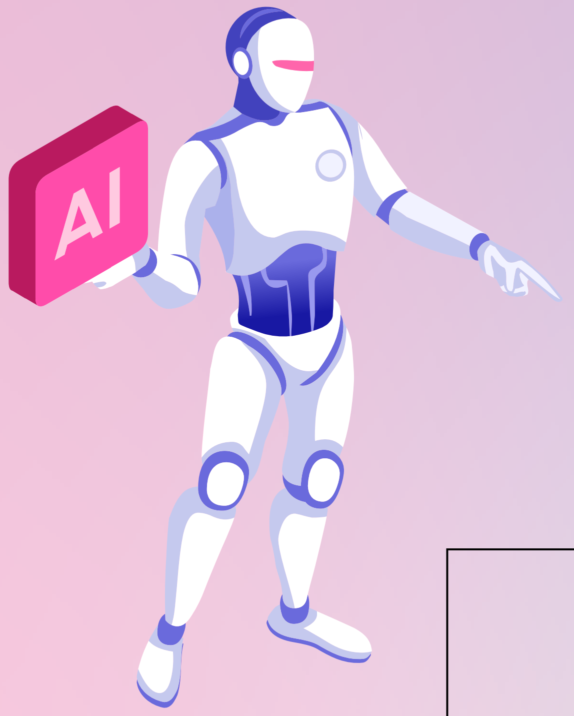


PROJECT 1

Machine Learning

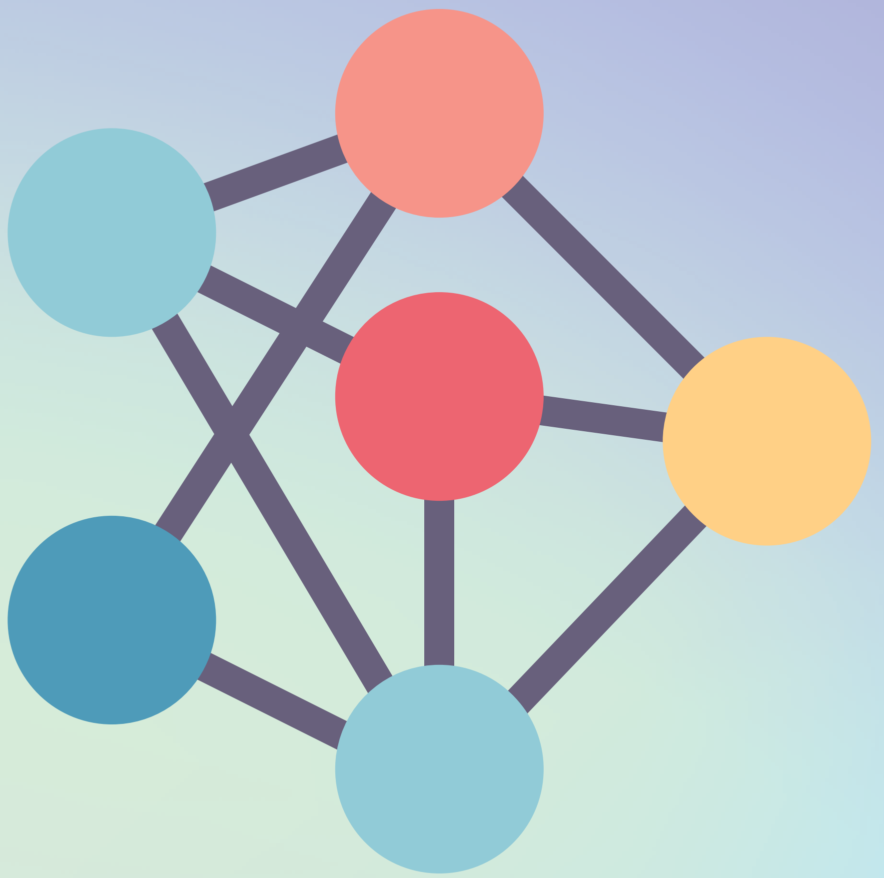
Synthetic Speech Detection & Attribution

Group No 16
Abdul Hannan Anjum Chaudhry 2023-11-0058
Muhammad Hamza 2023-11-0359
Zeeshan Ashraf 2023-02-0083

Overview

- Introduction
- Literary Review
- Methodology
- Implementation
- Results
- Analysis
- Conclusion

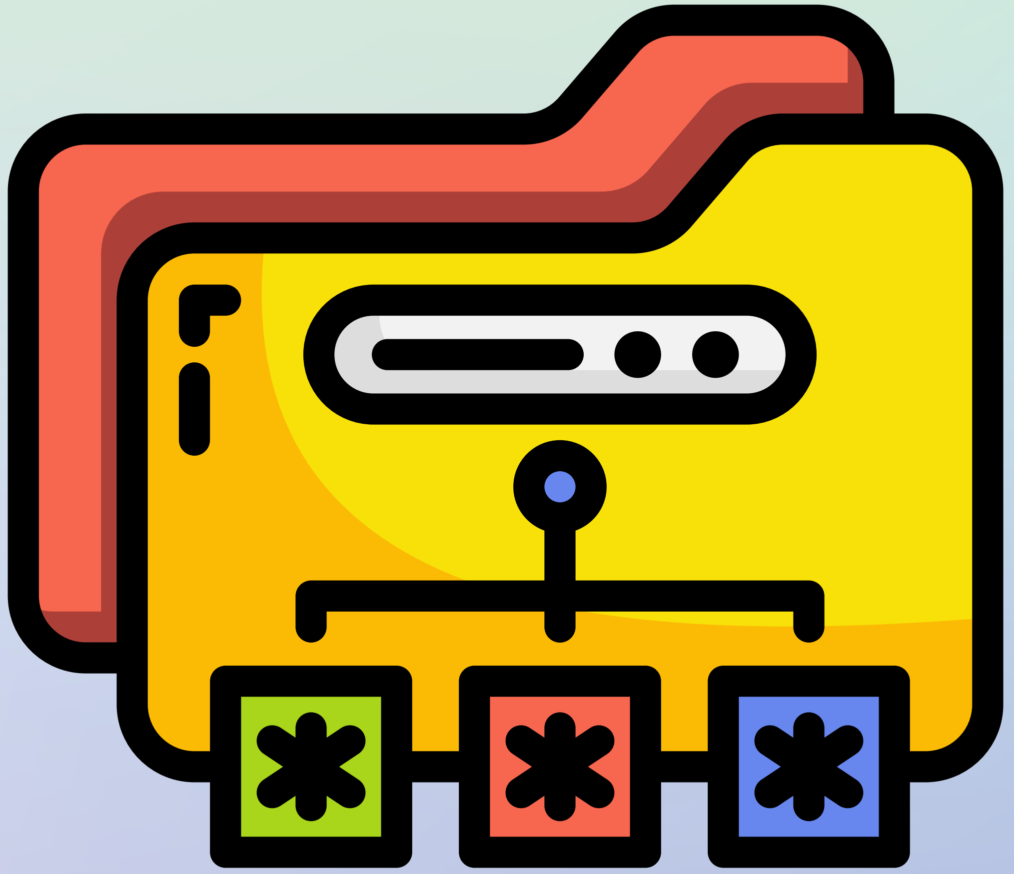
Introduction



- Manipulation of audio, speech and video has become easier
- logical advances in the area of signal processing, machine learning and deep learning
- classification of algorithms used to generate different synthetic audios
- development of a classifier to identify the algorithm used for the generation of a synthetic audio.

Data Set

- 5000 synthetic audio recordings generated from 5 different algorithms
- dataset of 15000 samples of noisy synthetic speech recordings using noise addition, reverberation, filtering, and lossy compression.



Literary Review

Features Extraction

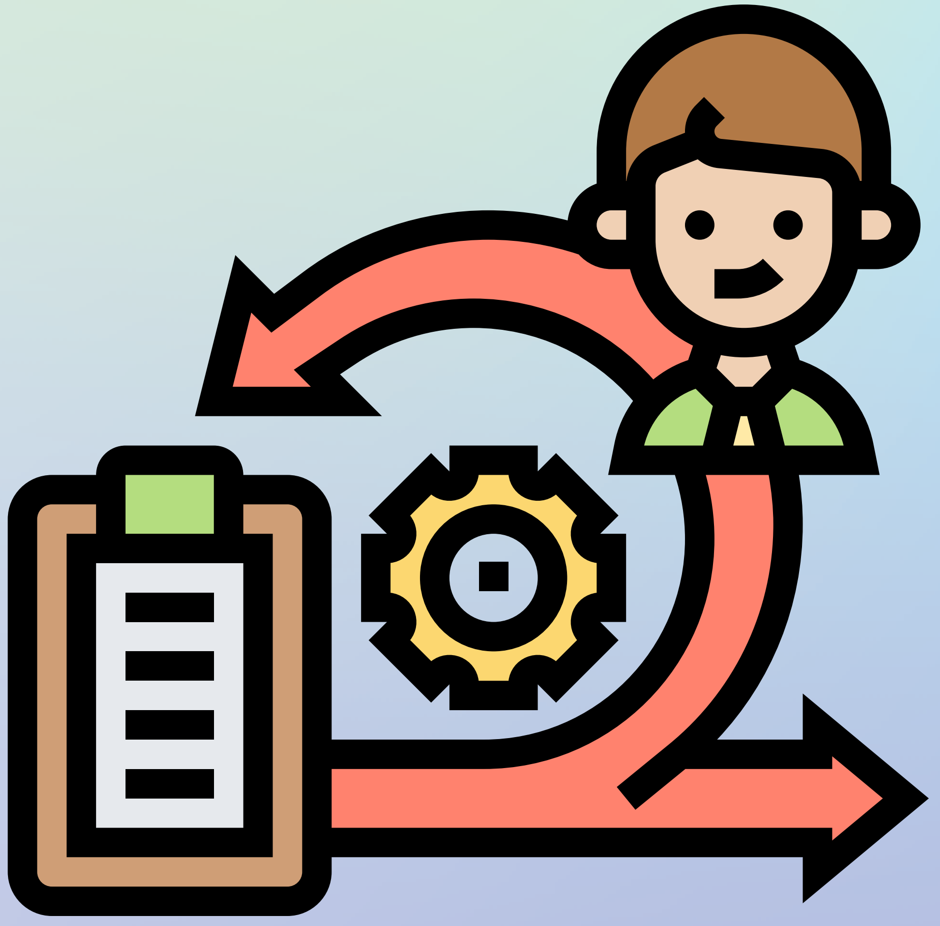
- zero-crossing rate (ZCR),
- harmonic distribution
- Mel-Frequency Cepstral Coefficients
- Constant-Q transform (CQT),

Classification Models

- Logistic Multi Linear Regression
- Gaussian mixture models (GMMs)
- Multilayer perceptron (MLPs))
- Recurrent neural networks (RNNs)
- Kalman filters
- convolutional neural networks (CNNs)

Methodology

- Pre-Processing
- Feature Extraction
- Feature Selection
- Dimensionality Reduction
- Machine Learning Models Selection



Implementation

Feature Extraction	• The feature extraction in our models is done through the librosa library, which is very efficient MFCC, Spectral Centroid, Chromagram
Dimensionality reduction	• Principle Component Analysis(PCA) to reduce the dimensions because it is difficult to handle large dimensions in the ML models
Model Implementation	• SVM, KNN, Naïve Bayes, Neural Network, and Logistic regression model using the Sklearn library and Hyperparameter tuning to find the best possible parameters

Features extraction through librosa

```
mfcc_list = []
chroma_list = []
spectral_centroid=[]

for idx,aud in enumerate(aud_list):
  print(idx)
  signal, sr = librosa.load(aud)
  signal = signal.flatten()
  mfccs = librosa.feature.mfcc(signal, n_mfcc=13,sr=sr) #extracting mfccs
  #now for delta and delta2 mfccs
  delta_mfcc = librosa.feature.delta(mfccs) #the delta features show how the signals vary with time, will be useful for stuff
  delta2_mfcc=librosa.feature.delta(mfccs,order=2)
  final_mfcc = np.concatenate((mfccs,delta_mfcc,delta2_mfcc))
  #scaling
  final_scaled=np.mean(final_mfcc.T,axis=0) #Scaled features,
  mfcc_list.append(final_scaled)
  chroma_cq = librosa.feature.chroma_stft(y=signal, sr=sr, n_fft=4096) #extracting chroma stft
  chroma_cq = np.mean(chroma_cq.T,axis=0)
  chroma_list.append(chroma_cq)
  cent = librosa.feature.spectral_centroid(y=signal, sr=sr) #extracting spectral centroid
  cent = np.mean(cent.T,axis=0)
  spectral_centroid.append(cent)
```

80 20 split after standardization and PCA implementation

```
scaler = preprocessing.StandardScaler().fit(mfcc_clean_train)
mfcc_clean_train_scaled = scaler.transform(mfcc_clean_train)
print(mfcc_clean_train_scaled[0])
mfcc_clean_test_scaled = scaler.transform(mfcc_clean_test)
from sklearn.decomposition import PCA
pca = PCA(n_components = 0.95)
pca.fit(mfcc_clean_train_scaled)
mfcc_train1 = pca.transform(mfcc_clean_train_scaled)
mfcc_test1 = pca.transform(mfcc_clean_test_scaled)
print("After mfcc shape")
print(mfcc_train1[0].shape)
```

Sklearn implementation of a Neural Network

```
] from sklearn.metrics import classification_report
  from sklearn.neural_network import MLPClassifier

  def NeuralNetwork(X_train,X_test,y_train,y_test):
    parameters = {
      'learning_rate_init': [0.05, 0.01, 0.005, 0.001],
      'hidden_layer_sizes': [4, 8, 12],
      'activation': ["relu","logistic", "tanh"],
      'batch_size':[1000],
      'max_iter':[10000]}
    final3 = GridSearchCV(estimator=MLPClassifier(),param_grid=parameters,scoring='accuracy',cv=5)
    final4=final3.fit(X_train,y_train)
    predict_list= final4.predict(X_test)

    acc_score = accuracy_score(y_test,predict_list)
    report = classification_report(y_test,predict_list)
    return((acc_score,final4.best_params_['learning_rate_init'],final4.best_params_['hidden_layer_sizes'],final4.best_params_['activ
```

Results

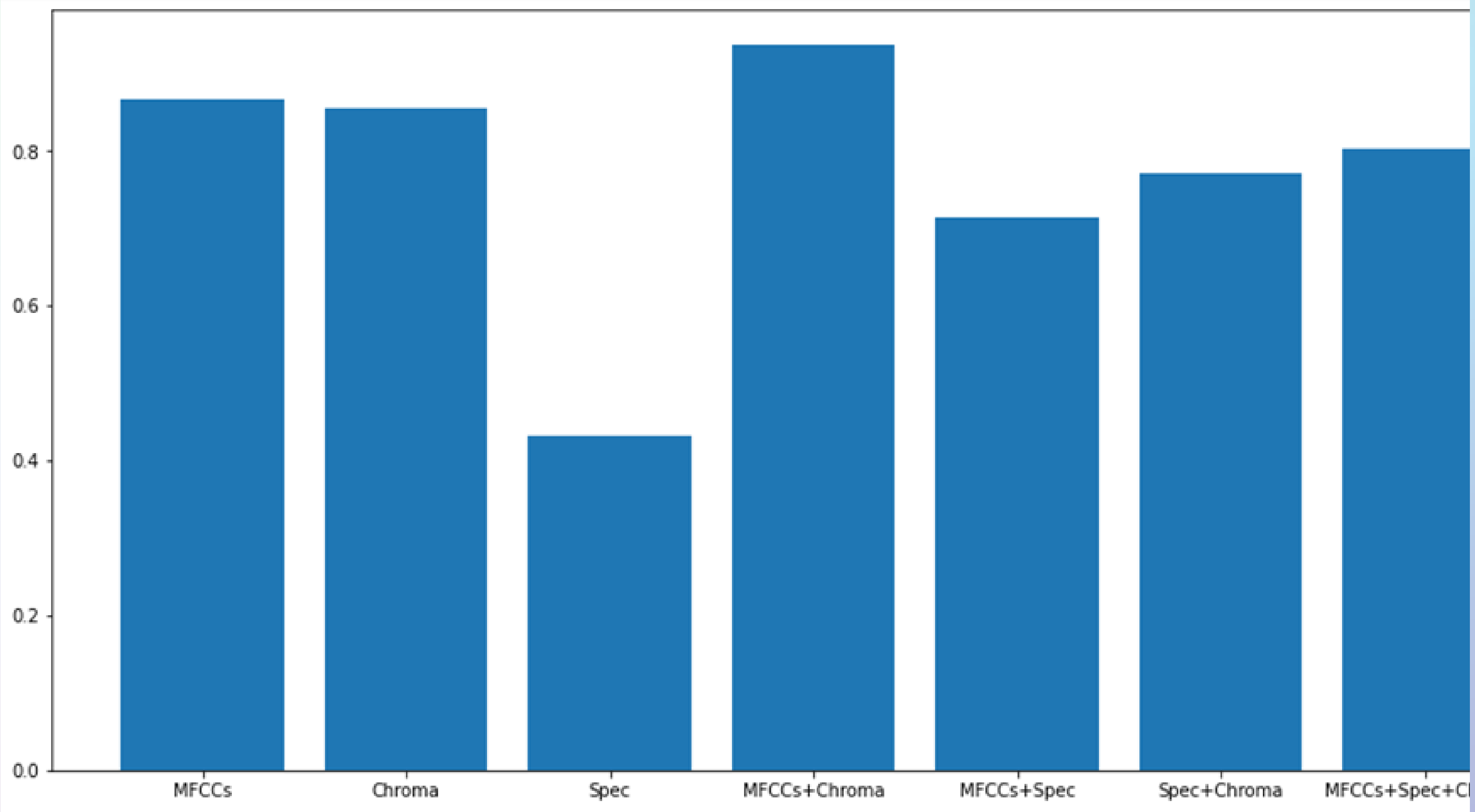
Clean dataset

	precision	recall	f1-score	support
0	0.99	1.00	1.00	208
1	0.84	0.89	0.86	202
2	0.87	0.78	0.82	170
3	0.97	0.99	0.98	217
4	1.00	1.00	1.00	203
accuracy			0.94	1000
macro avg	0.93	0.93	0.93	1000
weighted avg	0.94	0.94	0.94	1000

K-Nearest Neighbor

- MFCCs and Chromagram
- Manhattan distance
- class 0, class 3 and class 4 have a near perfect precision score.

Accuracy 93.6%.



Results

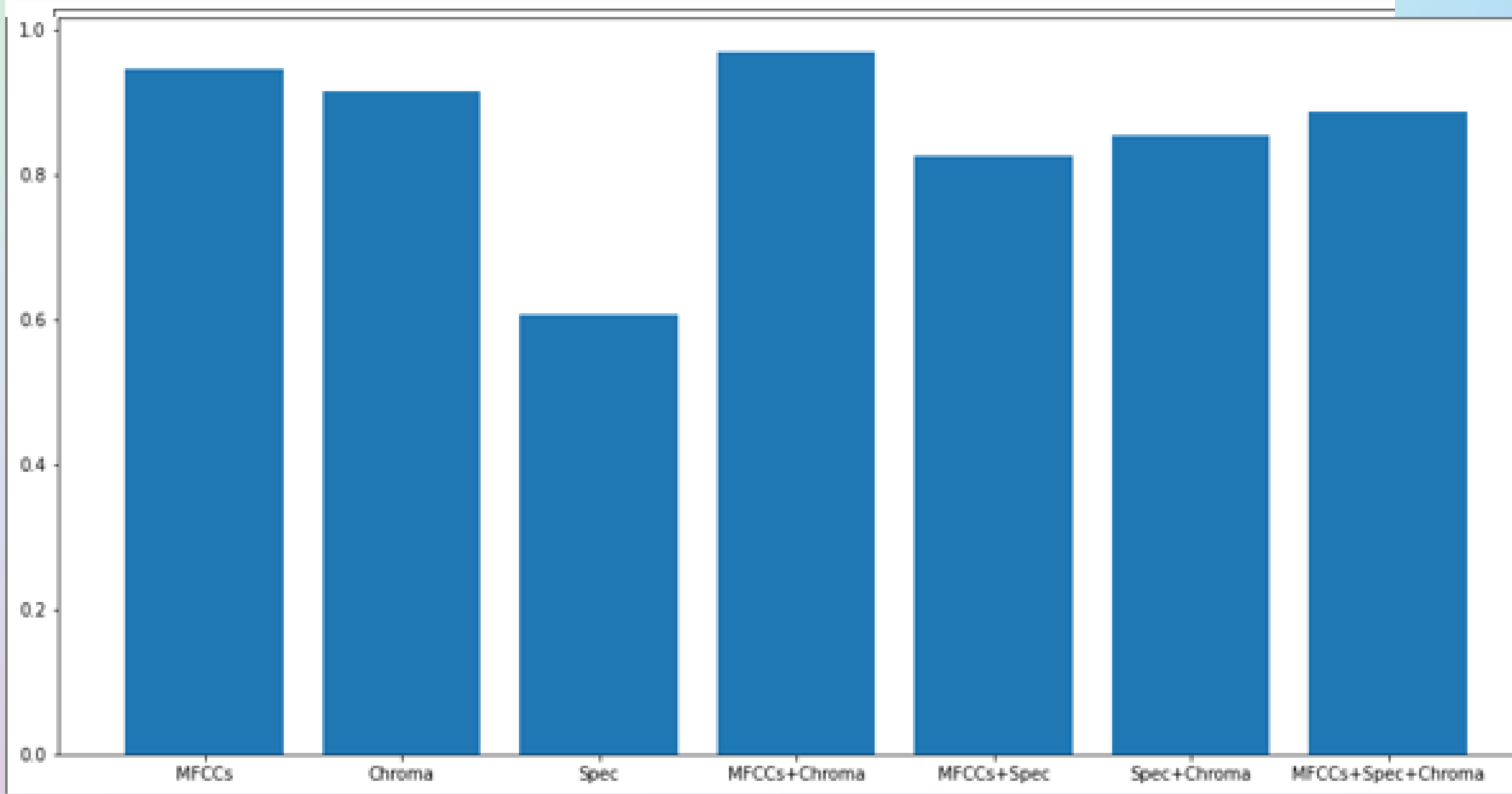
Clean and Augmented dataset combined

	precision	recall	f1-score	support
0	0.99	1.00	0.99	816
1	0.93	0.94	0.93	842
2	0.94	0.91	0.93	795
3	0.98	1.00	0.99	772
4	1.00	0.99	1.00	775
accuracy			0.97	4000
macro avg	0.97	0.97	0.97	4000
weighted avg	0.97	0.97	0.97	4000

K-Nearest Neighbor

- MFCCs and Chromagram
- Manhattan distance
- class 0, class 3 and class 4 have a near perfect precision score.

Accuracy 96.8%.



Results

Clean dataset

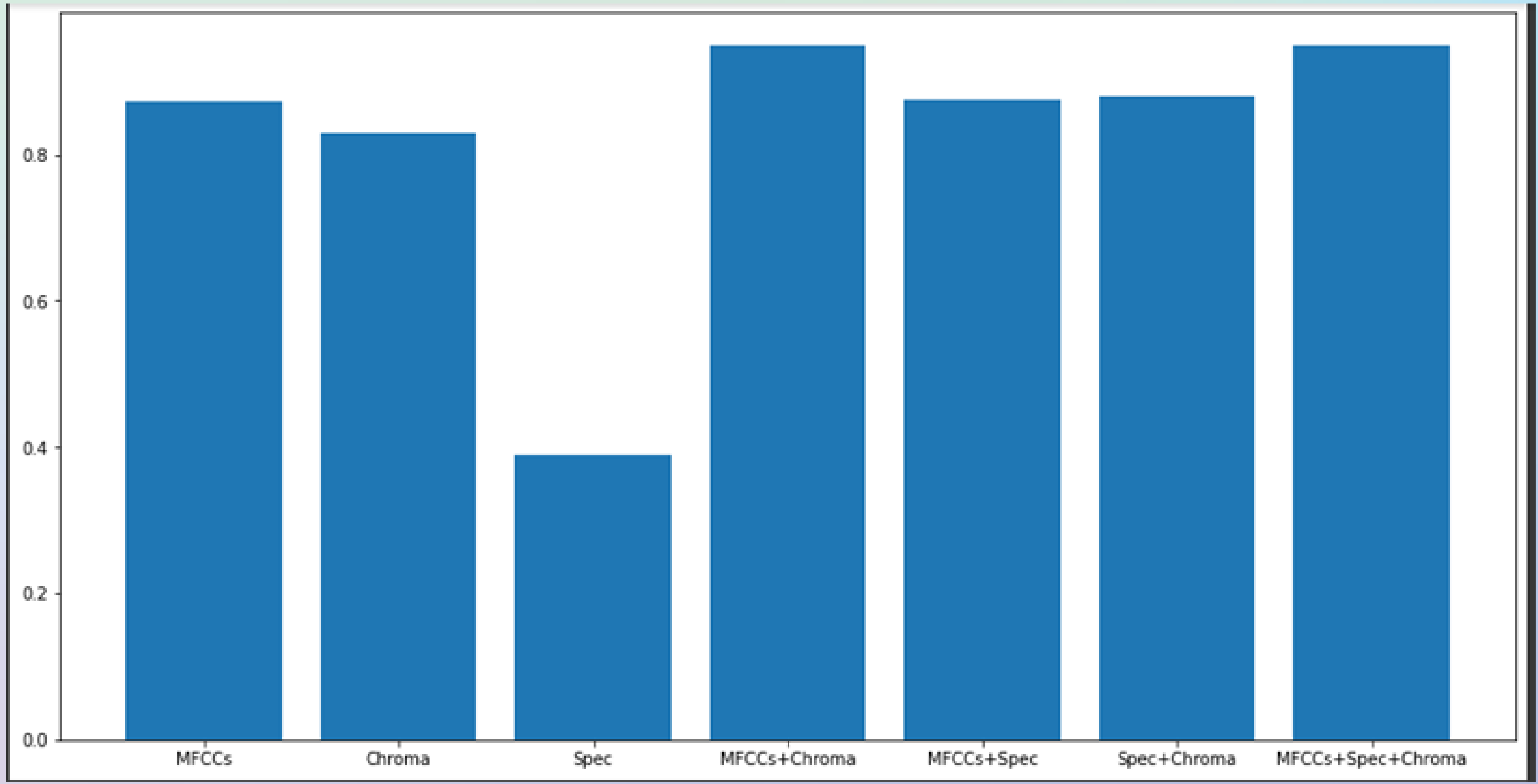
For MFCCs+Chroma
The classifier used is Logistic Regression with the l1 penalty

	precision	recall	f1-score	support
0	1.00	1.00	1.00	208
1	0.90	0.85	0.87	202
2	0.83	0.89	0.86	170
3	0.99	0.99	0.99	217
4	1.00	1.00	1.00	203
accuracy			0.95	1000
macro avg	0.94	0.94	0.94	1000
weighted avg	0.95	0.95	0.95	1000

Logistic Regression

- MFCCs and Chromagram
- L1 penalty

Accuracy 95%.



Results

Clean and augmented dataset Combined

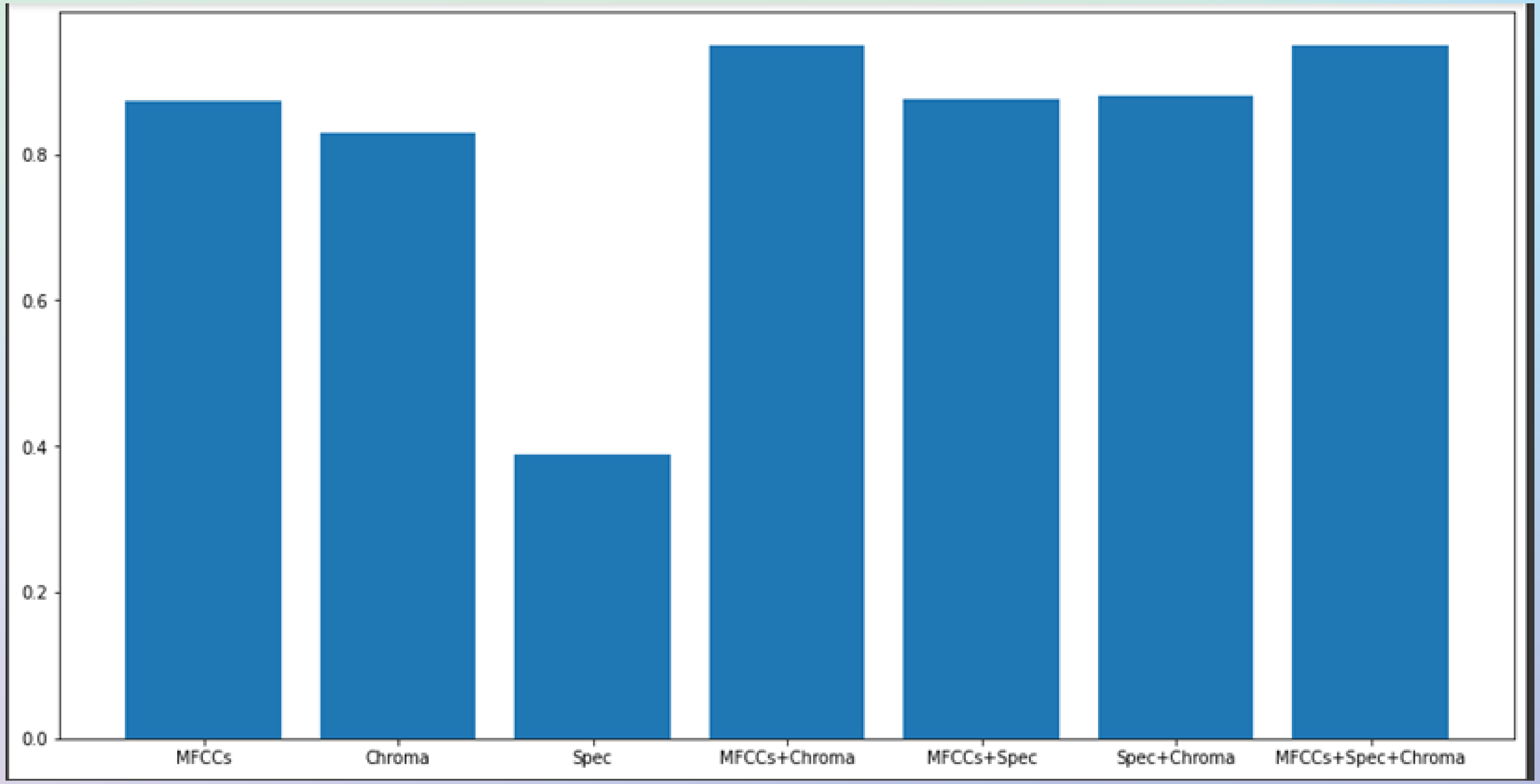
	precision	recall	f1-score	support
0	1.00	1.00	1.00	816
1	0.83	0.86	0.85	842
2	0.85	0.82	0.83	795
3	0.99	0.98	0.98	772
4	1.00	1.00	1.00	775
accuracy			0.93	4000
macro avg	0.93	0.93	0.93	4000
weighted avg	0.93	0.93	0.93	4000

Logistic Regression

- MFCCs and Chromagram
- L1 penalty
- elastic net regression

Accuracy 93.075%

L1 ratio 0.6666



Results

Clean dataset

Naïve Bayes

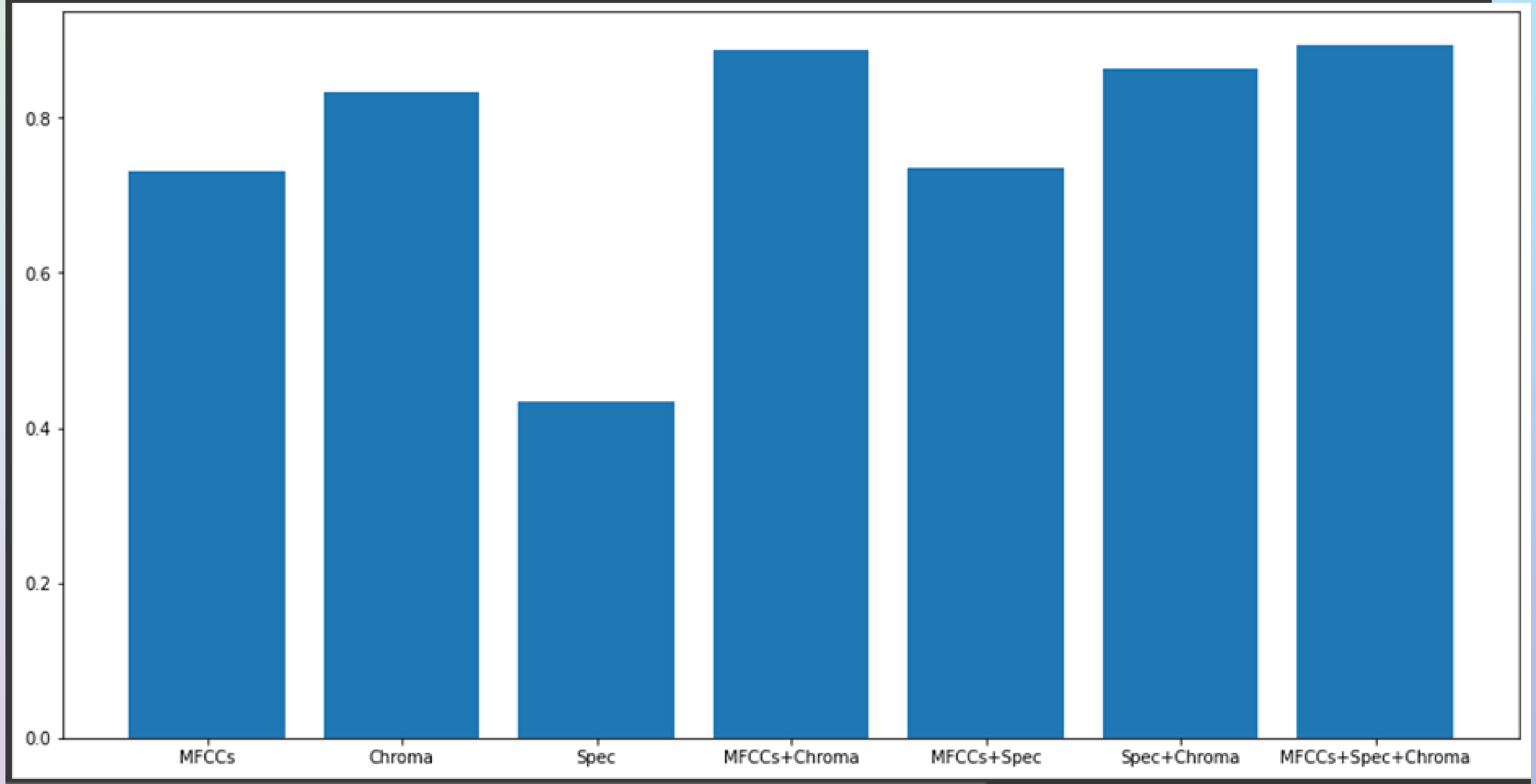
- MFCCs and Chromagram
- Class 0, 1 , 2, 3, and 4

Accuracy 88.8%

```
For MFCCs+Chroma
The classifier used is  Naive Bayes  with accuracy  0.888
              precision    recall  f1-score   support

           0       1.00      0.91      0.96       199
           1       0.77      0.86      0.81       219
           2       0.76      0.75      0.75       187
           3       0.96      0.91      0.93       191
           4       0.99      1.00      0.99       204

    accuracy                           0.89      1000
   macro avg       0.89      0.89      0.89      1000
weighted avg       0.89      0.89      0.89      1000
```



Results

Clean and augmented dataset Combined

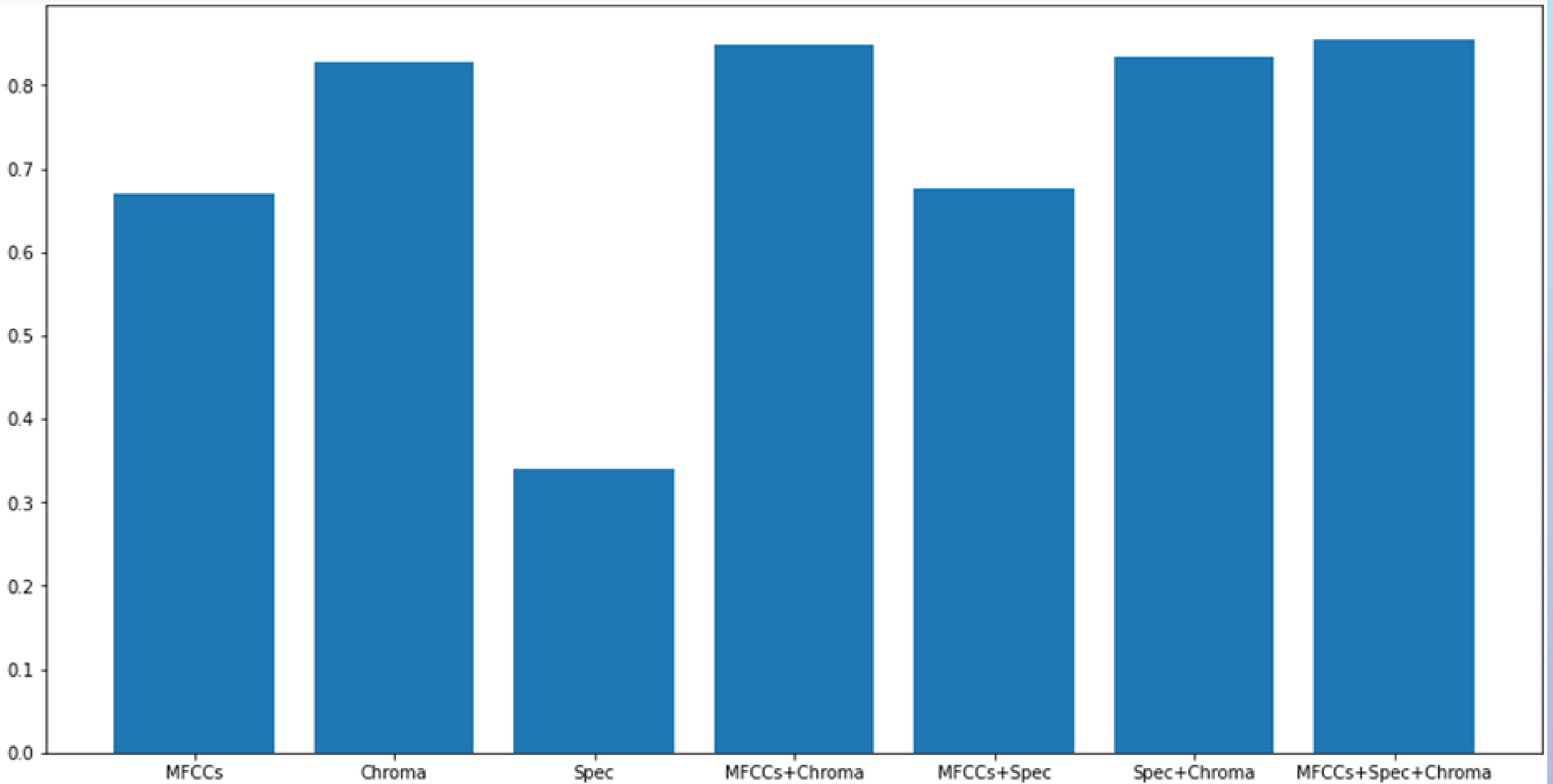
Naïve Bayes

- MFCCs, Spectral Centroid and Chromagram
- Class 0, 1 , 2, 3, and 4
- precision scores of class 1 and 2 are relatively poorer.

Accuracy 85.4 %

For MFCCs+Spec+Chroma
The classifier used is Naive Bayes with accuracy 0.854

	precision	recall	f1-score	support
0	0.97	0.93	0.95	771
1	0.68	0.80	0.74	791
2	0.73	0.65	0.68	857
3	0.96	0.92	0.94	785
4	0.97	1.00	0.98	796
accuracy			0.85	4000
macro avg	0.86	0.86	0.86	4000
weighted avg	0.86	0.85	0.85	4000



Results

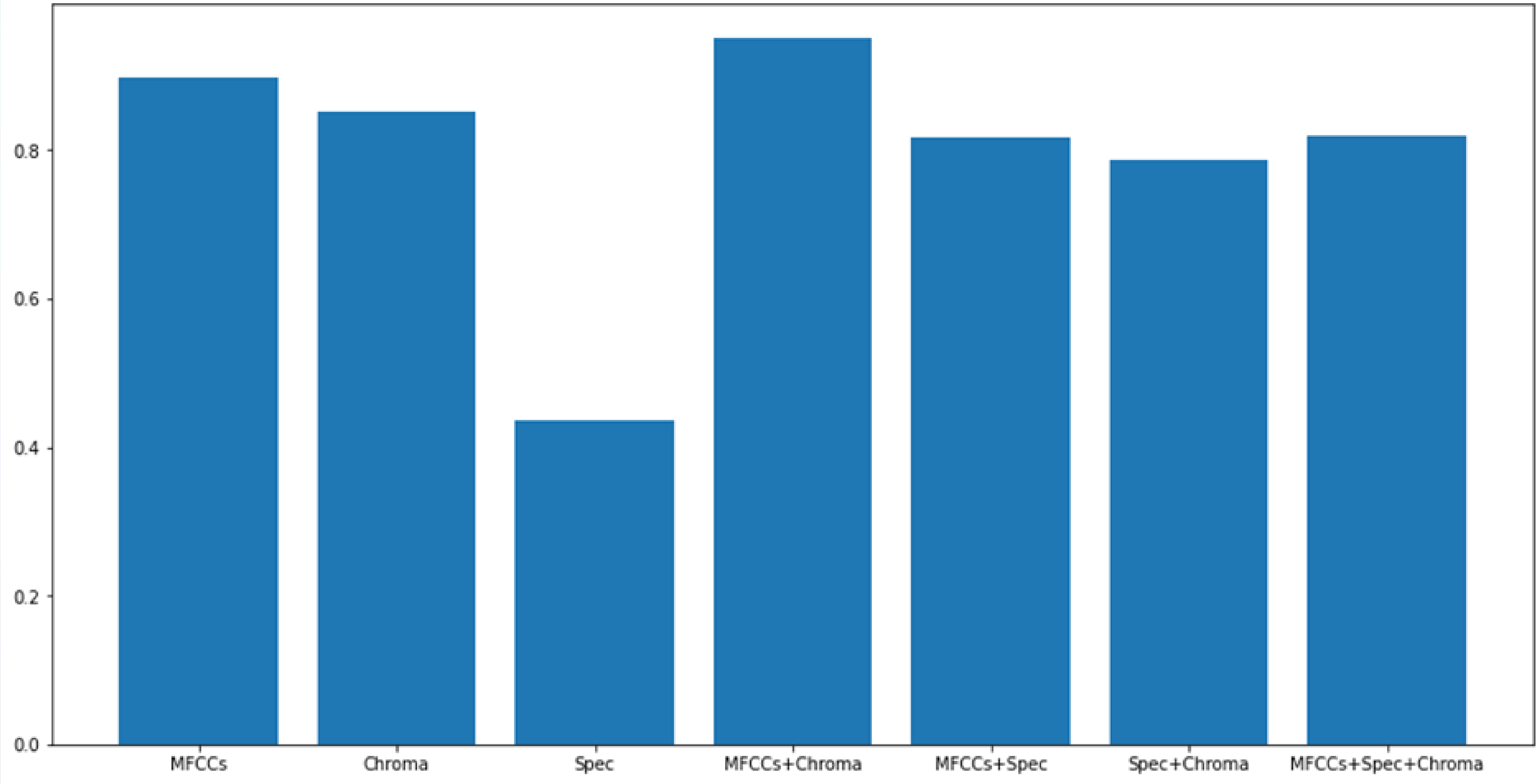
Clean dataset

Support Vector machines

- MFCCs and Chromagram
- Radial basis function as the activation function
- 0.01 regularization parameter
- precision scores of class 1 and 2 are much better as comapred to classes 0,3 and 4.

Accuracy 95%

	precision	recall	f1-score	support
0	1.00	1.00	1.00	204
1	0.89	0.87	0.88	203
2	0.87	0.90	0.88	194
3	1.00	0.99	0.99	207
4	1.00	1.00	1.00	192
accuracy			0.95	1000
macro avg	0.95	0.95	0.95	1000
weighted avg	0.95	0.95	0.95	1000



Results

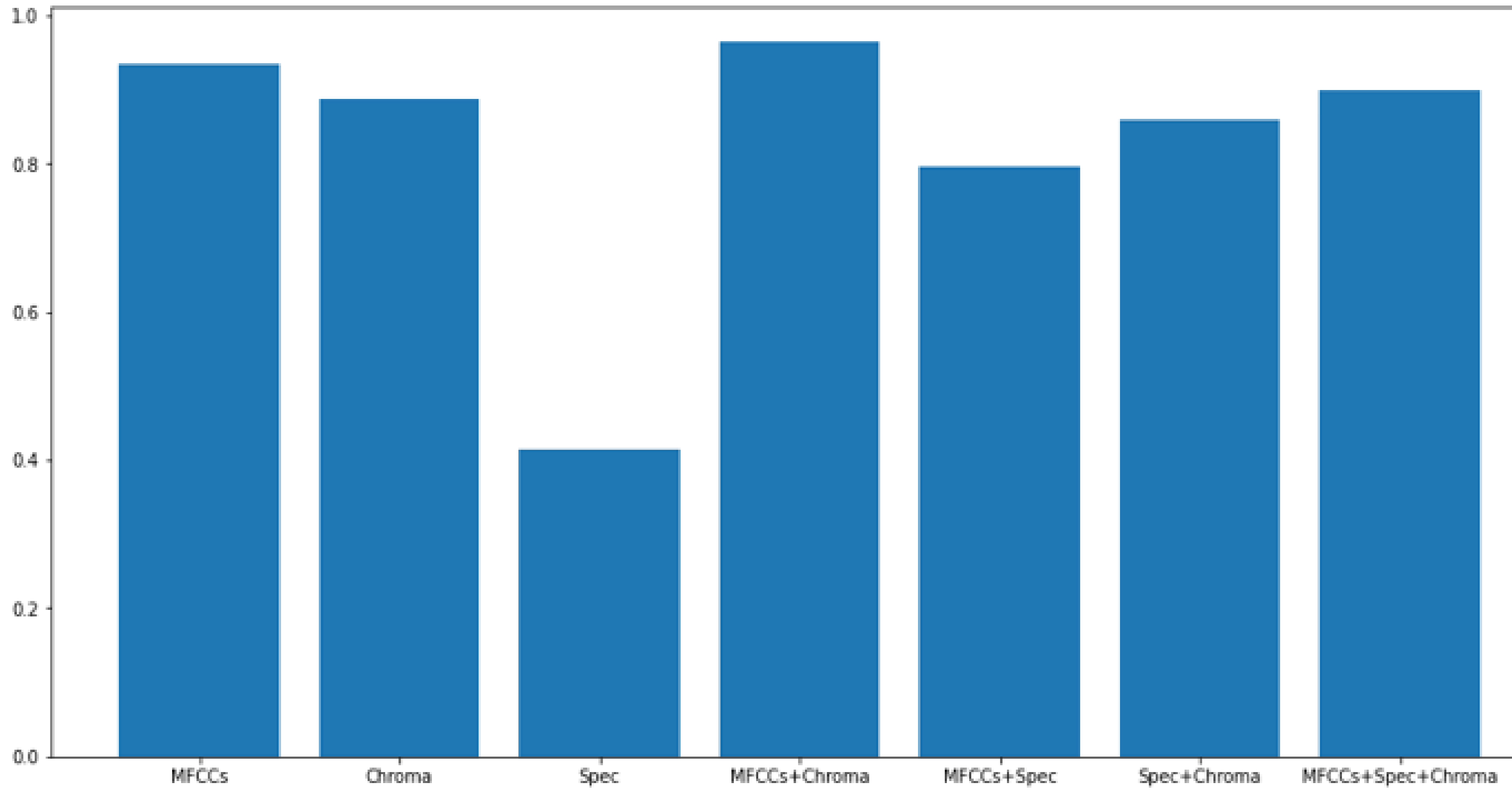
Clean dataset

Support Vector machines

- MFCCs and Chromagram
- Radial basis function as the activation function
- 0.01 regularization parameter
- precision scores of class 1 and 2 are much better as comapred to classes 0,3 and 4.

Accuracy 95%

0	1.00	0.99	0.99	792
1	0.95	0.93	0.94	797
2	0.93	0.93	0.93	816
3	1.00	0.97	0.99	811
4	0.94	1.00	0.97	784
accuracy			0.96	4000
macro avg	0.96	0.96	0.96	4000
weighted avg	0.96	0.96	0.96	4000



Results

Clean dataset

Neural Networks

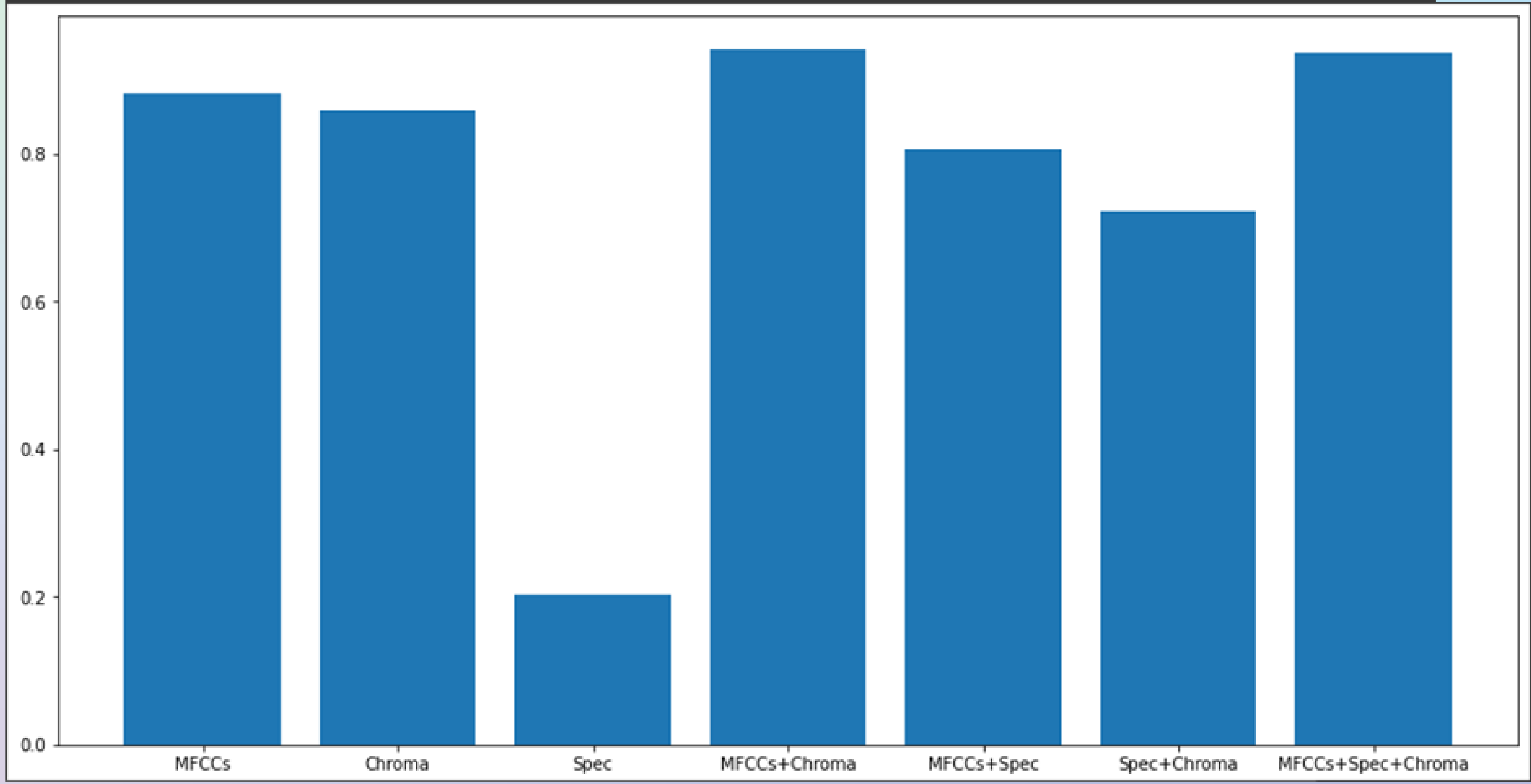
- MFCCs, and Chromagram
- Relu as the activation function
- 0.001 as the initial learning rate
- Number of hidden layers was 12
- Class 1 low precision of 79%, class 2 precision of 86%.

Accuracy 94%

```
For MFCCs+Chroma
The classifier used is  Neural Network  with accuracy  0.94
with the activation function:  relu
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       807
           1       0.79      0.87      0.83       787
           2       0.86      0.79      0.83       799
           3       0.98      0.97      0.98       845
           4       1.00      1.00      1.00       762

    accuracy                           0.92      4000
   macro avg       0.93      0.92      0.92      4000
weighted avg       0.93      0.92      0.92      4000
```



Results

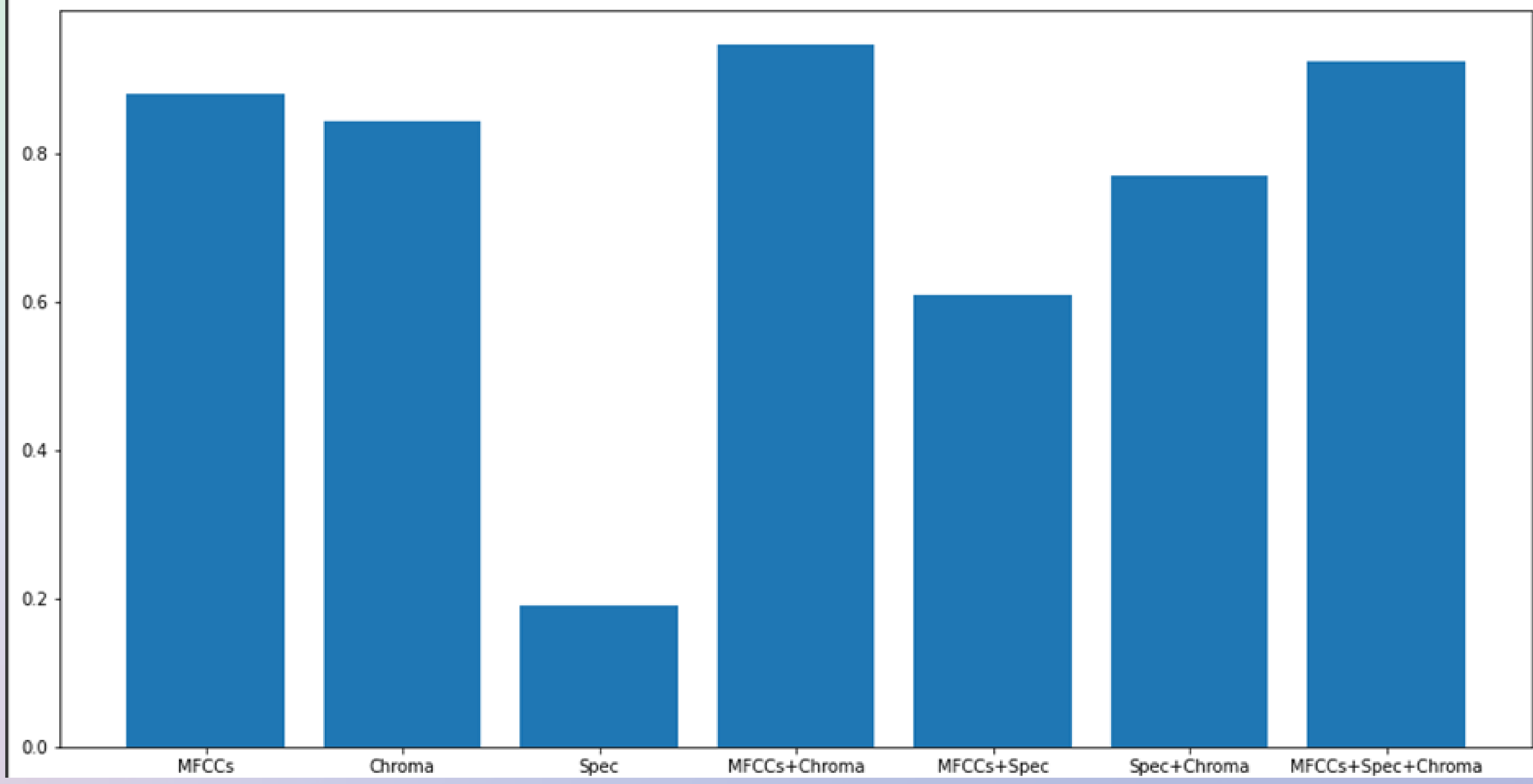
Clean and Augmented dataset combined

Neural Networks

- MFCCs and Chromagram
- Relu as the activation function
- 0.005 as the initial learning rate
- Number of hidden layers was 12

Accuracy 94.625 %

	precision	recall	f1-score	support
0	1.00	1.00	1.00	807
1	0.86	0.88	0.87	787
2	0.89	0.86	0.87	799
3	0.99	0.99	0.99	845
4	1.00	1.00	1.00	762
accuracy			0.95	4000
macro avg	0.95	0.95	0.95	4000
weighted avg	0.95	0.95	0.95	4000



Analysis

The First Analysis of Obtained Results

- Classes 1 and 2 suffer from the worst precision scores as compared to other classes

The Second Analysis of Obtained Results

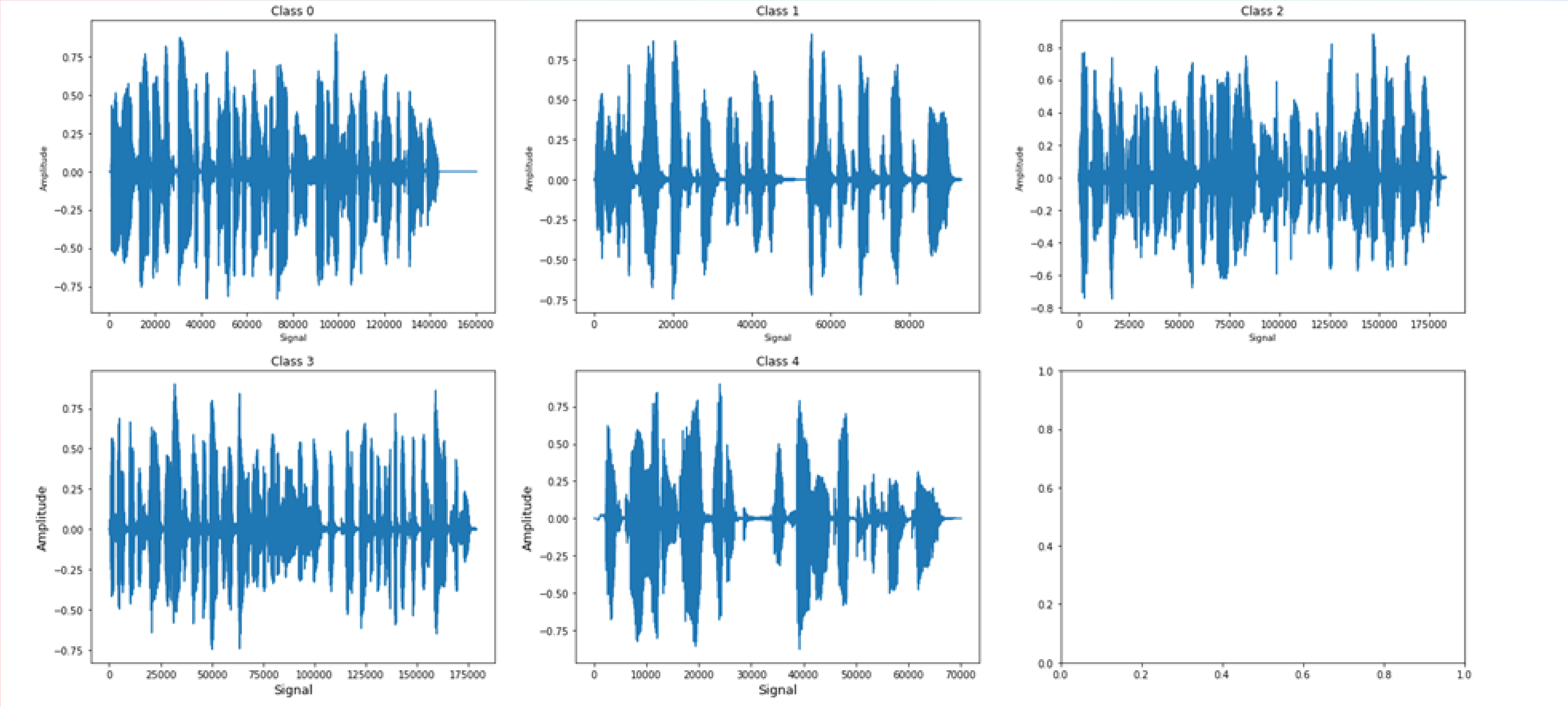
- Highest accuracy we have achieved on the clean data set until now is 95 percent

The Third Analysis of Obtained Results

- A major feature we decided to overlook was Fourier transform, so now we will investigate that feature

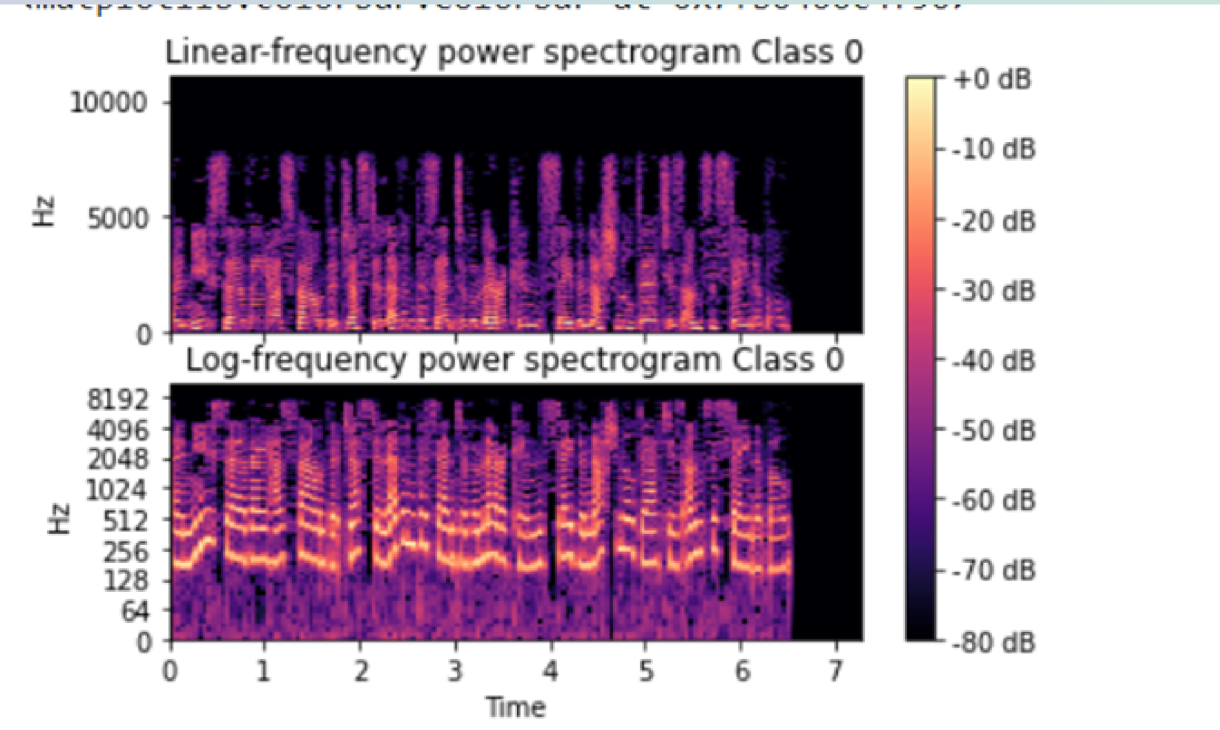
Insights

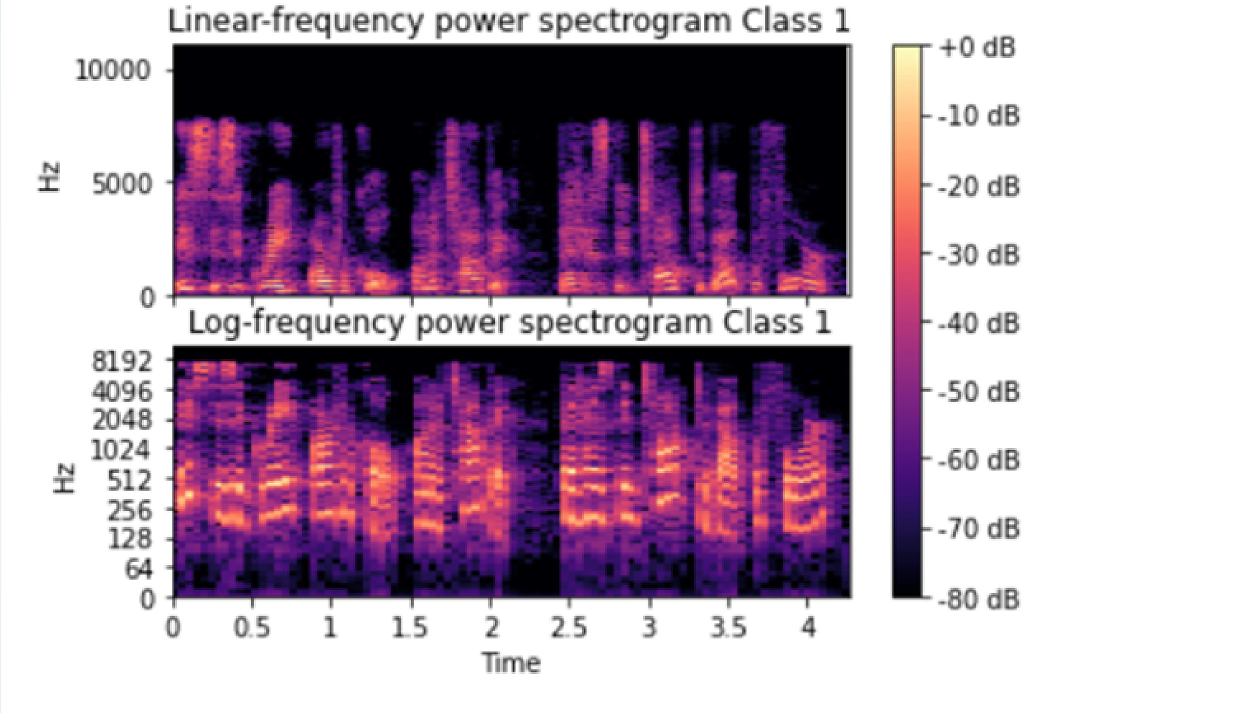
Amplitude/Signal

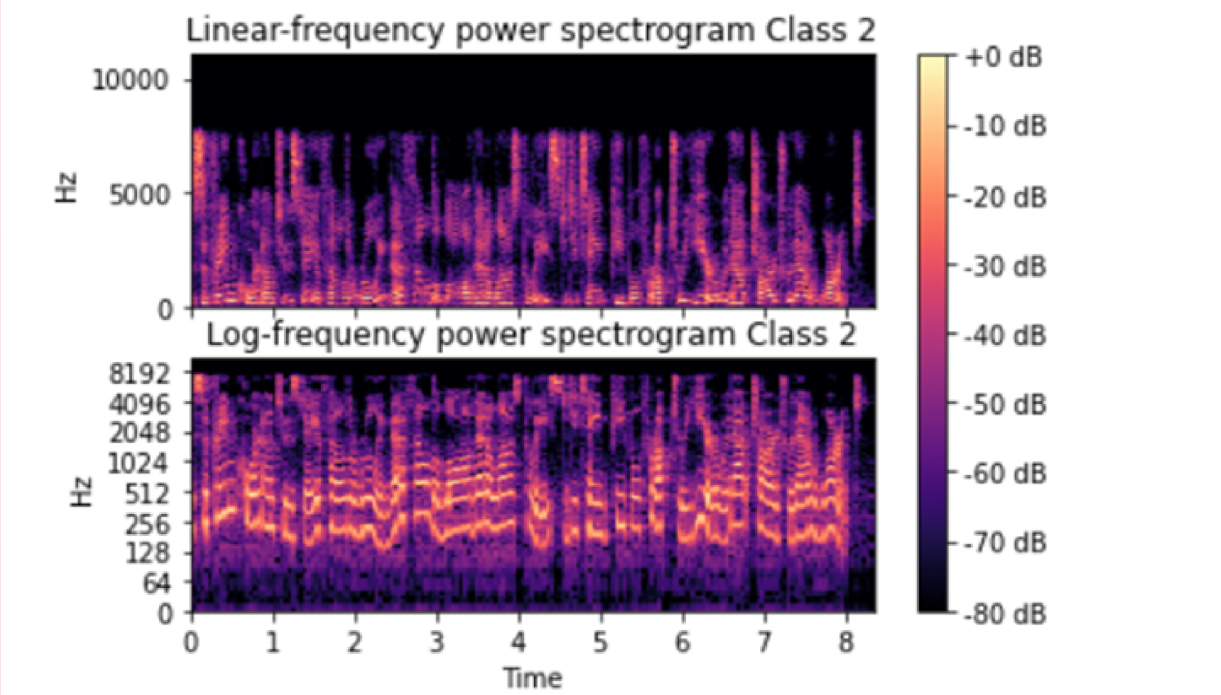


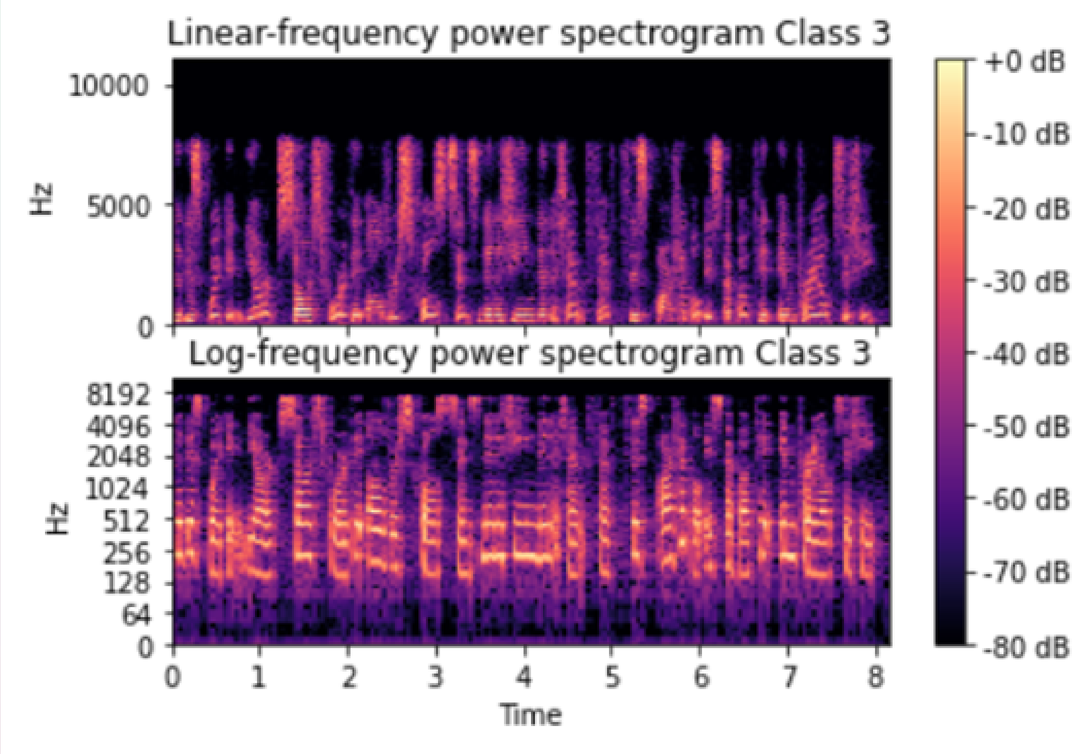
Insights

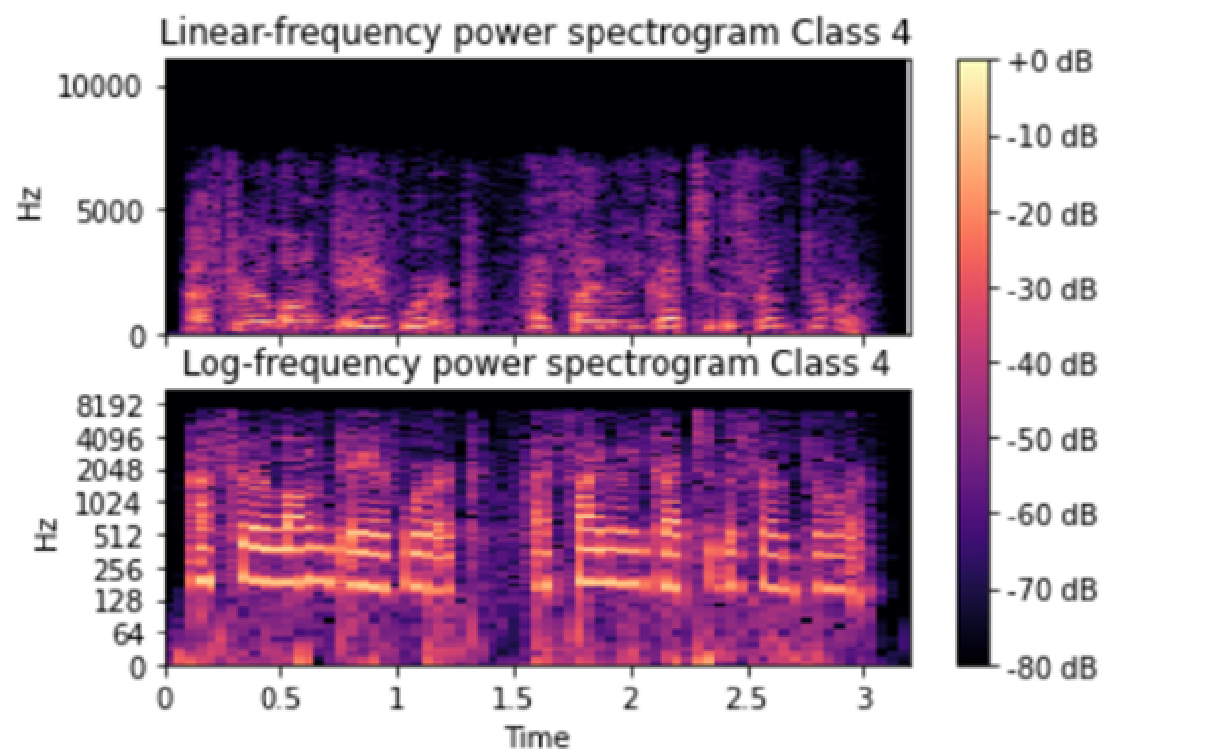
Linear/Log Frequency Power Spectrogram











Conclusion

- Spectral Centroid was not a good feature in terms of synthetic speech algorithm classification
- MFCCs and Chromagram performed the best (in our case)
- Support Vector Machines tend to perform the best with an achieved accuracy of 95% on the clean dataset
- K-Nearest Neighbor performs the best with an achieved accuracy of 96.775% on the augmented + Clean Dataset combined
- Classes 1 and classes 2 suffer from poorer precision scores in KNN algorithm, SVM balances them out a bit with class 4
- Support Vector Machines (which achieved an accuracy of 96.4 on our test data) as the final model