Analysis and Classification of Audio Files Using MFCC Features

Team Members
Jeet Gurbani(23B0023)
Chaitanya Patel(23B028)
Shreeram Jadhav (23B0033)
Falguni Kashyap(23B0058)

Executive Overview:

- In this project, we have to analyze and classify audio files with the help of MFCC (Mel-Frequency Cepstral Coefficients), which is a well-known feature in audio signal processing. Using MFCC, we have to sort and categorize the audio files by different groups, including specific categories & artists.
- Classification and Grouping Objectives:

1. **Genre-Based Grouping:** Organize the dataset of 116 audio files into broad genre categories. Specifically, these categories include:
    - National Anthem songs
    - Marathi Bhav Geet (emotive, poetic songs)
    - Marathi Lavni (traditional, rhythm-driven songs)
    - Hindi Film Songs (by renowned Bollywood artists)
    - English Pop Songs (e.g., by Michael Jackson)
2. **Artist-Based Identification:** Within the Hindi and English song categories, further classify files by the artist, aiming to:
    - Identify solo songs by Asha Bhosale and Kishor Kumar (Hindi songs)
    - Identify English pop songs by Michael Jackson

Basic workflow of the project :

- Researched about MFCCs to get insights into how they work and which factors to extract for efficient classification.
- Did exploratory data analysis and PCA to find out how MFCCs are distributed and how much data explains most of the variation in data.
- Used 2 different approaches to predict the files according to the given sections.
- Extracted different features from MFCCs to get useful and effective data.

- **Unsupervised Learning**
  - Used different models to cluster given unlabeled songs and selected the best model.
  - Trained these songs and labels as cluster numbers with different classification models.
  - Tested known labeled songs to find which cluster is related to which song section.

- **Supervised Learning**
  - Downloaded 600+ labeled songs and trained them using Neural network and SVM models.
  - Selected the model with the best confidence.
  - Tested the given songs to predict which section they belong to.

What Are MFCCs ?

- **Definition:** Mel-Frequency Cepstral Coefficients (MFCC) are a set of coefficients that reflect the short-term power spectrum of sound, inspired by how the human ear perceives frequencies.
- **Process:** Extracts essential frequency information from audio by replicating the Mel scale, which points out frequencies that are most significant to human perception.
- **Uses of MFCC**
  - **Speech and Audio Recognition:** Used extensively in speech processing (e.g., voice recognition) and music genre classification.
  - **Emotion and Instrument Classification:** Assists in distinguishing auditory features, identifying instruments, and even detecting emotional tones in speech.

Why Use MFCC for Music Genre Classification?

- **Models Human Perception of Sound:** Because MFCC is based on the Mel scale (intended to mirror how humans hear frequencies), it can capture aspects in music that are significant for genre differentiation, such as the emphasis on specific rhythms or pitches in various styles.
- **Reduces Complexity and Focuses on Key Features:** The MFCC Coefficients re sufficient to convey a song's overall tone, rhythm, and texture, which are frequently what distinguishes one genre from another.
- **Reliably Represents Musical Characteristics:** By capturing aspects like timbre (the unique quality of a sound), rhythm patterns, and harmonic content (such melody and chords), MFCC gives a solid foundation for identifying

MFCC Extraction Process

- **Pre-Emphasis:** Amplifies high frequencies to balance sound.
- **Frame Blocking:** Splits audio into short frames to capture changes over time.
- **Windowing:** Applies a Hamming window to reduce edge effects.
- **Fourier Transform:** Converts frames to frequency data.
- **Mel Filter Bank:** Focuses on frequencies humans hear best.
- **Log Amplitude:** Scales frequencies to mimic human loudness perception.
- **Discrete Cosine Transform (DCT):** Produces MFCCs by compressing the most important sound features.

What each MFCC Coeff Represents

- **Coefficient 0 (Energy or Loudness):** This measures how loud or soft the sound is overall.
- **Coefficients 1-2 (Low-Frequency Sounds):** These capture deep sounds, like the bass or lower tones in music.
- **Coefficients 3-5 (Middle-Frequency Sounds):** These are linked to voices or mid-range instruments, like guitars or pianos.
- **Coefficients 6-8 (Bright Sounds):** These represent higher sounds, like cymbals or sharp vocal tones.
- **Coefficients 9-12 (High-Frequency Details): T**hese pick up small, high-pitched sounds that make music clearer and sharper.
- **Coefficients 13-19 (Fine Sound Details):** These are tiny details in the sound that help to distinguish one genre from another.

How MFCCs Help in Music Genre Classification

- **Representing Sound Characteristics**: MFCCs extract important sound characteristics like pitch, loudness, and texture, which are key to identifying a song's genre.
- **Unique Fingerprint for Each Genre:** Every genre has its own sound traits. For example, jazz has complex harmonies, while rock focuses on strong beats. MFCCs capture these unique traits..
- **Comparison Across Songs:** MFCCs help machine learning models identify similarities between songs of the same genre and differences between genres, allowing for genre prediction.
- **Practical Example:** Songs like Lavnis, Janaganamana, Bhaavgeet, Hindi songs by artists, and Michael Jackson's tracks will show different MFCC patterns, enabling the system to classify them correctly.

EDA:

- Each CSV file contains MFCC coefficients for a song. So first, we loaded the MFCC data.
- Then for each song, calculated mean and variance of each MFCC coefficient, which resulted 40 features per song (20 means and 20 variances). The data is stored in a DataFrame for further analysis.

Summary Statistics:

	MFCC_mean_1	MFCC_mean_2	MFCC_mean_3	MFCC_mean_4	MFCC_mean_5	\
count	116.000000	116.000000	116.000000	116.000000	116.000000	
mean	-223.685816	149.294454	-39.298437	33.742982	3.276054	
std	48.179351	30.305888	24.148395	22.530922	15.857214	
min	-446.827918	91.387647	-93.961705	-28.187547	-30.221894	
25%	-257.472932	127.234253	-56.828854	25.974569	-6.874532	
50%	-221.786107	141.293248	-43.571203	38.870333	-0.112606	
75%	-191.291758	166.712191	-20.156386	48.312487	11.068414	
max	-106.734410	228.158156	26.270366	79.220040	47.968517	

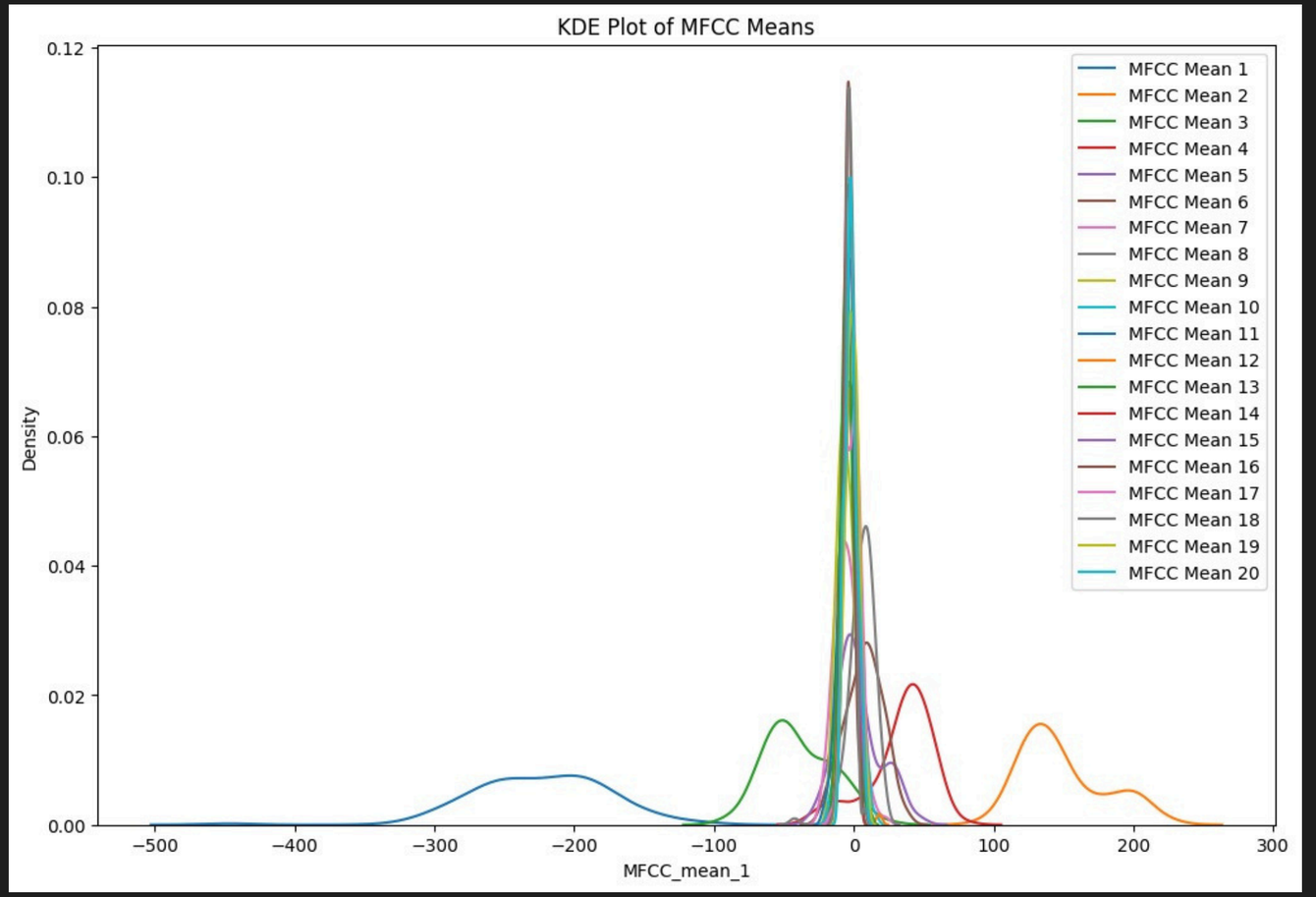
	MFCC_mean_6	MFCC_mean_7	MFCC_mean_8	MFCC_mean_9	MFCC_mean_10	...	\
count	116.000000	116.000000	116.000000	116.000000	116.000000	...	
mean	7.480534	-5.404420	5.753877	-5.975003	-1.893513	...	
std	13.167991	8.461683	9.112946	5.820130	5.835531	...	
min	-22.647828	-23.358137	-42.326281	-18.593659	-17.767540	...	
25%	-2.146836	-10.898134	1.184934	-10.351618	-5.271525	...	
50%	7.976706	-5.594203	7.115966	-6.390081	-1.836311	...	
75%	16.625103	-0.506028	12.001604	-2.071938	1.501222	...	
max	36.890027	22.871955	21.455668	7.926159	15.474307	...	

	MFCC_var_11	MFCC_var_12	MFCC_var_13	MFCC_var_14	MFCC_var_15	\
count	116.000000	116.000000	116.000000	116.000000	116.000000	
mean	105.514584	101.550034	99.512944	95.887413	98.573682	
std	32.523468	29.054716	33.161626	34.211065	44.275695	
min	45.572870	47.990367	50.936310	43.236622	40.079408	
25%	80.453537	79.683944	74.022349	68.953277	65.363883	
50%	100.636527	96.876630	90.665287	88.412309	89.673359	
75%	124.229402	119.418982	116.656996	117.020489	115.648239	
max	212.223071	180.154574	234.869263	221.346609	251.975661	

	MFCC_var_16	MFCC_var_17	MFCC_var_18	MFCC_var_19	MFCC_var_20
count	116.000000	116.000000	116.000000	116.000000	116.000000
mean	102.234562	104.239310	100.628588	101.126922	103.861936
std	47.147035	50.696084	49.716025	53.233671	55.479172
min	41.187895	39.795873	35.327652	33.952562	28.544018
25%	67.475588	66.843320	64.301603	58.828528	59.360446
50%	88.085483	90.934933	89.452215	86.659493	88.234666
75%	130.286164	133.379819	122.942752	129.405738	131.026345
max	284.649206	314.784869	286.379382	289.551162	298.937040

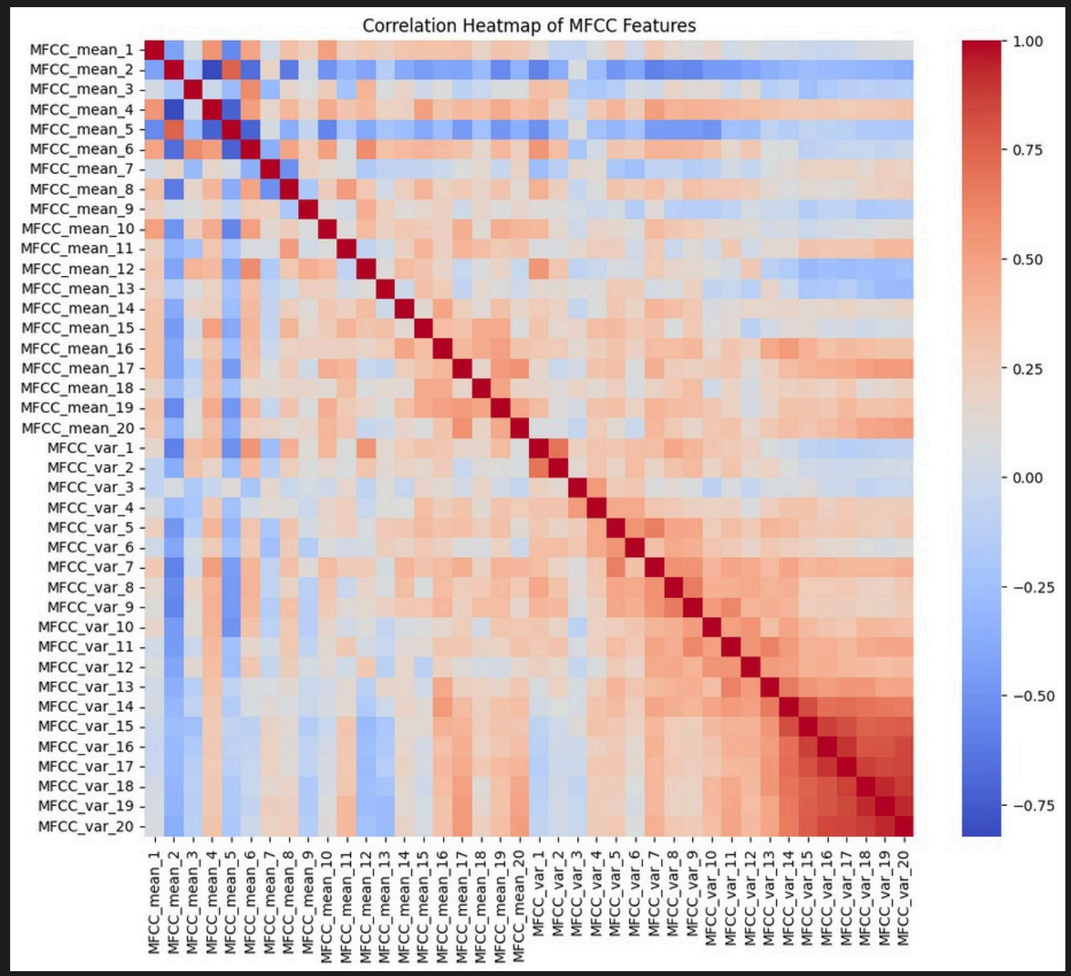
EDA:

- We then used **KDE (Kernel Density Estimation)** plots to show the distribution of MFCC means for each coefficient. This helped visualize how each MFCC coefficient's mean varies across songs and gave insights into overall audio characteristics.



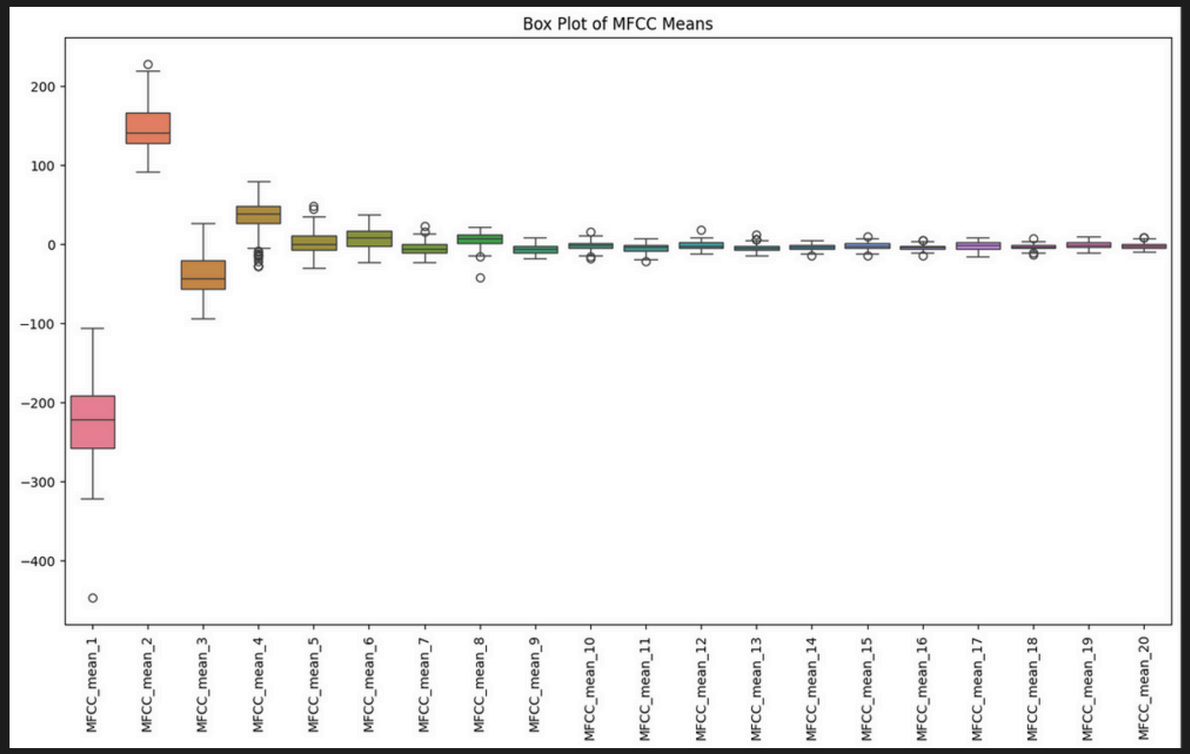
EDA:

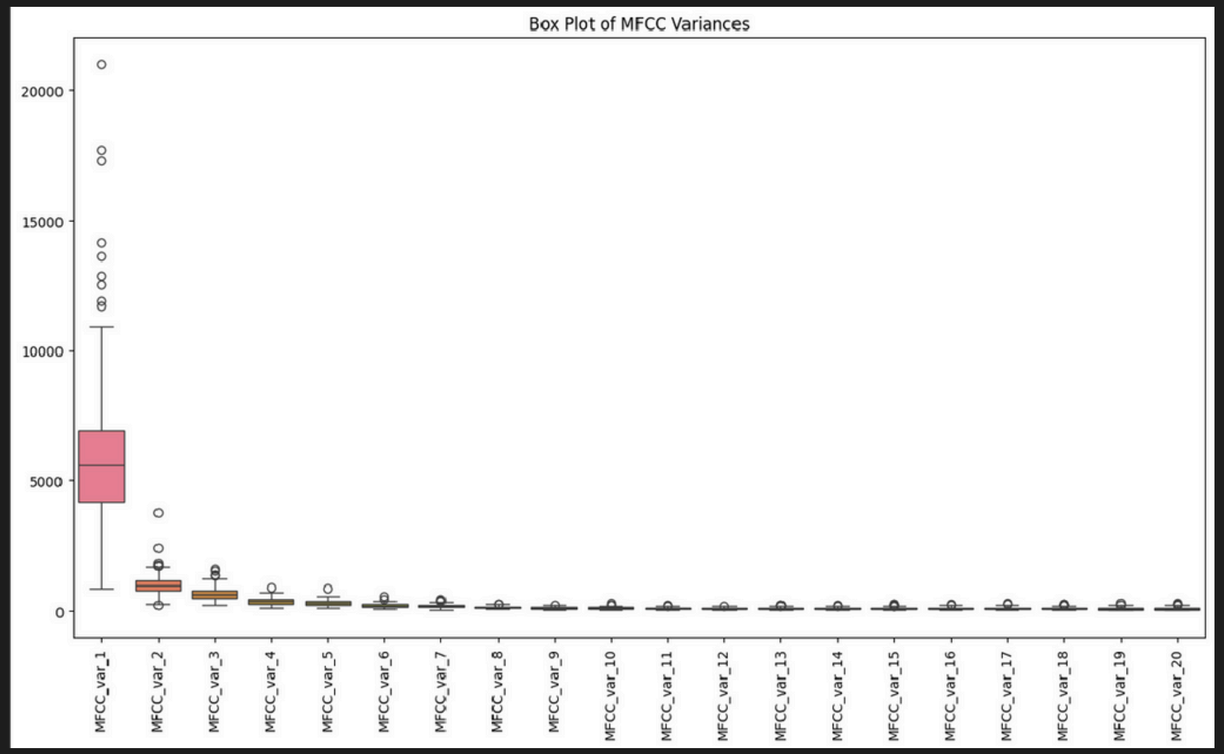
- After that, we computed and visualized a **correlation heatmap** plot to understand relationships between MFCC features. It helps identify which MFCC coefficients are strongly correlated, which can inform feature reduction.



EDA:

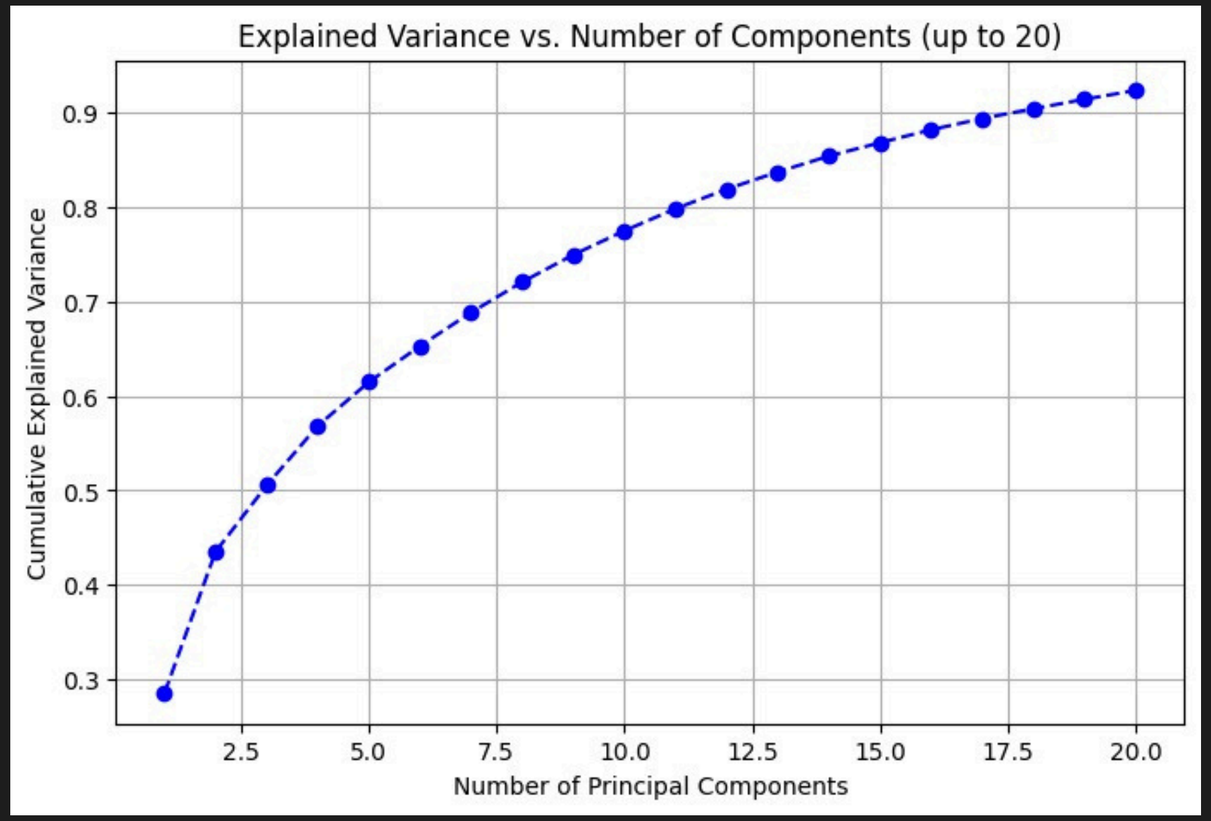
- Next up, we did **outlier detection** in MFCC Means & Variances to identify unusual data points in them. We used box plots to show distributions of MFCC means and variances across songs.
- Outliers appear as individual points outside the whiskers of each box, which helps us quickly spot coefficients with values significantly different from the norm. Outliers suggest specific songs or segments that have unique audio features, e.g. unusual pitch, timbre, or intensity.





PCA:

- Moving on, we applied **PCA (Principal Component Analysis)** to reduce dimensionality while retaining the most variance in data.
- Then we plotted **cumulative explained variance** to show how much information (variance) is retained as the number of principal components increases. This plot helped determine the minimum number of components needed to capture a significant portion of the data's variance.



How will Un-supervised approach work:

- We used different clustering models to divide the given unlabeled songs into six clusters.
- Using various metrics we picked the best model results.
- Then we trained different models on these clusters and labeled as cluster numbers and compared these models using various metrics.
- After that, we created a dataset of 10 songs for each of these 6 types, and converted them into .csv file containing mfcc coefficients, and predicted each song using the best model that we trained.
- Then saving each genre or artist specific song into different folder, we performed validation about how many song in each folder belong to which cluster.
- By this, we got to know about which cluster belong to which genre or artist, and thus retrieved detail about which .csv file corresponds to what type of song or artist.

Feature Extraction:

- We loaded the MFCC data from a CSV file and for each row of MFCC data, computed statistical features like
  - **Mean**- The average value of the MFCC coefficient data
  - **Standard Deviation**- A measure of spread or dispersion of the MFCC coefficient values
  - **75th Percentile**- The value below which 75% of the data points fall
  - **Max/Min**- The highest/lowest value in the MFCC coefficient data
  - **Kurtosis**- A measure of how peaked or flat a data distribution is
  - **Skewness**- A measure of the asymmetry or skew in the distribution of the data
  - **Spectral Bandwidth**- An indication of how the energy is distributed across frequencies
  - **RMS**- The root mean square of the signal, representing its energy or loudness
- These features capture the general properties of each MFCC coefficient.
- Then we combined the extracted statistical and audio features into a single feature vector. This vector represents a compact summary of the MFCC data for further analysis.

Clustering Models :

- To see which model would perform best in clustering, we checked 3 metrices over 3 different models i.e., GMM, Kmeans and Agglomerative.

1. **Gaussian Mixture Model (GMM):** GMM assumes that the data is generated from a mixture of several Gaussian distributions and assigns each data point to a cluster based on the probability of belonging to each distribution. It uses probabilistic assignments to clusters (each data point can belong to multiple clusters with different probabilities).
2. **K-Means Clustering:** K-means divides data into K clusters by assigning each data point to the nearest cluster center (centroid) and then updating the centroids until convergence. It minimizes the variance within each cluster.
3. **Agglomerative Clustering:** It starts with each data point as its own cluster and then iteratively merges the closest clusters based on a similarity measure. It builds a hierarchical tree of clusters.

Metrices used to compare:

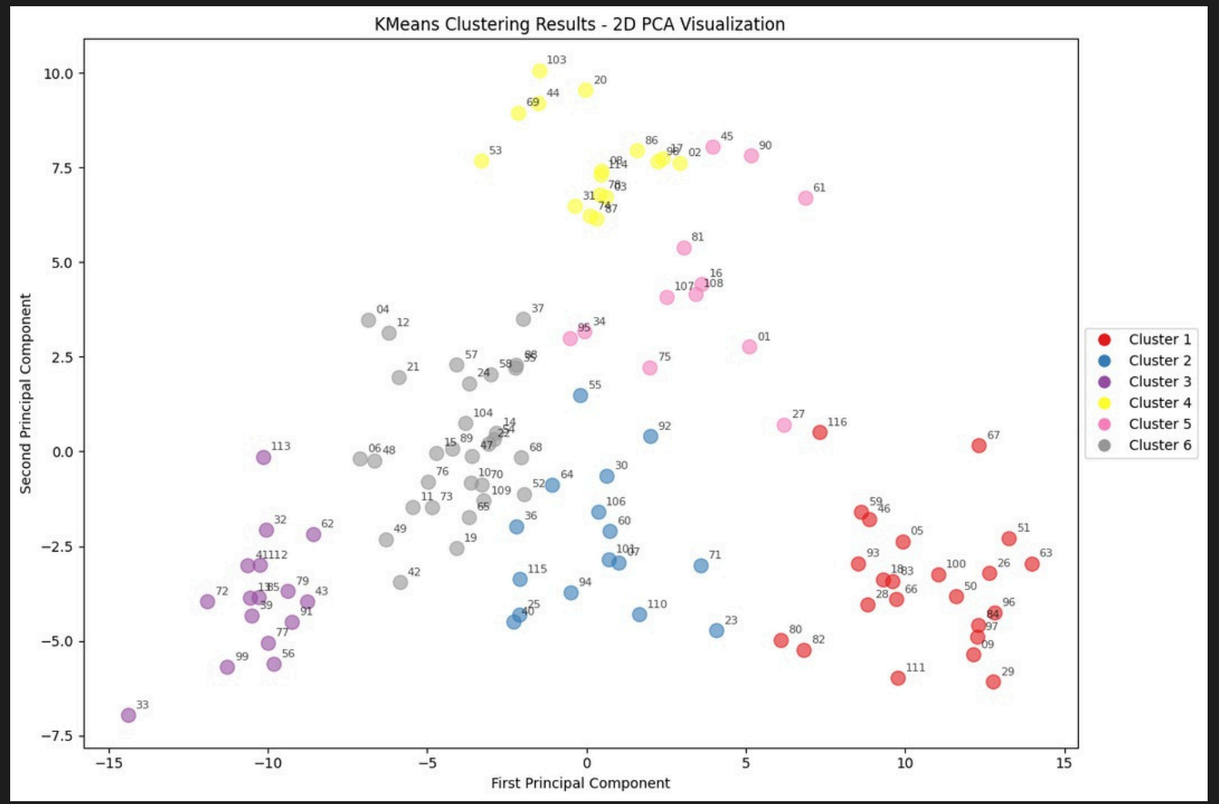
- **Silhouette score** measures how similar each point is to its own cluster. It ranges from -1 to 1 and a higher value represents a better model.
- **Davies-Bouldin** Score quantifies average cluster similarity, with lower values indicating more distinct clusters.
- **Calinski-Harabasz** Index relates the dispersion within and between clusters. Higher values reflect well-separated and compact clusters.

```
GMM Silhouette Score: 0.4481
GMM Davies-Bouldin Score: 0.8305
GMM Calinski-Harabasz Index: 152.1893
GMM clustering results saved to 'results_gmm.csv'

KMeans Clustering Metrics:
Silhouette Score: 0.4769
Davies-Bouldin Score: 0.7677
Calinski-Harabasz Index: 206.9050

Agglomerative Clustering Metrics:
Silhouette Score: 0.4698
Davies-Bouldin Score: 0.7225
Calinski-Harabasz Index: 186.4346
```

Clustering Model chose :

- As we can see, **KMeans** showed the best results, so we proceeded to use KMeans clustering. The results are shown here.
- We reduced dimension to two for sake of plotting curve between two components, but all the 180 features were used for clustering.
- NOTE: We removed csv ‘38’, ‘102’ and ‘105’ as they were acting as outlier in clustering result.
- We then created an array that maps each CSV file to its corresponding cluster, snippet of the array is shown below for reference.

```
array([['01-MFCC.csv', 4],
       ['02-MFCC.csv', 3],
       ['03-MFCC.csv', 3],
       ['04-MFCC.csv', 5],
       ['05-MFCC.csv', 0],
       ['06-MFCC.csv', 5],
       ['07-MFCC.csv', 1],
       ['08-MFCC.csv', 3],
       ['09-MFCC.csv', 0],
       ['10-MFCC.csv', 5],
```



Classification :

- We then converted the array into a dictionary, where the key represents the file name and the value corresponds to the cluster number.
- Next, we created two lists: X to store the feature vectors for each file and y to store the corresponding cluster labels. Both lists were then converted into arrays for easier manipulation in the following steps
- We proceeded to split the X and y arrays into train and test datasets using an 80:20 ratio to evaluate the performance of 3 different classifiers: **MLP, Random Forest, and Gradient Boosting**. We compared their performance across 3 key metrics: **Accuracy, F1 Score & Confusion Matrix**, along with a detailed classification report.
- We also used GridSearchCV to find the best hyperparameters for each classification model.

Comparing Classification Models:

- The 3 classifier models used are
  - **MLPClassifier (Multi-Layer Perceptron)** - Neural Network
  - **Random Forest Classifier** - Ensemble of decision trees
  - **Gradient Boosting Classifier** - Boosted decision trees
- The metrics displayed are
  - **Accuracy**: Measures the proportion of correctly predicted samples
  - **Precision**: The ratio of true positive predictions to total positive predictions
  - **Recall**: The ratio of true positive predictions to total actual positive instances.
  - **F1 Score**: The harmonic mean of Precision and Recall

```
MLP Validation Accuracy (Base): 0.5217391304347826
Best MLP Validation Accuracy: 0.2608695652173913

Best MLP Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         5
           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         3
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         3
           5       0.26      1.00      0.41         6

    accuracy                           0.26        23
   macro avg       0.04      0.17      0.07        23
weighted avg       0.07      0.26      0.11        23
```

Comparing Classification Models(contd.):

Random Forest Validation Accuracy (Base): 0.8695652173913043

Best Random Forest Validation Accuracy: 0.8695652173913043

Best Random Forest Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5
1	1.00	1.00	1.00	3
2	0.75	1.00	0.86	3
3	0.60	1.00	0.75	3
4	1.00	0.33	0.50	3
5	1.00	0.83	0.91	6
accuracy			0.87	23
macro avg	0.89	0.86	0.84	23
weighted avg	0.92	0.87	0.86	23

Gradient Boosting Validation Accuracy (Base): 0.6086956521739131

Best Gradient Boosting Validation Accuracy: 0.782608695652174

Best Gradient Boosting Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5
1	1.00	1.00	1.00	3
2	0.75	1.00	0.86	3
3	0.00	0.00	0.00	3
4	1.00	0.33	0.50	3
5	0.67	1.00	0.80	6
accuracy			0.78	23
macro avg	0.74	0.72	0.69	23
weighted avg	0.75	0.78	0.73	23

Comparing Classification Models(contd.):

We also computed the **confusion matrix** for all the three models to evaluate the model's performance by visualizing the number of correct and incorrect predictions across each class.

```
Confusion Matrix for MLP:
 [[0 0 0 0 0 5]
 [0 0 0 0 0 3]
 [0 0 0 0 0 3]
 [0 0 0 0 0 3]
 [0 0 0 0 0 3]
 [0 0 0 0 0 6]]
```

```
Confusion Matrix for Random Forest:
 [[5 0 0 0 0 0]
 [0 3 0 0 0 0]
 [0 0 3 0 0 0]
 [0 0 0 3 0 0]
 [0 0 0 2 1 0]
 [0 0 1 0 0 5]]
```

```
Confusion Matrix for Gradient Boosting:
 [[5 0 0 0 0 0]
 [0 3 0 0 0 0]
 [0 0 3 0 0 0]
 [0 0 0 0 0 3]
 [0 0 1 1 1 0]
 [0 0 0 0 0 6]]
```

Comparing Classification Models(contd.):

- After that, we calculated the overall performance summary i.e., compared the best accuracy and f1 score of 3 used classifier models to find out the best among them.
- From this, we concluded that **Random Forest** is the best model for classification as it has higher accuracy and f1 score.

```
--- Overall Model Performance Summary ---

MLP - Best Accuracy: 0.2609, F1 Score: 0.1079
Random Forest - Best Accuracy: 0.8696, F1 Score: 0.8598
Gradient Boosting - Best Accuracy: 0.7826, F1 Score: 0.7335
```

Classification Using Random Forest Model:

- We did Random Forest Classification optimized with the best parameters i.e., ‘n_estimators’: 200, 'max_depth': 10, 'min_samples_split': 2, 'random_state': 42.
- Next up, we used **Stratified K-Fold**(with 5 splits) **Cross-Validation** to assess performance. In scoring Metrics, we
- displayed **accuracy, F1 Score (Weighted), ROC AUC** and compared them with that of Random Forest Model.
- Finally, we trained the model on the full training set after cross-validation to ensure optimal performance and used joblib to save the model, enabling quick loading and reusability.

```
Cross-Validation Results:
Accuracy - Train: 1.0000, Test: 0.8419
F1_weighted - Train: 1.0000, Test: 0.8254
Roc_auc - Train: nan, Test: nan

Random Forest Validation Metrics:
Validation Accuracy: 0.8696
F1 Score (Weighted): 0.8598
ROC AUC Score: 0.9851
```

Results:

- For the final step, we loaded the MFCC coefficients for each song, computed basic statistics i.e., mean, standard deviation, 75th percentile, max, min, kurtosis, skew & spectral bandwidth, and root mean square energy using librosa used them as features same as our unlabeled data.
- Iterated through folders where each folder represents a different singer. For each audio .csv file, extracted features with extract_features_from_file and predicted the cluster for each song using the pre-trained RandomForest model.
- Stored cluster counts for each singer in a dictionary to understand which clusters are most common for each singer, highlighting trends in their vocal characteristics.
- The results of this are shown here ➡

```
Singer: Asha Bhosle
 - Cluster 5: 5 songs
 - Cluster 1: 4 songs
 - Cluster 0: 1 songs

Singer: Bhaav Geet
 - Cluster 1: 1 songs
 - Cluster 3: 6 songs
 - Cluster 5: 2 songs
 - Cluster 0: 1 songs

Singer: Jana Gana Mana
 - Cluster 0: 3 songs
 - Cluster 3: 4 songs
 - Cluster 4: 2 songs
 - Cluster 1: 1 songs

Singer: Kishore kumar
 - Cluster 0: 7 songs
 - Cluster 1: 1 songs
 - Cluster 5: 1 songs
 - Cluster 3: 1 songs

Singer: Lavni
 - Cluster 3: 3 songs
 - Cluster 1: 3 songs
 - Cluster 5: 4 songs

Singer: Micheal Jackson
 - Cluster 3: 10 songs
```

Conclusion:

- We can see how good the ‘Michael Jackson’s results are. All the downloaded testing dataset for Michael Jackson belong to cluster 3, which suggests that cluster 3 contains songs of Michael Jackson.
- But problem arose when we noticed there were overlapping cases. Kishore Kumar’s dataset had 7 songs which would belong to cluster 0, at the same time, ‘National Anthem’ dataset also had 3 files which would belong to cluster 0.
- This suggests unsupervised learning model is not robust enough. Due to overlapping cases, we couldn’t jump to the conclusion that if highest number of songs for a particular genre or artist lies in a particular cluster, then that cluster belongs to that genre/artist.
- Hence, we felt the need of a different approach, that’s where we decided to train a supervised learning model.

How will Supervised approach work :

- We created a labeled dataset of 608 songs and converted them to MFCC coefficients.
- We then trained 2 classification models on this dataset and used different parameters for each of the models.
- By calculating different metrics for different parameters corresponding to each model we selected parameters that give us the best metrics.
- Then we used the trained models with the best metrics to predict the category of the given unlabeled dataset of 115 songs.
- From the results of the 2 models, we can use the model that is more confident as it will classify the song into the predicted category that has much more probability than others.

Feature Extraction:

- We first extracted features from the labeled MFCC dataset to create meaningful data for effective use to train and predict
  Features used :
  - Mean, Standard deviation, 75 percentile, max, min, kurtosis, skew, RMS (Described in page 15)
  - **25 percentile**- The value below which 25% of the data points fall. It helps to understand the lower spread of the data.
  - **Median**- The middle value of the dataset when sorted in order
  - **Entropy**- A measure of uncertainty or randomness in the data

  These features provide different aspects of the songs useful for training.

Neural network model

- We used a neural network model within total 4 Dense layers with reLU activation and 1 dense layer with softmax activation for classification.
- Used dropout (0.3) after each layer for regularisation.
- The figure shows the model summary.

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 220)	0
dense (Dense)	(None, 128)	28,288
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16,512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 32)	2,080
dense_4 (Dense)	(None, 6)	198

Training and Optimizer selection (NN)

- We also converted the labels for use by one-hot coding.
- After loading the training data set we trained the model on them using learning rate 10^-5, 200 epochs, and batch_size 32 with 3 different optimizers **Adam, RMSprop & Lion**.
- Using these optimizers we got the best training and validation accuracy with RMSprop.

```
Epoch 190/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8540 - loss: 0.4739 - val_accuracy: 0.8279 - val_loss: 0.5601
Epoch 191/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8563 - loss: 0.3807 - val_accuracy: 0.7951 - val_loss: 0.6833
Epoch 192/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8458 - loss: 0.4613 - val_accuracy: 0.8279 - val_loss: 0.5883
Epoch 193/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8695 - loss: 0.3892 - val_accuracy: 0.7951 - val_loss: 0.6850
Epoch 194/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8588 - loss: 0.3747 - val_accuracy: 0.8689 - val_loss: 0.6773
Epoch 195/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8326 - loss: 0.5125 - val_accuracy: 0.8033 - val_loss: 0.6241
Epoch 196/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8763 - loss: 0.3591 - val_accuracy: 0.8115 - val_loss: 0.7074
Epoch 197/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8340 - loss: 0.4387 - val_accuracy: 0.8279 - val_loss: 0.6422
Epoch 198/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8735 - loss: 0.3594 - val_accuracy: 0.8197 - val_loss: 0.8354
Epoch 199/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8290 - loss: 0.4919 - val_accuracy: 0.8279 - val_loss: 0.7118
Epoch 200/200
16/16 ━━━━━━━━━━━━━━━━━━━━ accuracy: 0.8777 - loss: 0.4188 - val_accuracy: 0.8607 - val_loss: 0.6124
```

SVM model

- We developed an SVM (Support Vector Machine) model for audio classification based on MFCC features. The model aims to classify audio files into different categories using the MFCC coefficients and statistical features extracted from the audio data.
- Svm model finds feature at hyperplane to find the best plane that can cluster and classify the model using best hyper plane.
- Different parameters, such as the kernel type (e.g., linear, RBF), C (regularization parameter), and gamma (kernel coefficient), were tuned to optimize the SVM model’s performance.

Traning and Parameter selection (SVM)

- We have used C_values = [0.1,1,10], gamma_values = [’scale’, 0.01, 0.1] and kernels = [’linear’, ‘rbf’, ‘poly’].
- We have used different martices such as C, gamma, kernel, f1 score , accuracy ,precision, recall to compare model from which we found poly, gamma = 0.1 and c= 10 gave the best results.

Model performance for different hyperparameters:

	C	gamma	kernel	accuracy	precision	recall	f1_score
26	10.0	0.1	poly	0.778689	0.787709	0.778689	0.779853
23	10.0	0.01	poly	0.778689	0.787709	0.778689	0.779853
5	0.1	0.01	poly	0.778689	0.787709	0.778689	0.779853
17	1.0	0.1	poly	0.778689	0.787709	0.778689	0.779853
8	0.1	0.1	poly	0.778689	0.787709	0.778689	0.779853
14	1.0	0.01	poly	0.778689	0.787709	0.778689	0.779853
12	1.0	0.01	linear	0.770492	0.790553	0.770492	0.774505
24	10.0	0.1	linear	0.770492	0.790553	0.770492	0.774505
21	10.0	0.01	linear	0.770492	0.790553	0.770492	0.774505
18	10.0	scale	linear	0.770492	0.790553	0.770492	0.774505
15	1.0	0.1	linear	0.770492	0.790553	0.770492	0.774505
0	0.1	scale	linear	0.770492	0.790553	0.770492	0.774505
9	1.0	scale	linear	0.770492	0.790553	0.770492	0.774505
6	0.1	0.1	linear	0.770492	0.790553	0.770492	0.774505
3	0.1	0.01	linear	0.770492	0.790553	0.770492	0.774505
20	10.0	scale	poly	0.745902	0.744530	0.745902	0.743551
19	10.0	scale	rbf	0.729508	0.729961	0.729508	0.728378
11	1.0	scale	poly	0.581967	0.614305	0.581967	0.570020
10	1.0	scale	rbf	0.565574	0.615980	0.565574	0.527636
2	0.1	scale	poly	0.377049	0.196014	0.377049	0.255697
1	0.1	scale	rbf	0.270492	0.126043	0.270492	0.140128
16	1.0	0.1	rbf	0.254098	0.262363	0.254098	0.122888
22	10.0	0.01	rbf	0.254098	0.262363	0.254098	0.122888
25	10.0	0.1	rbf	0.254098	0.262363	0.254098	0.122888
13	1.0	0.01	rbf	0.254098	0.262363	0.254098	0.122888
7	0.1	0.1	rbf	0.237705	0.056504	0.237705	0.091304
4	0.1	0.01	rbf	0.237705	0.056504	0.237705	0.091304

Testing on given MFCC dataset

- Using the trained model we predicted the given dataset to find which file belongs to which section.
- We also listed out the probabilities for each song to be in each section.

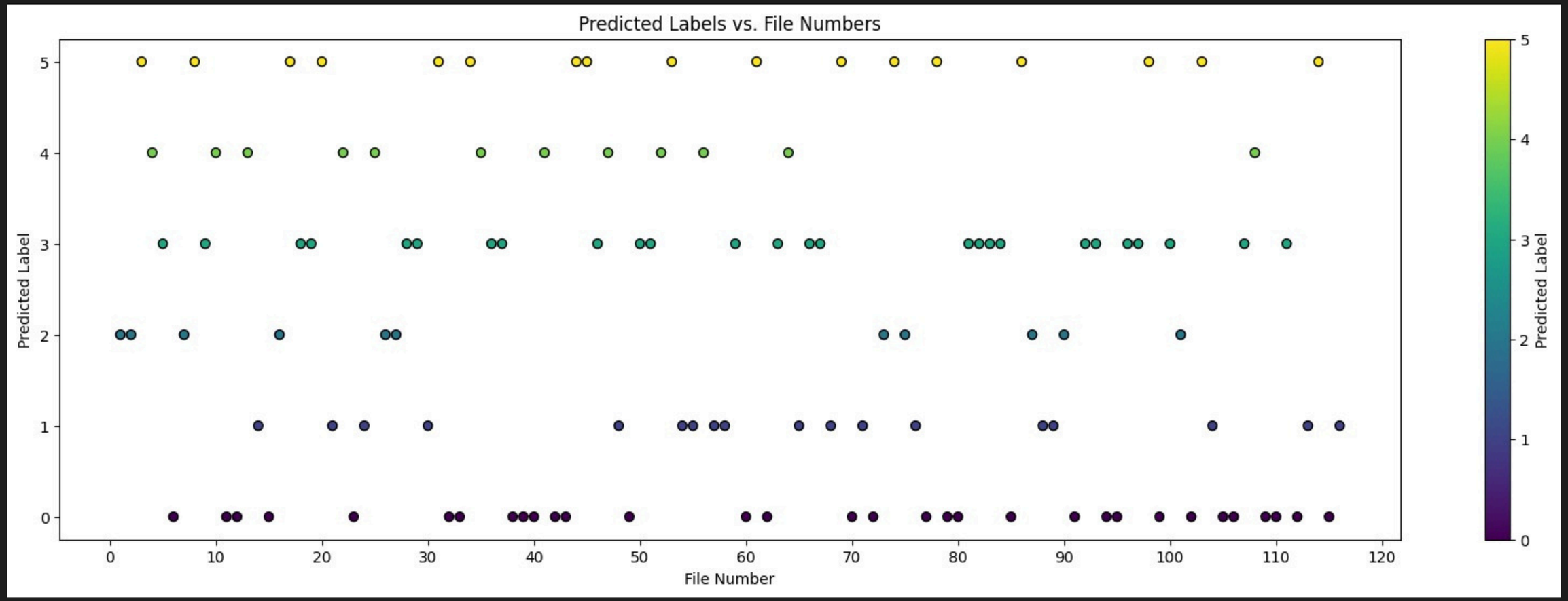
```
01-MFCC.csv:
  Predicted Label: 2
  Probabilities: [0.04232587 0.2987292  0.33293816 0.12826408 0.05912467 0.13861807]

02-MFCC.csv:
  Predicted Label: 2
  Probabilities: [9.8277542e-06 4.0834281e-03 9.9301964e-01 7.8510796e-04 7.8556710e-05
 2.0234208e-03]

03-MFCC.csv:
  Predicted Label: 5
  Probabilities: [4.7541109e-15 1.6344549e-09 3.2071759e-10 5.3493824e-11 9.4714278e-11
 1.0000000e+00]

04-MFCC.csv:
  Predicted Label: 4
  Probabilities: [0.25842032 0.00210767 0.05003594 0.0034637  0.6759412  0.01003112]

05-MFCC.csv:
  Predicted Label: 3
  Probabilities: [1.6905126e-03 1.6767140e-03 1.1582850e-03 9.9546742e-01 1.9344550e-06
 5.0620051e-06]
```

Neural network

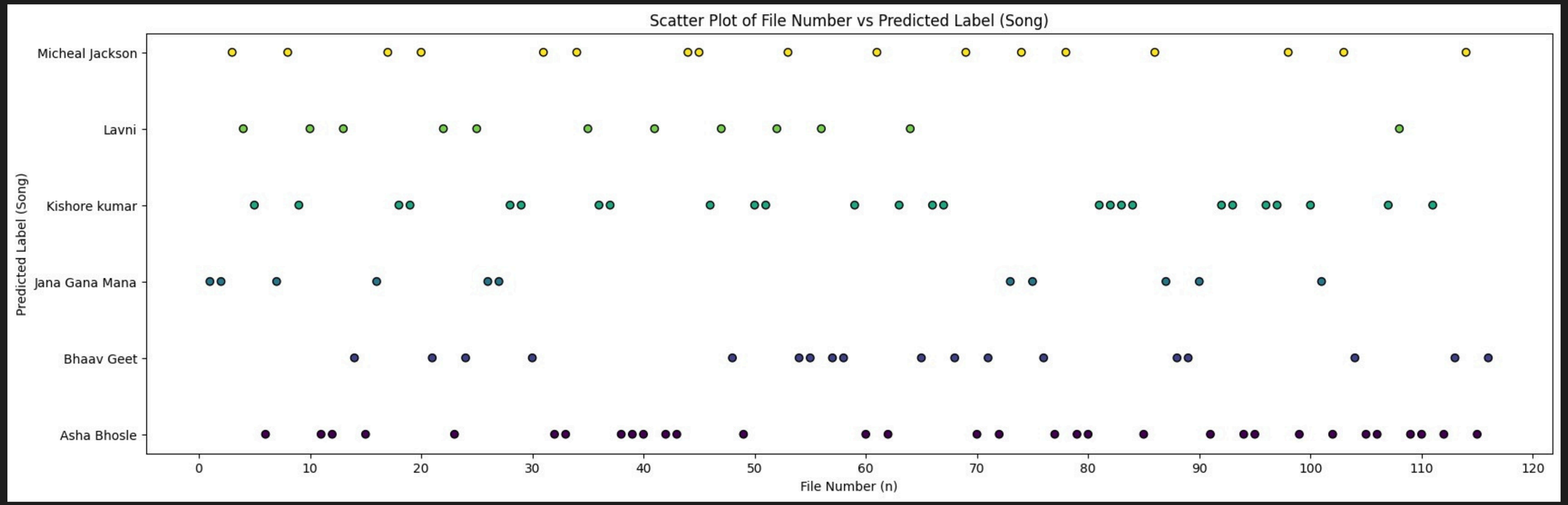
Filename	Predicted_Label	Probabilities
01-MFCC.csv	2	[0.0864656611685492, 0.07174937895345629, 0.38...
02-MFCC.csv	2	[0.007948640031712665, 0.0045777308750499296, ...
03-MFCC.csv	5	[0.017461065738578623, 0.015223648042987821, 0...
04-MFCC.csv	4	[0.34840110769320853, 0.0285754603903909, 0.04...
05-MFCC.csv	3	[0.02492774867350828, 0.04616461032273601, 0.0...
...	...	...
95-MFCC.csv	4	[0.24663298372696882, 0.018504593519777544, 0....
96-MFCC.csv	3	[0.36659209721214714, 0.08032143729723046, 0.0...
97-MFCC.csv	3	[0.05492260084862037, 0.03944473198684227, 0.0...
98-MFCC.csv	1	[0.01981518167742911, 0.6055012350973249, 0.12...

SVM

Testing results (NN)



Testing results (NN)



Final categories (NN)

```
'Bhaav Geet': [('104-MFCC.csv', 0.9938524),
 ('113-MFCC.csv', 0.74066776),
 ('116-MFCC.csv', 0.99417084),
 ('14-MFCC.csv', 0.533282),
 ('21-MFCC.csv', 0.9950492),
 ('24-MFCC.csv', 0.9309341),
 ('30-MFCC.csv', 0.7566643),
 ('48-MFCC.csv', 0.99693465),
 ('54-MFCC.csv', 0.9868454),
 ('55-MFCC.csv', 0.99993014),
 ('57-MFCC.csv', 0.78921556),
 ('58-MFCC.csv', 0.99462306),
 ('65-MFCC.csv', 0.9950669),
 ('68-MFCC.csv', 0.83925843),
 ('71-MFCC.csv', 0.45424068),
 ('76-MFCC.csv', 0.44230887),
 ('88-MFCC.csv', 0.99972194),
 ('89-MFCC.csv', 0.25358343)],
```

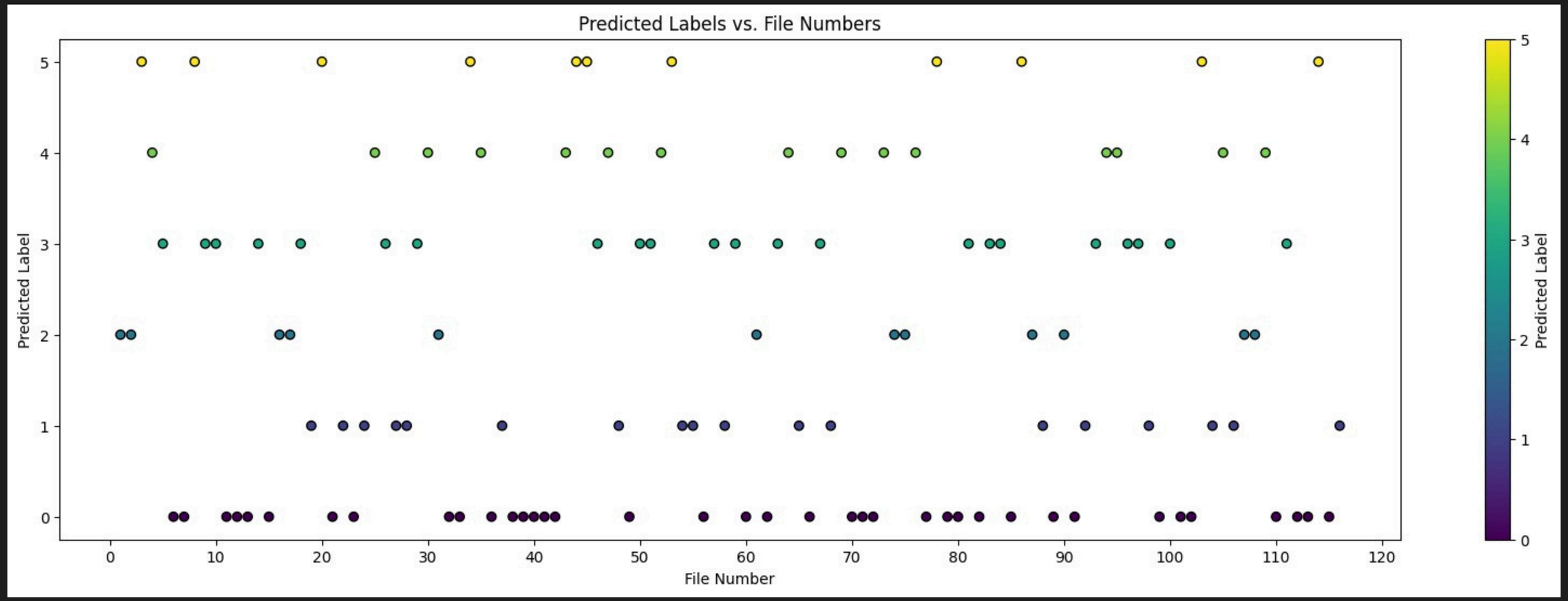
```
'Kishore kumar': [('05-MFCC.csv', 0.9954674),
 ('107-MFCC.csv', 0.883381),
 ('111-MFCC.csv', 0.9740699),
 ('18-MFCC.csv', 0.99455655),
 ('19-MFCC.csv', 0.44161752),
 ('28-MFCC.csv', 0.39222023),
 ('29-MFCC.csv', 0.99872845),
 ('36-MFCC.csv', 0.4953577),
 ('37-MFCC.csv', 0.88346463),
 ('46-MFCC.csv', 0.9998005),
 ('50-MFCC.csv', 0.99944407),
 ('51-MFCC.csv', 0.99970204),
 ('59-MFCC.csv', 0.99994004),
 ('63-MFCC.csv', 0.99865305),
 ('66-MFCC.csv', 0.9907227),
 ('67-MFCC.csv', 0.9649313),
 ('81-MFCC.csv', 0.99791616),
 ('82-MFCC.csv', 0.99425405),
 ('83-MFCC.csv', 0.9908213),
 ('84-MFCC.csv', 0.99982697),
 ('92-MFCC.csv', 0.8590325),
 ('93-MFCC.csv', 0.99969757),
 ('96-MFCC.csv', 0.95713985),
 ('97-MFCC.csv', 0.96777225)],
'Lavni': [('04-MFCC.csv', 0.6759412),
 ('10-MFCC.csv', 0.9739629),
 ('108-MFCC.csv', 0.25273576),
 ('13-MFCC.csv', 0.5206923),
 ('22-MFCC.csv', 0.2249803),
 ('25-MFCC.csv', 0.48603436),
 ('35-MFCC.csv', 0.7887029),
 ('41-MFCC.csv', 0.5253142),
 ('47-MFCC.csv', 0.994833),
```

```
'Bhaav Geet': [('104-MFCC.csv', 0.9938524),
 ('113-MFCC.csv', 0.74066776),
 ('116-MFCC.csv', 0.99417084),
 ('14-MFCC.csv', 0.533282),
 ('21-MFCC.csv', 0.9950492),
 ('24-MFCC.csv', 0.9309341),
 ('30-MFCC.csv', 0.7566643),
 ('48-MFCC.csv', 0.99693465),
 ('54-MFCC.csv', 0.9868454),
 ('55-MFCC.csv', 0.99993014),
 ('57-MFCC.csv', 0.78921556),
 ('58-MFCC.csv', 0.99462306),
 ('65-MFCC.csv', 0.9950669),
 ('68-MFCC.csv', 0.83925843),
 ('71-MFCC.csv', 0.45424068),
 ('76-MFCC.csv', 0.44230887),
 ('88-MFCC.csv', 0.99972194),
 ('89-MFCC.csv', 0.25358343)],
'Jana Gana Mana': [('01-MFCC.csv', 0.33293816)
 ('02-MFCC.csv', 0.99301964),
 ('07-MFCC.csv', 0.3081052),
 ('101-MFCC.csv', 0.22455193),
 ('16-MFCC.csv', 0.98549277),
 ('26-MFCC.csv', 0.3729531),
 ('27-MFCC.csv', 0.3370258),
 ('73-MFCC.csv', 0.36721513),
 ('75-MFCC.csv', 0.9240806),
 ('87-MFCC.csv', 0.952534),
 ('90-MFCC.csv', 0.8682872)],
'Kishore kumar': [('05-MFCC.csv', 0.9954674),
 ('09-MFCC.csv', 0.99981624),
 ('100-MFCC.csv', 0.9940951),
```

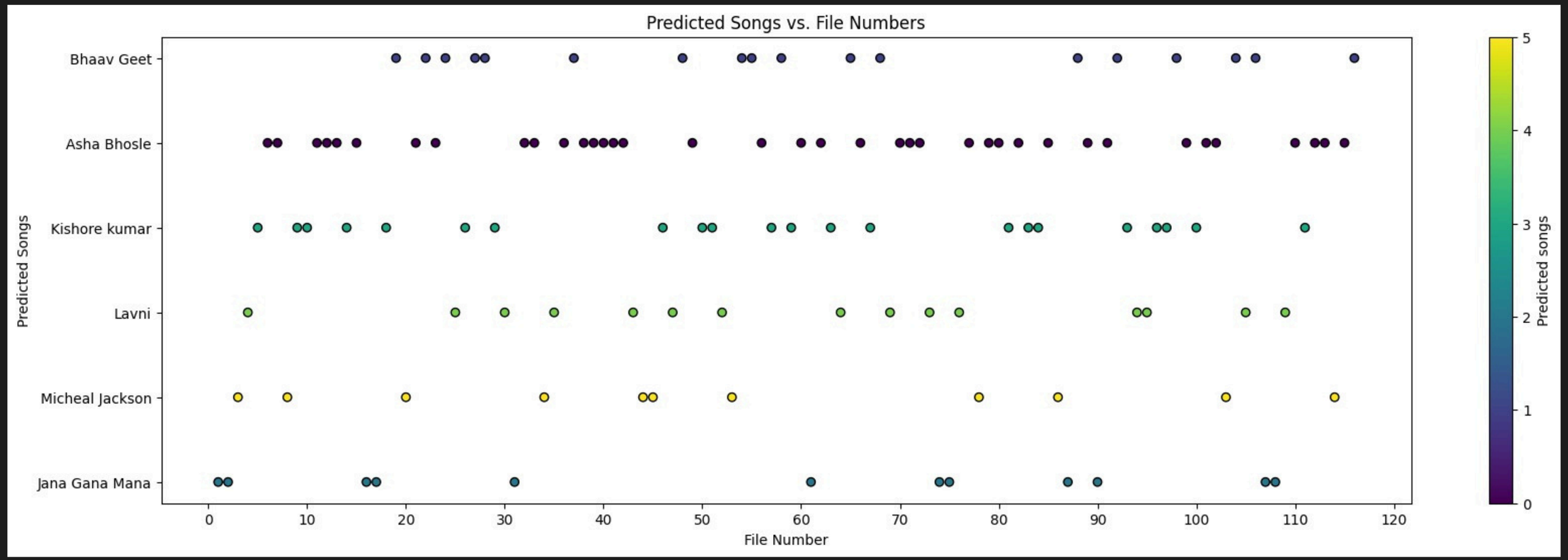
```
'Lavni': [('04-MFCC.csv', 0.6759412),
 ('52-MFCC.csv', 0.51412123),
 ('56-MFCC.csv', 0.48989853),
 ('64-MFCC.csv', 0.6220926)],
'Micheal Jackson': [('03-MFCC.csv', 1.0)
 ('08-MFCC.csv', 1.0),
 ('103-MFCC.csv', 1.0),
 ('114-MFCC.csv', 1.0),
 ('17-MFCC.csv', 0.44168958),
 ('20-MFCC.csv', 1.0),
 ('31-MFCC.csv', 0.9379874),
 ('34-MFCC.csv', 0.9846273),
 ('44-MFCC.csv', 1.0),
 ('45-MFCC.csv', 1.0),
 ('53-MFCC.csv', 0.99983704),
 ('61-MFCC.csv', 0.95328534),
 ('69-MFCC.csv', 0.87994945),
 ('74-MFCC.csv', 0.6160137),
 ('78-MFCC.csv', 0.9628731),
 ('86-MFCC.csv', 1.0),
 ('98-MFCC.csv', 0.9982601)]}
```

```
{'Asha Bhosle': [('06-MFCC.csv', 0.6495029),
 ('102-MFCC.csv', 0.9234652),
 ('105-MFCC.csv', 0.54315436),
 ('106-MFCC.csv', 0.8672886),
 ('109-MFCC.csv', 0.4795615),
 ('11-MFCC.csv', 0.75003785),
 ('110-MFCC.csv', 0.8079299),
 ('112-MFCC.csv', 0.6933528),
 ('115-MFCC.csv', 0.88220596),
 ('12-MFCC.csv', 0.31788373),
 ('15-MFCC.csv', 0.57680875),
 ('23-MFCC.csv', 0.7302246),
 ('32-MFCC.csv', 0.5738796),
 ('33-MFCC.csv', 0.90364736),
 ('38-MFCC.csv', 0.68848044),
 ('39-MFCC.csv', 0.7909123),
 ('40-MFCC.csv', 0.9618707),
 ('42-MFCC.csv', 0.84460205),
 ('43-MFCC.csv', 0.58718157),
 ('49-MFCC.csv', 0.76194596),
 ('60-MFCC.csv', 0.8697038),
 ('62-MFCC.csv', 0.86407584),
 ('70-MFCC.csv', 0.87037176),
 ('72-MFCC.csv', 0.5441827),
 ('77-MFCC.csv', 0.7307024),
 ('79-MFCC.csv', 0.9742485),
 ('80-MFCC.csv', 0.7407227),
 ('85-MFCC.csv', 0.73490804),
 ('91-MFCC.csv', 0.7704363),
 ('94-MFCC.csv', 0.77031815),
 ('95-MFCC.csv', 0.64564925),
 ('99-MFCC.csv', 0.57470024)],
```

Testing results (SVM)



Testing results (SVM)



Files certainly in predicted class (NN)

Listed here are a few of the files that are almost certain to be in the specific section.
These are files that have high probability of being in the predicted song.

- Top 5 MFCC files with the highest probability for the label 'Asha Bhosle':

79-MFCC.csv: Probability = 0.974
40-MFCC.csv: Probability = 0.961
02-MFCC.csv: Probability = 0.923
33-MFCC.csv: Probability = 0.903
115-MFCC.csv: Probability = 0.882

- Top 5 MFCC files with the highest probability for the label 'Bhaav Geet':

55-MFCC.csv: Probability = 0.999
88-MFCC.csv: Probability = 0.999
48-MFCC.csv: Probability = 0.996
65-MFCC.csv: Probability = 0.995
21-MFCC.csv: Probability = 0.995

Files certainly in predicted class (NN)

- Top 5 MFCC files with the highest probability for the label 'Jana Gana Mana':
  02-MFCC.csv: Probability = 0.993
  16-MFCC.csv: Probability = 0.985
  87-MFCC.csv: Probability = 0.952
  75-MFCC.csv: Probability = 0.924
  90-MFCC.csv: Probability = 0.868
- Top 5 MFCC files with the highest probability for the label 'Kishore Kumar':
  59-MFCC.csv: Probability = 0.999
  84-MFCC.csv: Probability = 0.999
  09-MFCC.csv: Probability = 0.999
  46-MFCC.csv: Probability = 0.999
  51-MFCC.csv: Probability = 0.999
- Top 5 MFCC files with the highest probability for the label 'Lavni':
  47-MFCC.csv: Probability = 0.994
  10-MFCC.csv: Probability = 0.973
  35-MFCC.csv: Probability = 0.788
  04-MFCC.csv: Probability = 0.675
  64-MFCC.csv: Probability = 0.622

Files certainly in predicted class (NN)

- Top 5 MFCC files with the highest probability for label 'Micheal Jackson':
  03-MFCC.csv: Probability = 1.0
  08-MFCC.csv: Probability = 1.0
  103-MFCC.csv: Probability = 1.0
  114-MFCC.csv: Probability = 1.0
  20-MFCC.csv: Probability = 1.0

Results and conclusions (NN)

- **High Classification Accuracy:** The NN model achieved high accuracy in categorizing audio files based on MFCC features, consistently performing well across different genres and artists.
- **Confidence in Predictions:** For certain classes, such as Michael Jackson and Kishore Kumar, the model showed near-perfect confidence levels, indicating a strong ability to identify unique characteristics of specific artists.
- **Reliable Genre Differentiation:** The model effectively distinguished genres like Lavni, National Anthems, and Bhaav Geet, demonstrating a precise understanding of genre-specific MFCC patterns.

Files certainly in predicted class (SVM)

- Top 5 MFCC files with the highest probability for the label 'Jana Gana Mana':
  75-MFCC.csv: Probability = 0.998
  16-MFCC.csv: Probability = 0.996
  02-MFCC.csv: Probability = 0.961
  90-MFCC.csv: Probability = 0.956
  31-MFCC.csv: Probability = 0.873
- Top 5 MFCC files with the highest probability for the label 'Micheal Jackson':
  20-MFCC.csv: Probability = 0.961
  114-MFCC.csv: Probability = 0.94
  45-MFCC.csv: Probability = 0.884
  103-MFCC.csv: Probability = 0.865
  03-MFCC.csv: Probability = 0.849
- Top 5 MFCC files with the highest probability for the label 'Lavni':
  52-MFCC.csv: Probability = 0.857
  73-MFCC.csv: Probability = 0.698
  25-MFCC.csv: Probability = 0.673
  105-MFCC.csv: Probability = 0.650
  95-MFCC.csv: Probability = 0.643

Files certainly in predicted class (SVM)

- Top 5 MFCC files with the highest probability for the label 'Kishore Kumar':
  09-MFCC.csv: Probability = 0.984
  100-MFCC.csv: Probability = 0.966
  84-MFCC.csv: Probability = 0.96
  29-MFCC.csv: Probability = 0.940
  51-MFCC.csv: Probability = 0.937
- Top 5 MFCC files with the highest probability for the label 'Asha Bhosle':
  79-MFCC.csv: Probability = 0.880
  110-MFCC.csv: Probability = 0.879
  80-MFCC.csv: Probability = 0.869
  39-MFCC.csv: Probability = 0.861
  32-MFCC.csv: Probability = 0.838
- Top 5 MFCC files with the highest probability for the label 'Bhaav Geet':
  116-MFCC.csv: Probability = 0.957
  55-MFCC.csv: Probability = 0.934
  65-MFCC.csv: Probability = 0.915
  54-MFCC.csv: Probability = 0.85
  104-MFCC.csv: Probability = 0.812

Option Question 1

- The question was to classify the song sung by male, females and male and females
- The model was trained using solo songs of Michael Jackson, Kishore Kumar, and Asha Bhosle.
- So female songs are the top songs of Asha Bhosle and male songs are top songs of Kishore Kumar and Michael Jackson
- the songs sung by male and female artists can be found by low probably of Asha Bhosle and Kishore Kumar
- also songs with similar probability for Asha Bhosle and Kishore Kumar from Bhaav Geet and lavni .

Option Question 1

Files containing songs sung by only male	Files containing songs sung by only female	Files containing songs sung by both Male and Female
59-MFCC.csv	79-MFCC.csv	76-MFCC.csv
84-MFCC.csv	40-MFCC.csv	105-MFCC.csv
09-MFCC.csv	102-MFCC.csv	109-MFCC.csv
46-MFCC.csv	33-MFCC.csv	12-MFCC.csv
51-MFCC.csv	115-MFCC.csv	89-MFCC.csv
		108-MFCC.csv

Final Conclusion

- **Model Recommendation:** After evaluating the performance, we recommend the Neural Network (NN) model as the final choice for audio classification.
- **Reason for Selection:** The NN model showed high confidence in predictions, which was crucial in achieving reliable classification. This metric was consistently stronger compared to other models.
- **Performance Reliability:** The NN model's ability to capture complex patterns in MFCC features allowed it to accurately differentiate genres and artists.
- **Conclusion:** Given its accuracy and robustness, the Neural Network model is the optimal solution for classifying audio files in this project.

Key Learnings:

- MFCC features played a crucial role in capturing the distinct traits of audio, showcasing how important it is to choose the right features for audio classification tasks.
- While initial unsupervised clustering gave us some useful insights, we found that supervised models were key for achieving accurate classification, emphasizing the importance of both methods in evaluating models.
- By testing various models like Neural Networks, SVM, and Random Forest, we discovered that the Neural Network was the best option because it can manage complex, non-linear relationships in audio data.
- Working with audio files reinforced the need for models that can decode intricate patterns, proving that deep learning models like Neural Networks work best in these scenarios.
- The overlap between genres and similarities among artists revealed the challenges in audio classification, highlighting the necessity for strong models and thorough training data.
- Overall, this project deepened our understanding of audio feature extraction, model selection, and the complexities involved in creating an effective audio classification system.

Evaluation Criteria

- We solved all three mandatory problems and an optional problem.
- The first approach we used contained un-supervised learning and then classification and testing to define which cluster belongs to which category This approach was about of the box concept we used in this project.
- Different features were used for better genre classification. Some of the important features are:
  - **Entropy:** Shows Randomness in the song which helps in better classification
  - **RMS:** This shows the energy or loudness of a particular frequency hence helping in differentiating the genres, Eg: Lavni will have higher RMS as compared to Bhaavgeet hence a parameter to differentiate.
  - **Spectral Bandwidth**- An indication of how the energy is distributed across frequencies , so if spectral bandwidth of a frequency is high then it is more probably a female singer rather than male.
- The rest of the features are given in 15th and 28th slide.

Learning and Hurdles

- **Low accuracy:** Initially we were getting low accuracy for models hence we optimised it by iterating through different optimizers and learning rates
- **Limited Features:** Accuracy was still coming low because of the limited number of features that we were using in the dataset (mean and variance), hence we started to explore other features and added them so that our model will distinguish better between the genres.
- **Data Shuffling:** Initially when we were training the NN with the help of 600+ downloaded songs, since the songs were in an organised fashion, we were getting a lot of fluctuations in our loss function, hence we randomised the data which gave a more consistent training,
- **Low Silhouette Score:** In unsupervised learning, we were constantly facing with low silhouette score and low accuracy, hence we tested different models to find the best one.
- **Feature Engineering:** Finding the best features that would aid in genre classification and then testing them if they aided in better clustering.
- **Validation optimisation:** We needed better validation scores hence to optimise it we ran GridSearchCV to fine tune model parameters which helped us achieve better validation performances.

THANK YOU!