  1.27074578  1.5037787   1.55149938  1.87316154  2.18517039
   2.55678033  2.65081813]
 [ 5.86651209  4.11344819  3.32430422  2.91667646  2.65467789  2.54477587
   2.49765907  2.27945961  2.15726025  1.69420113  1.38752178  0.87106103
   0.67452007  0.81461573  0.52179871  0.25989006  0.56495444  0.8932853
   0.79781567  1.1248368   1.69579586  1.63115796  1.88482033  2.19828251
   2.45063971  2.62355144]]
```

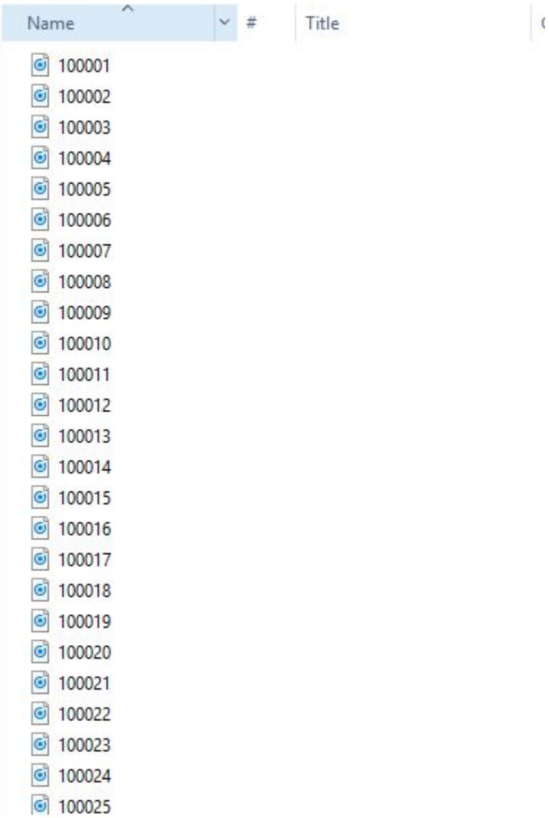
MFCC

Name	Date modified	Type
SF1	8/2/2018 3:36 PM	File folder
SF1_0.57	8/2/2018 3:37 PM	File folder
SF1_1.06	8/2/2018 3:38 PM	File folder
SF2	8/2/2018 3:38 PM	File folder
SF2_0.44	8/2/2018 3:38 PM	File folder
SF2_0.91	8/2/2018 3:38 PM	File folder
SF3	8/2/2018 3:38 PM	File folder
SM1	8/2/2018 3:38 PM	File folder
SM1_1.21	8/2/2018 3:38 PM	File folder
SM1_2.25	8/2/2018 3:38 PM	File folder
SM2	8/2/2018 3:38 PM	File folder
SM2_0.73	8/2/2018 3:39 PM	File folder
SM2_1.51	8/2/2018 3:39 PM	File folder
TF1	8/2/2018 3:39 PM	File folder
TF2	8/2/2018 3:39 PM	File folder
TM1	8/2/2018 3:39 PM	File folder
TM2	8/2/2018 3:39 PM	File folder
TM3	8/2/2018 3:39 PM	File folder

18 Folders containing Parallel Speech Data

(8 converted/ 10 authentic)

MFCC



216 WAV files in each Folder

3888 Total WAV Files

- 1728 Generated
- 2160 Authentic

MFCC

1	Type	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
2	real	4.983098	4.489068	3.080906	2.705197	3.049317	2.081103	2.658669	2.271081	2.321995	1.628969	1.515151	0.680619	0.919965	0.688471	-0.23563	0.280108
3	real	3.314698	-0.06731	-1.41503	-1.488	-2.01247	-1.1119	-1.50148	-2.18614	-0.74443	-0.68792	-0.39716	0.252342	0.440372	-0.30139	-0.27842	0.378102
4	real	4.550292	1.095015	-0.60739	-0.90986	-0.63432	-0.78976	-1.35809	-1.1733	-1.04979	-1.0442	-0.48244	-0.42573	0.263374	0.232708	0.497629	0.619203
5	real	4.749839	2.372827	1.315197	0.355386	0.050234	0.135686	-0.49658	-0.46441	-0.20285	-0.42562	-0.1245	0.345643	0.583021	0.533467	0.33706	-0.17954
6	real	5.535902	2.65094	1.074992	0.725572	0.558506	-0.09089	-0.18571	-0.32225	-0.8328	-0.26233	0.150567	0.264002	1.029821	1.051315	1.161307	0.293816
7	real	5.770577	2.342821	2.270461	2.009692	1.892082	2.063533	1.593983	1.247262	0.747946	0.922378	0.304684	0.987046	0.43662	0.160083	0.195957	0.291032
8	real	4.95503	1.600573	0.379949	-0.10311	-0.53827	-0.53635	-0.6365	-0.58182	-0.92122	-0.1504	-0.26131	-0.06378	0.397151	0.842921	0.564432	0.743246
9	real	3.897684	1.397563	-0.77849	0.119335	-1.49356	-0.2688	-0.2381	-0.47414	-0.56734	-0.17714	-0.23689	-0.00785	0.256228	0.341339	-0.24123	0.349786
10	real	3.684922	0.241989	-0.47044	-0.97043	-0.60976	-0.57284	-0.82467	-1.0243	-0.15505	-0.62275	-0.60039	-0.30864	0.858797	0.696079	0.345473	0.894775
11	real	4.941478	0.905687	0.334489	-0.49487	-0.63986	-0.91365	-1.15318	-0.93354	-0.71477	-0.72963	-0.32277	-0.00075	0.903851	0.232054	0.328084	0.550693
12	real	4.808288	2.686507	2.187295	2.057421	2.009557	2.034851	1.985161	1.665429	1.612475	1.344821	0.869291	0.533287	0.20495	0.581571	-0.15567	0.426671
13	real	4.527809	1.802199	0.821447	-0.08631	-0.25756	-0.49814	-0.41132	-0.18374	-0.65587	-0.95527	-0.43845	-0.60427	-0.26598	0.188272	0.559177	-0.3272
14	real	4.464064	1.815888	1.109144	0.496515	-0.57821	-0.58072	-0.54972	-0.59817	-0.35988	-0.66114	0.007674	-0.00437	-0.00925	0.476236	0.599886	0.290076
15	real	6.048739	2.97133	2.200134	1.055099	0.515096	0.447058	0.073658	0.173143	-0.20931	-0.27892	0.180882	0.032865	0.696579	0.547744	0.443815	0.680122
16	real	4.88094	2.153436	0.853172	0.385164	-0.12414	-0.56387	-0.68173	-0.4391	-0.7779	0.117604	0.366922	0.275437	0.580432	0.923758	0.596237	0.032365
17	real	4.830605	2.450225	1.358987	0.83337	0.013302	0.009603	0.066191	-0.02057	0.021858	-0.71416	0.430282	-0.28896	0.120816	0.326746	-0.02427	0.401616
18	real	4.648372	1.874	-0.69503	-1.18928	-1.08621	-1.62213	-0.42249	-0.57621	-0.57465	-0.9366	-0.7494	-0.52436	0.54078	0.710553	0.649634	0.806962
19	real	6.433318	3.917564	3.313918	2.090278	1.633197	1.754937	1.380551	0.60573	0.581155	0.71402	0.926527	0.663831	0.701624	0.854801	1.070711	0.573121
20	real	4.824956	1.495707	-0.25823	-0.72961	-0.75254	-1.49324	-0.74531	-0.97599	-0.22295	-0.16182	-0.51434	0.084583	0.252973	0.691631	0.410443	0.76529
21	real	4.295682	1.915804	0.848909	0.414462	-0.72583	-1.06559	-1.1903	-0.66534	-0.08727	-0.24052	0.309077	0.120964	0.626627	0.937172	0.945478	0.940304
22	real	5.810046	3.167758	1.697073	1.044547	0.78277	0.076173	0.257847	-0.20568	-0.08857	-0.0762	0.230601	-0.04124	0.463588	0.217836	0.042364	0.467227
23	real	3.470187	1.529184	0.475503	-0.29906	-0.62636	-0.74372	-1.03745	-0.58112	-0.3982	-0.6541	-0.05935	-0.27105	-0.2116	0.262873	0.278661	0.35186

Logistic Regression

```{r}
# DATA UPLOAD AND PARTITION

# Connect to the MFCC CSV
mfcc <- read.csv("mfcc.csv", header=TRUE)

set.seed(126)  # Set a Seed to make Results Reproducable

# Create a Partition with a 80/20 Train and Test Split
indexes <- createDataPartition(mfcc$Type, times = 1, p = .8, list = FALSE)

train_data <- mfcc[indexes, ] # Train Data
test_data <- mfcc[-indexes, ] # Test Data
```

```{r}
# MODEL CREATION

# Logistic Regression Model
glm.out <- glm(fmla, data=train_data, family=binomial)

# Display Model Results
summary(glm.out)
```

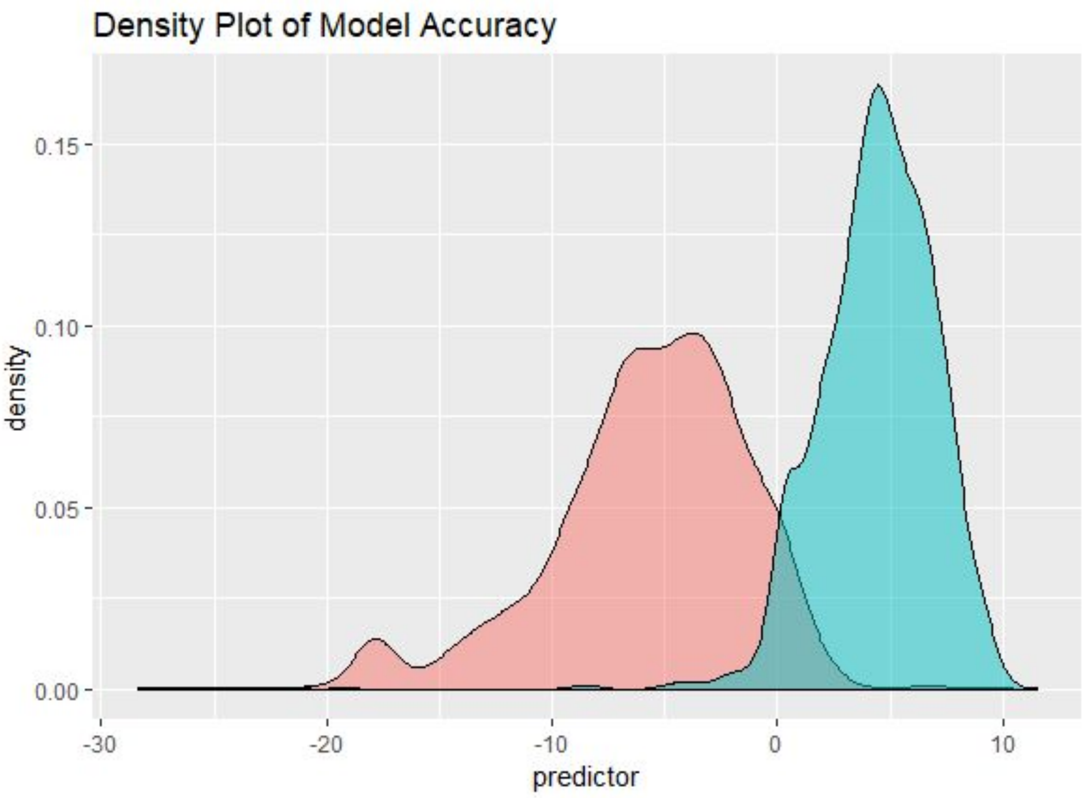
80% Train
20% Test

Logistic Regression

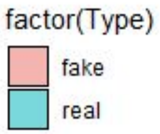
Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.147019	0.312540	-3.670	0.000243	***
f0	1.343646	0.155454	8.643	< 2e-16	***
f1	-0.447764	0.168947	-2.650	0.008041	**
f2	-0.073959	0.222231	-0.333	0.739283	
f3	0.275166	0.225325	1.221	0.222011	
f4	0.222459	0.232909	0.955	0.339511	
f5	0.009015	0.253479	0.036	0.971630	
f6	0.279762	0.243222	1.150	0.250047	
f7	-0.455193	0.251779	-1.808	0.070621	.
f8	-0.271292	0.282378	-0.961	0.336683	
f9	-0.270565	0.286234	-0.945	0.344528	
f10	0.162280	0.296640	0.547	0.584338	
f11	0.240621	0.320025	0.752	0.452123	
f12	0.106190	0.322467	0.329	0.741925	
f13	-0.072673	0.348726	-0.208	0.834919	
f14	0.612167	0.362341	1.689	0.091128	.
f15	-0.055865	0.348631	-0.160	0.872692	
f16	-0.100628	0.369034	-0.273	0.785100	
f17	0.461113	0.396483	1.163	0.244826	
f18	-0.486431	0.438879	-1.108	0.267711	
f19	0.056419	0.428892	0.132	0.895344	
f20	-0.355895	0.433357	-0.821	0.411504	
f21	-0.017025	0.486384	-0.035	0.972077	
f22	-0.103284	0.506796	-0.204	0.838512	
f23	-0.192789	0.471939	-0.409	0.682903	
f24	-0.159135	0.528696	-0.301	0.763418	
f25	-1.217319	0.491385	-2.477	0.013237	*
f26	1.237061	0.155667	7.947	1.91e-15	***
f27	-0.484449	0.181778	-2.665	0.007698	**
f28	-0.025460	0.232281	-0.110	0.912721	
f29	0.110565	0.217190	0.509	0.610702	
f30	0.382974	0.215793	1.775	0.075942	.
f31	-0.238361	0.242999	-0.981	0.326636	
f32	0.431847	0.231132	1.868	0.061707	.
f33	-0.270182	0.244673	-1.104	0.269482	
f34	-0.691284	0.277391	-2.492	0.012699	*
f35	0.159514	0.286944	0.556	0.578276	
f36	-0.289484	0.297473	-0.973	0.330482	
f37	0.478282	0.309537	1.545	0.122308	
f38	0.458000	0.332974	1.375	0.168981	
f39	-0.410817	0.343405	-1.196	0.231578	
f40	-0.167971	0.349815	-0.480	0.631106	
f41	-0.214030	0.349534	-0.612	0.540320	
f42	0.200394	0.378940	0.529	0.596925	
f43	0.473407	0.378735	1.250	0.211311	
f44	0.033142	0.445328	0.074	0.940674	
f45	0.124820	0.399545	0.312	0.754731	
f46	0.389209	0.444321	0.876	0.381050	
f47	-1.611413	0.509740	-3.161	0.001571	**
f48	0.100661	0.485720	0.207	0.835822	
f49	-0.206162	0.463695	-0.445	0.656604	
f50	2.488942	0.500953	4.968	6.75e-07	***
f51	-1.989800	0.499832	-3.981	6.86e-05	***

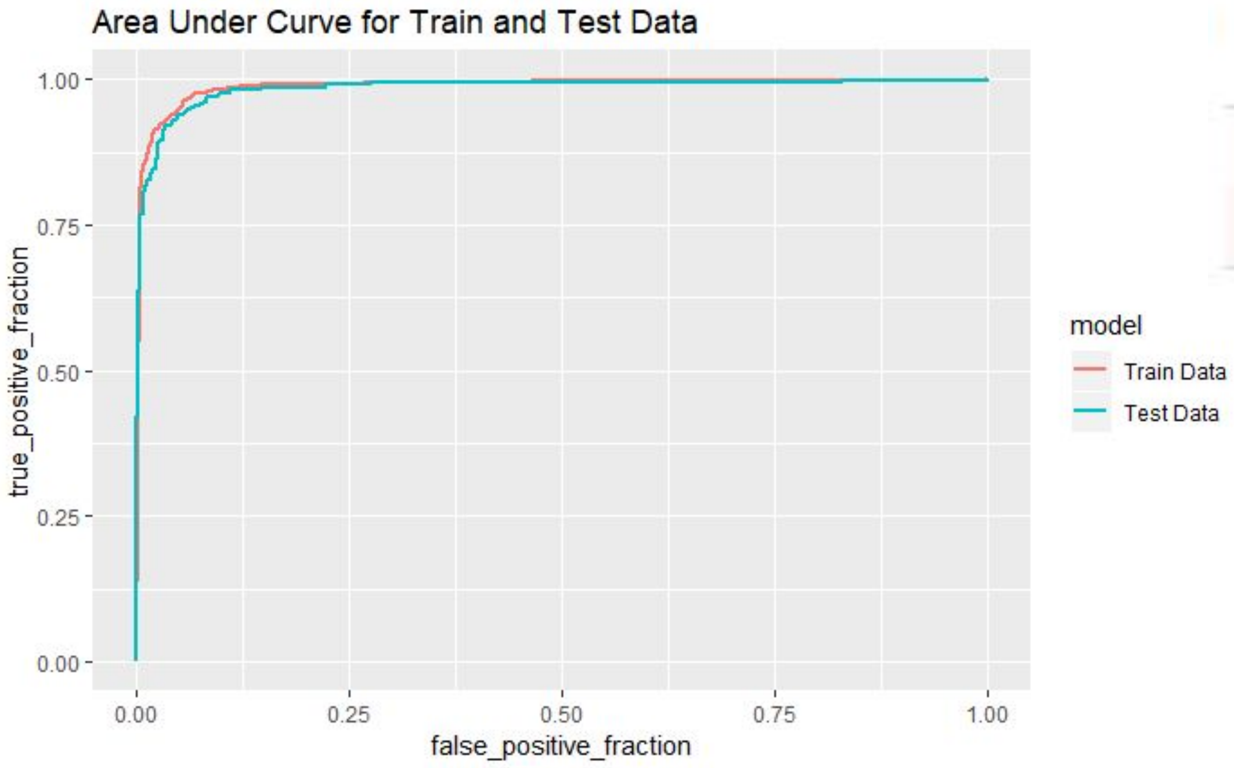
Logistic Regression




```{r}
# DENSITY PLOT OF CLASS SEPERATION

# Assessing Model Efficacy with the Training Data
lr_data <- data.frame(predictor=predict(glm.out, train_data),
                      Type = train_data$Type)

# Plot the Results
ggplot(lr_data, aes(x=predictor, fill=factor(Type))) +
  geom_density(alpha=.5) + ggtitle("Density Plot of Model Accuracy")
```



Logistic Regression



model <fctr>	AUC <dbl>
Train Data	0.9870760
Test Data	0.9838164

Logistic Regression

Confusion Matrix

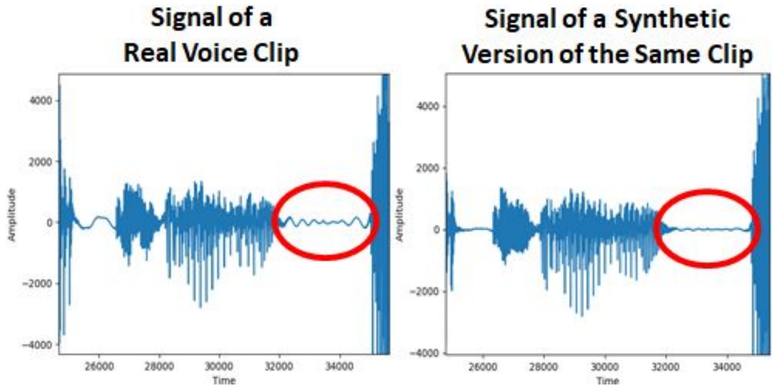
	fake <int>	real <int>
0	202	2
1	143	430

```
"Accuracy: 81.3%"
"Precision: 99.5%"
"Recall: 75%"
"f1: 85.6%"
```

```
# MODEL RESULTS

# Determine Model Performance on Test Data
predicted <- predict(glm.out, test_data, type="response")

# Find Optimal Threshold
optCutOff <- optimalCutoff(test_data$Type, predicted)

# Create a Confusion Matrix
confusion_mfcc <- confusionMatrix(test_data$Type, predicted,
                                  threshold = optCutOff)

# Display Confusion Matrix
confusion_mfcc
```

Conclusion

- Quantified Slight Differences in Signal of Real vs Converted using MFCC



- AUC of of .98 seems a little too high

- Artificial Data came from one source: Sprocket

Future Research

- Other Machine Learning Approaches
- Testing Model on New Data
- Using only Significant MFCC Features
- Discovering more features to include
- Gender / Age / Accent / Pitch
- MCC vs MFCC

THANK YOU