Group 16

THE

Music Genre Classification

by Machine Learning Models

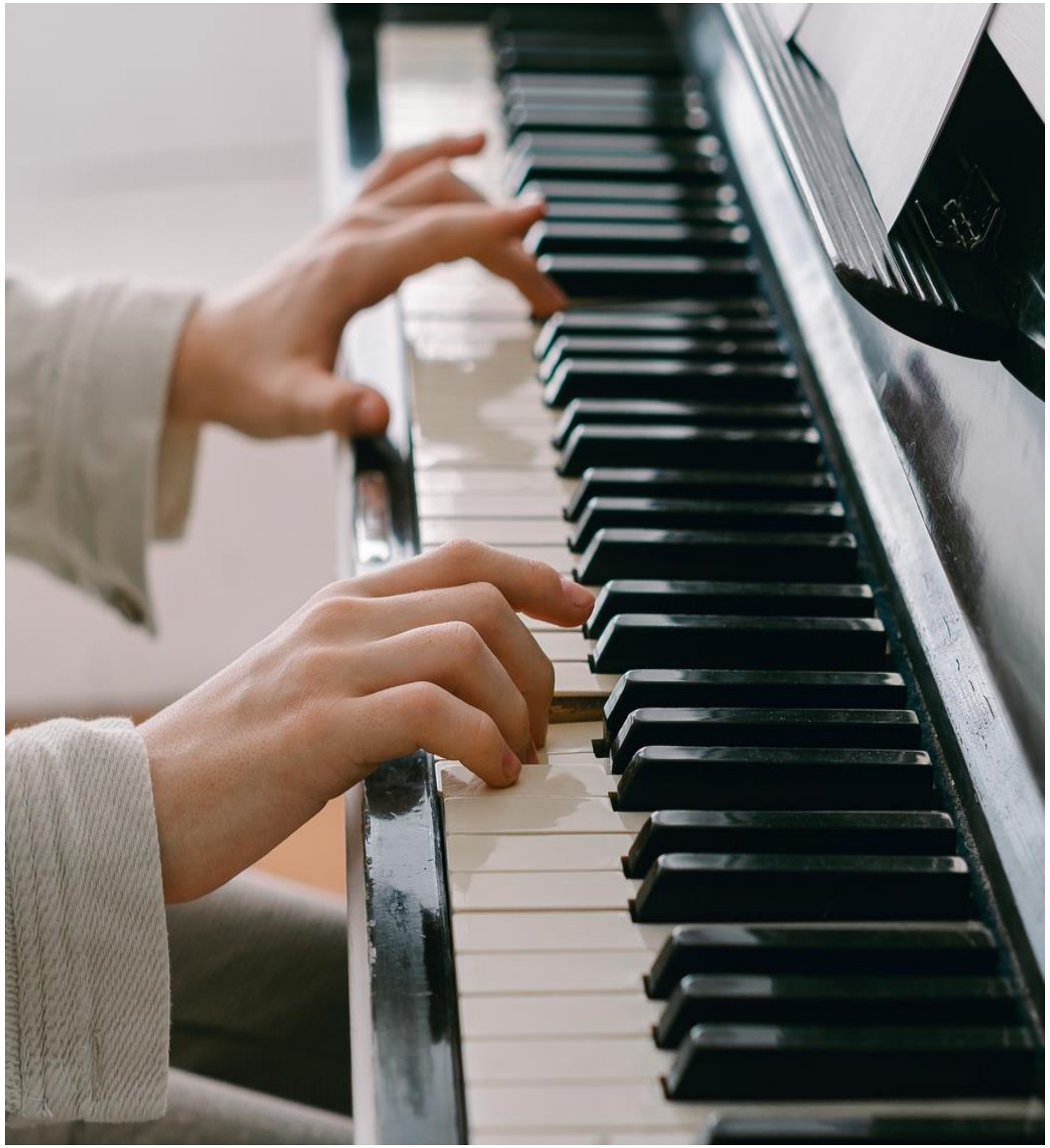
Group members:

Rui Gao, Chenfeng Wu, Haorui Wu, Yi Xu, Lin Zhang

Motivation

As I am currently learning the piano, I am interested in applications of machine learning in the domain of music, which can include tasks such as music modeling, music genre classification, and artificial music composition.

So, I want to use some machine learning models such as logistic regression, decision tree, bagging and boosting to do classification of music genre.



Dataset

We choose the **GTZAN** dataset ,which is the ”most-used public dataset for evaluation in machine listening research for music genre recognition.

Most notably, it includes a dataset of 30-second audio clips from 100 audio files for 10 genres each. And each sample has **57 features**.

We will do some same modification of the original dataset. We split each audio file **into 3-second** clips and choose only the classical and pop genres. As a result, we will have 2*100*10 = 2000 samples.

Main Variables

Rms : the Root Mean Square loudness of an audio segment.

Chroma : capture the distribution of energy across the 12 pitch classes (C, C#, D, ..., B) of the musical scale.

Spectral centroid : indicates where the ”center of mass” for a sound is located.

Rolloff : measure of the frequency below which a specified percentage (usually 85%) of the total

Zero crossing rate: the rate at which the signal waveform crosses zero.

Harmony: the combination of different musical notes simultaneously to create chords and chord progressions. When a piece is ”harmony high,” it emphasizes complex on creating a fuller textured sound.

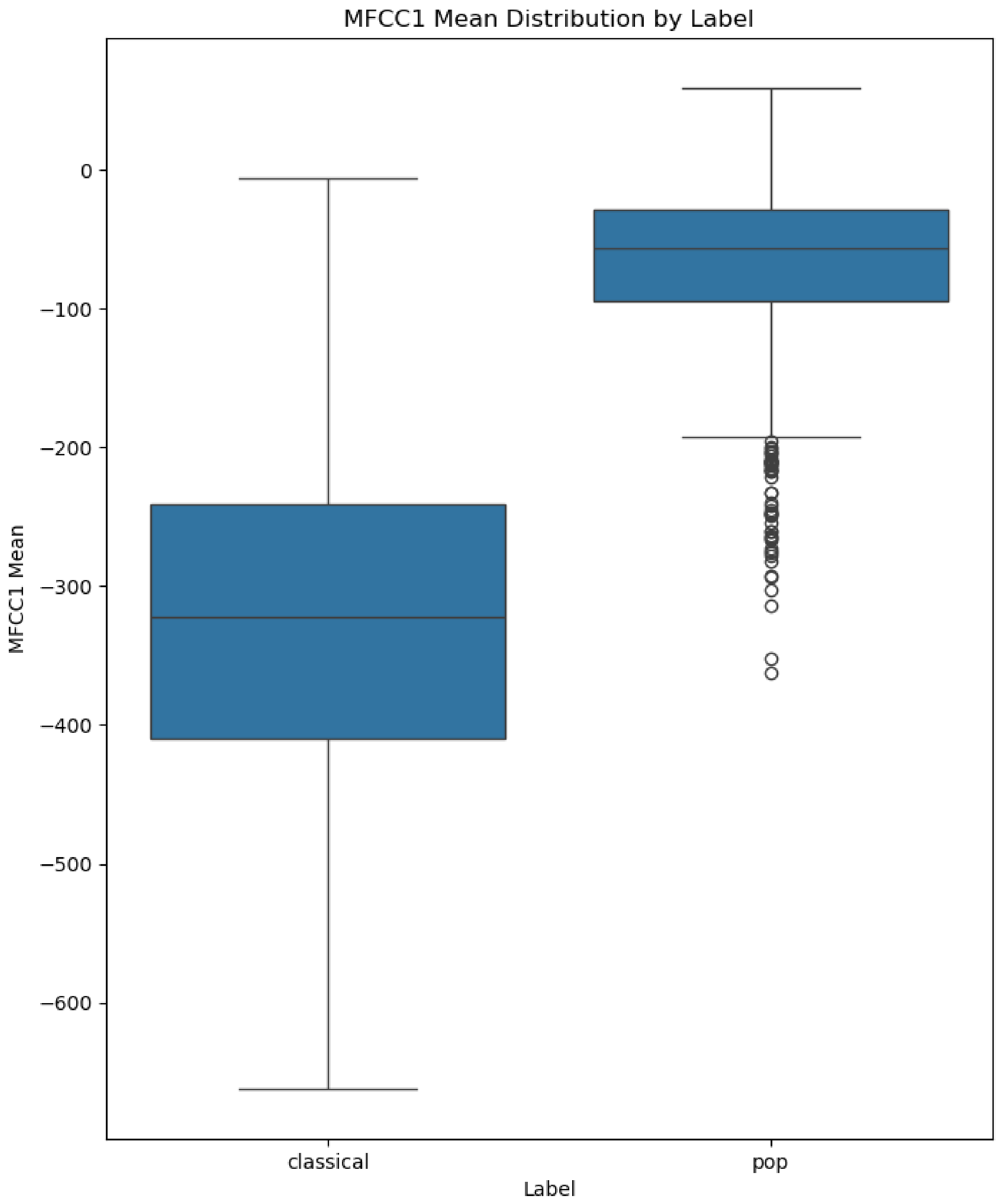
Perceptual : refer to audio features that relate to how humans perceive sound

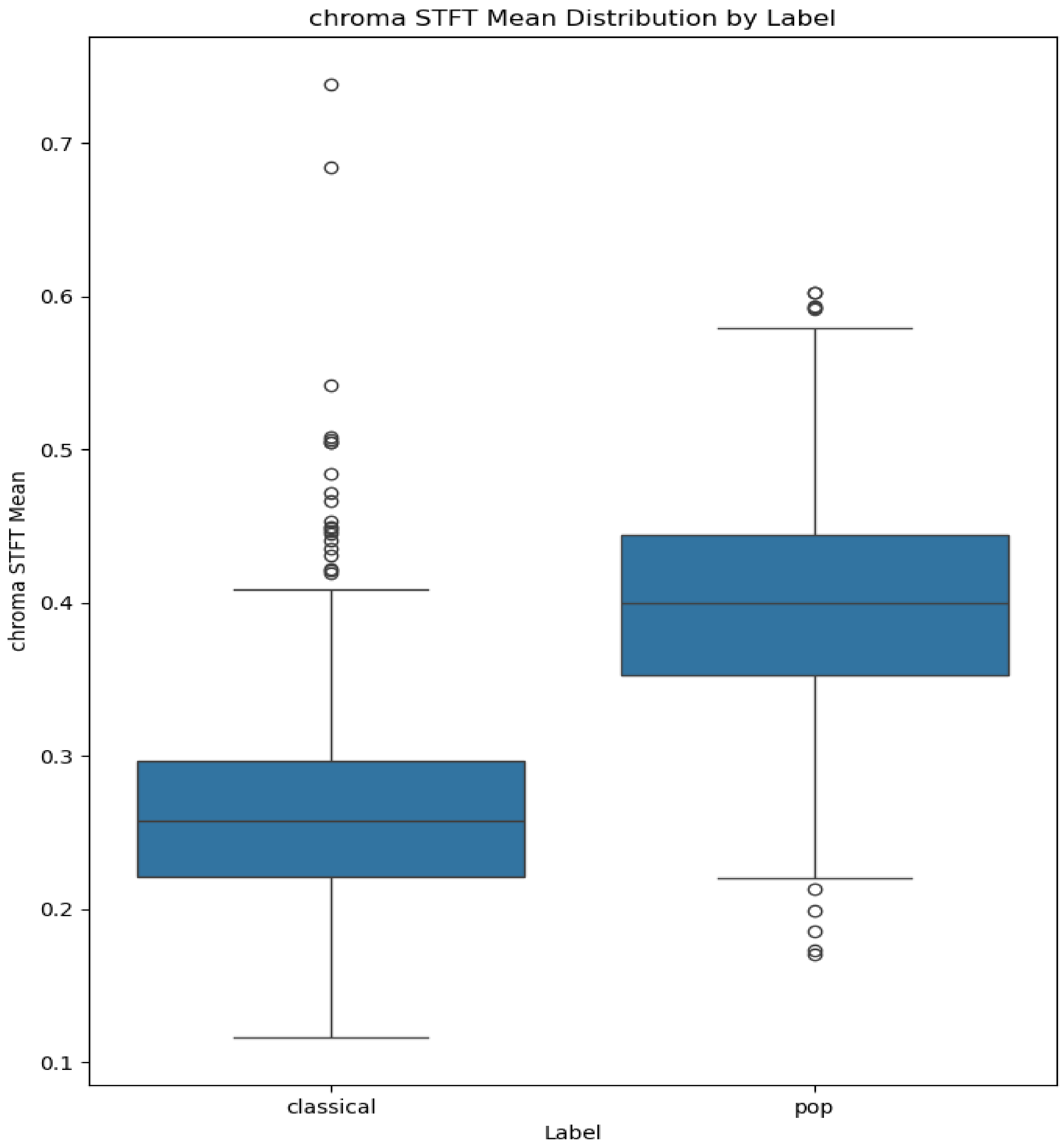
Tempo: beats per minute.

20 mfccs : refers to ”mel-frequency cepstral coefficients”, it is used to capture timbral qualities of sound.

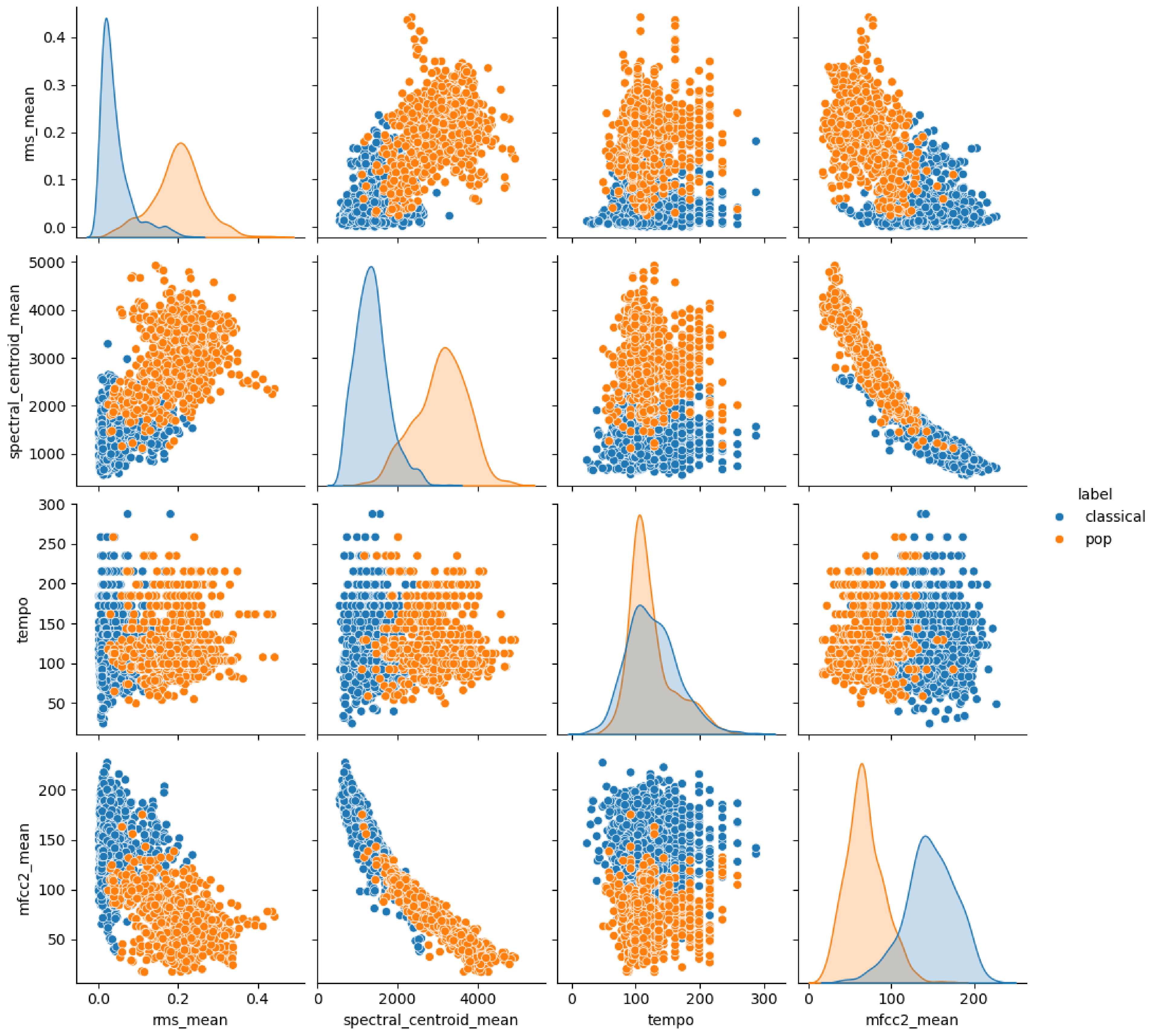
Note: Except for tempo, every other feature has a sample mean and variance.

Boxplots of Some Features

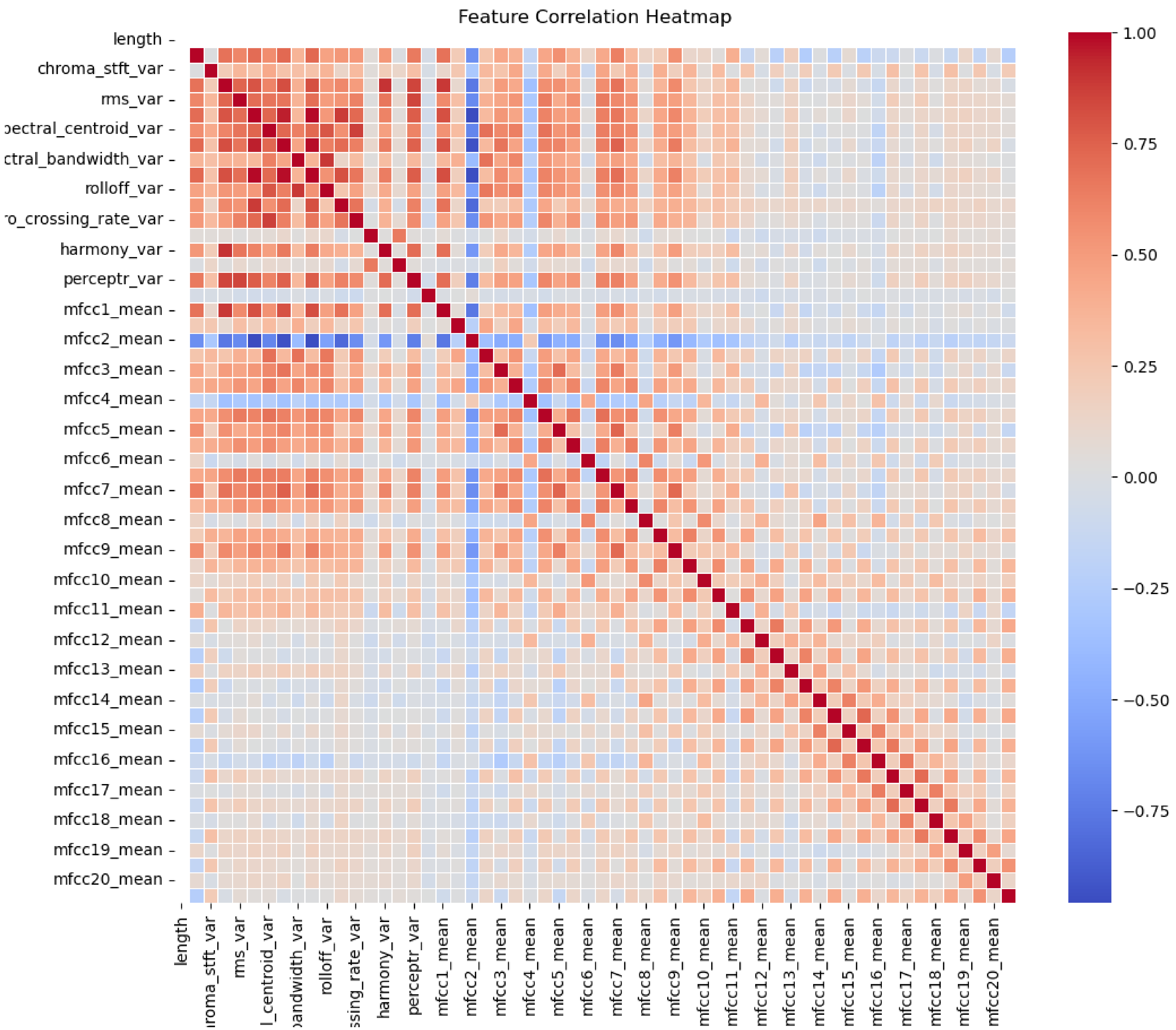




Pairplots


rms_mean
spectral_centroid_mean
tempo
mfcc2_mean
label
classical
pop

Heatmap



Feature Pairs with High Correlation (greater than 0.75):

rolloff_mean	spectral_centroid_mean	0.986815
	spectral_bandwidth_mean	0.977943
spectral_bandwidth_mean	spectral_centroid_mean	0.950121
harmony_var	rms_mean	0.905822
mfcc1_mean	rms_mean	0.891388
rolloff_var	spectral_bandwidth_var	0.888235
zero_crossing_rate_mean	spectral_centroid_mean	0.879322
zero_crossing_rate_var	spectral_centroid_var	0.862412
perceptr_var	rms_mean	0.853679
	rms_var	0.847452
mfcc1_mean	rolloff_mean	0.822883
spectral_bandwidth_mean	rms_mean	0.819893
rolloff_mean	rms_mean	0.815552
mfcc1_mean	spectral_centroid_mean	0.813627
zero_crossing_rate_mean	rolloff_mean	0.808917
mfcc1_mean	spectral_bandwidth_mean	0.806290
spectral_centroid_mean	rms_mean	0.795427
rolloff_var	spectral_centroid_var	0.789379
perceptr_var	spectral_centroid_mean	0.770322
	rolloff_mean	0.768551
mfcc2_mean	rms_mean	-0.757909
	mfcc1_mean	-0.759145
	zero_crossing_rate_mean	-0.824188
	spectral_bandwidth_mean	-0.932321
	rolloff_mean	-0.946767
	spectral_centroid_mean	-0.957169

Logistic Regression

- Implement package directly and evaluate
- Implement batch gradient approach and evaluate
- Implement mini-batch gradient approach and evaluate

Equations and Codes

Loss function and derivatives

- Logistic regression model (single input): $\hat{y}^{(i)} = \sigma(\mathbf{w}^T x^{(i)} + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$
- Logistic regression model (N inputs): $\hat{y} = \sigma(X\mathbf{w} + \mathbf{b})$, $\sigma(\mathbf{z}) = \frac{1}{1+e^{-\mathbf{z}}}$, $\mathbf{b} = \begin{bmatrix} b & \cdots & b \end{bmatrix}^T$
- Loss (single training example): $l(y^{(i)}, \hat{y}^{(i)}) = -\left(y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})\right)$
- Loss (set of N training examples): $L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} l(y^{(i)}, \hat{y}^{(i)})$
- Derivative of $L(y, \hat{y})$ w.r.t $\mathbf{w}$: $\frac{\partial L}{\partial \mathbf{w}} = \frac{1}{N} X^T (\hat{y} - y)$
- Derivative of $L(y, \hat{y})$ w.r.t b: $\frac{\partial L}{\partial b} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})$

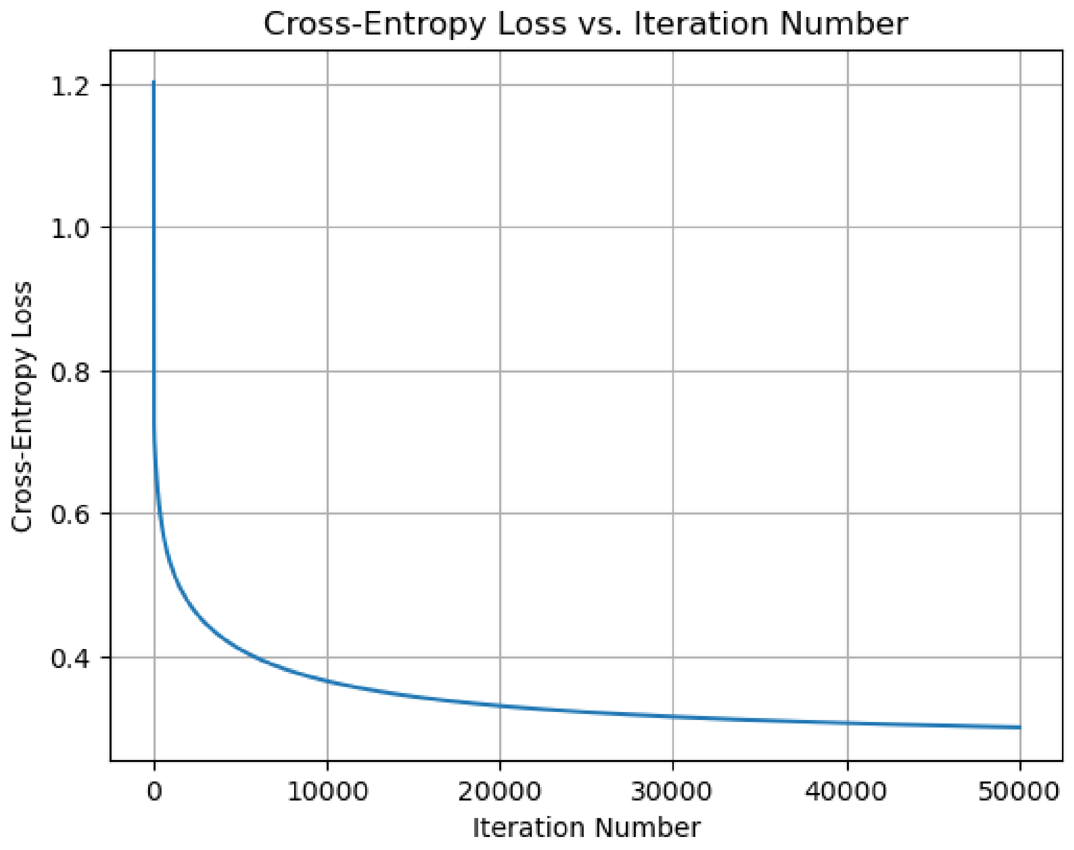
Define functions for batch gradient descent, mini-batch gradient descent, and evaluation

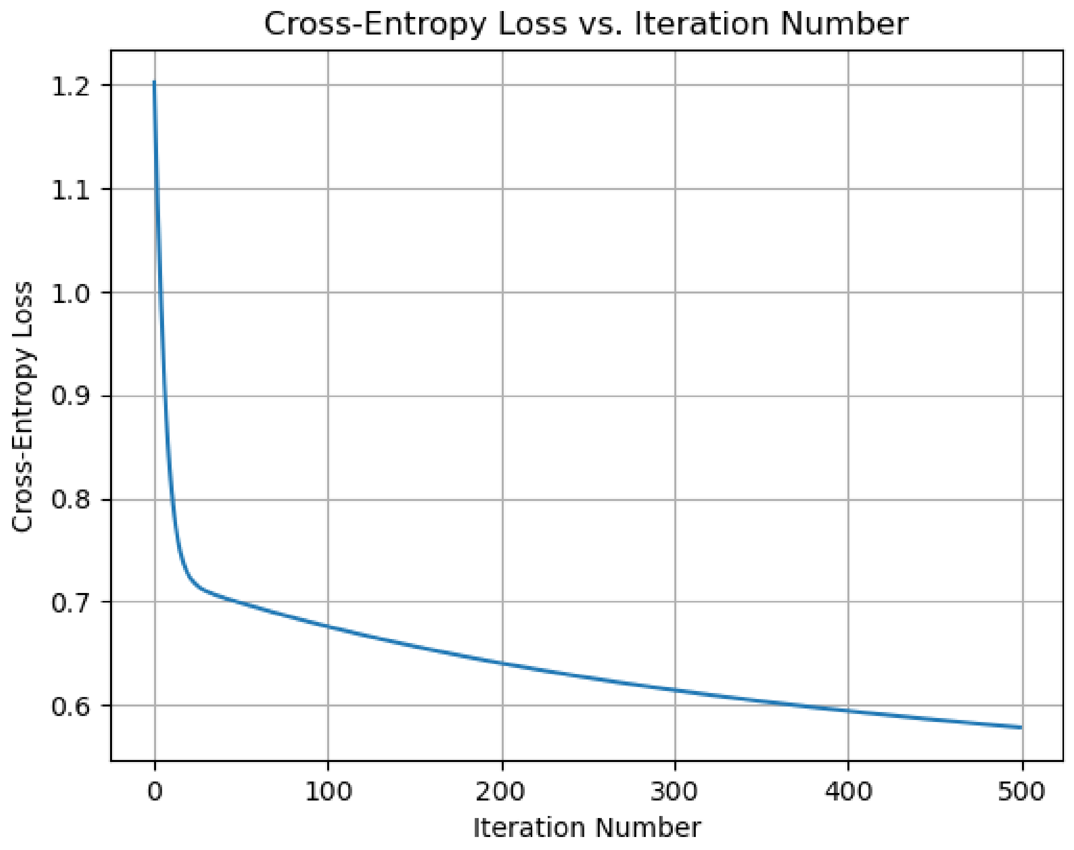
```
[ ] # Returns the sigmoid of the input
    def sigmoid(z):
      return 1.0 / (1.0 + np.exp(-z))
```

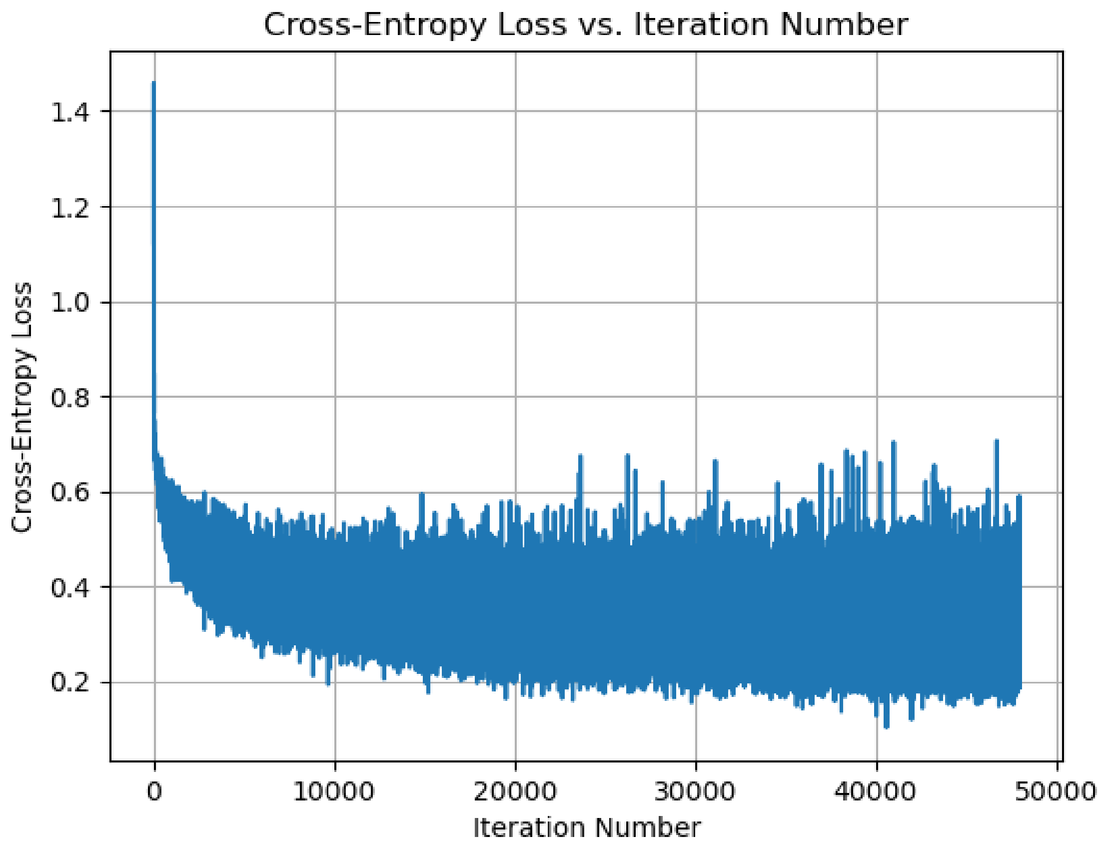
```
[ ] # Returns the probability predictions for a set of inputs X given weights w and bias b
    def get_yhat(X, w, b):
      return sigmoid(np.matmul(X, w) + b)
```

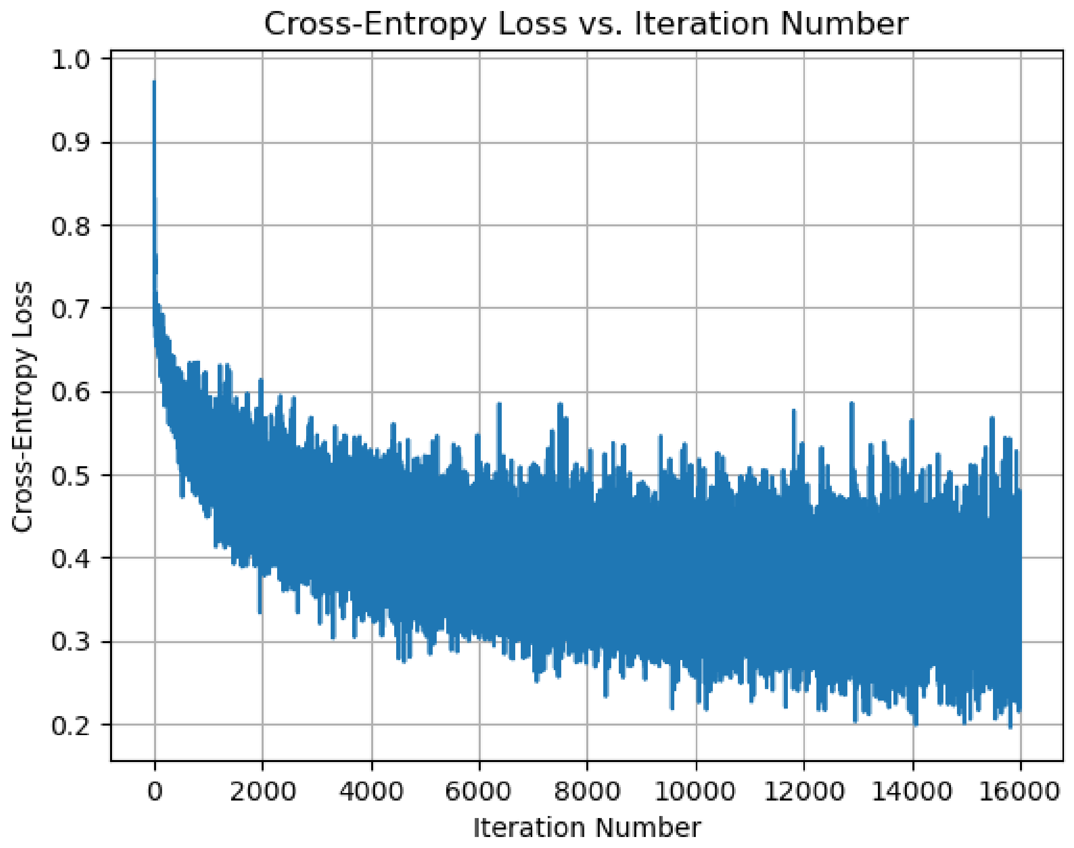
```
[ ] # Returns the cross-entropy loss as well as derivatives w.r.t. weights w and bias b for a set of N training examples (X, y)
    def get_loss_and_derivatives(X, y, w, b, N):
      yhat = get_yhat(X, w, b)
      loss = -1.0/N * np.sum(y * np.log(yhat) + (1.0 - y) * (np.log(1.0 - yhat)))
      dw = np.matmul(X.T, (yhat - y))/N
      db = np.sum(yhat - y)/N
      return loss, dw, db
```

Outcome

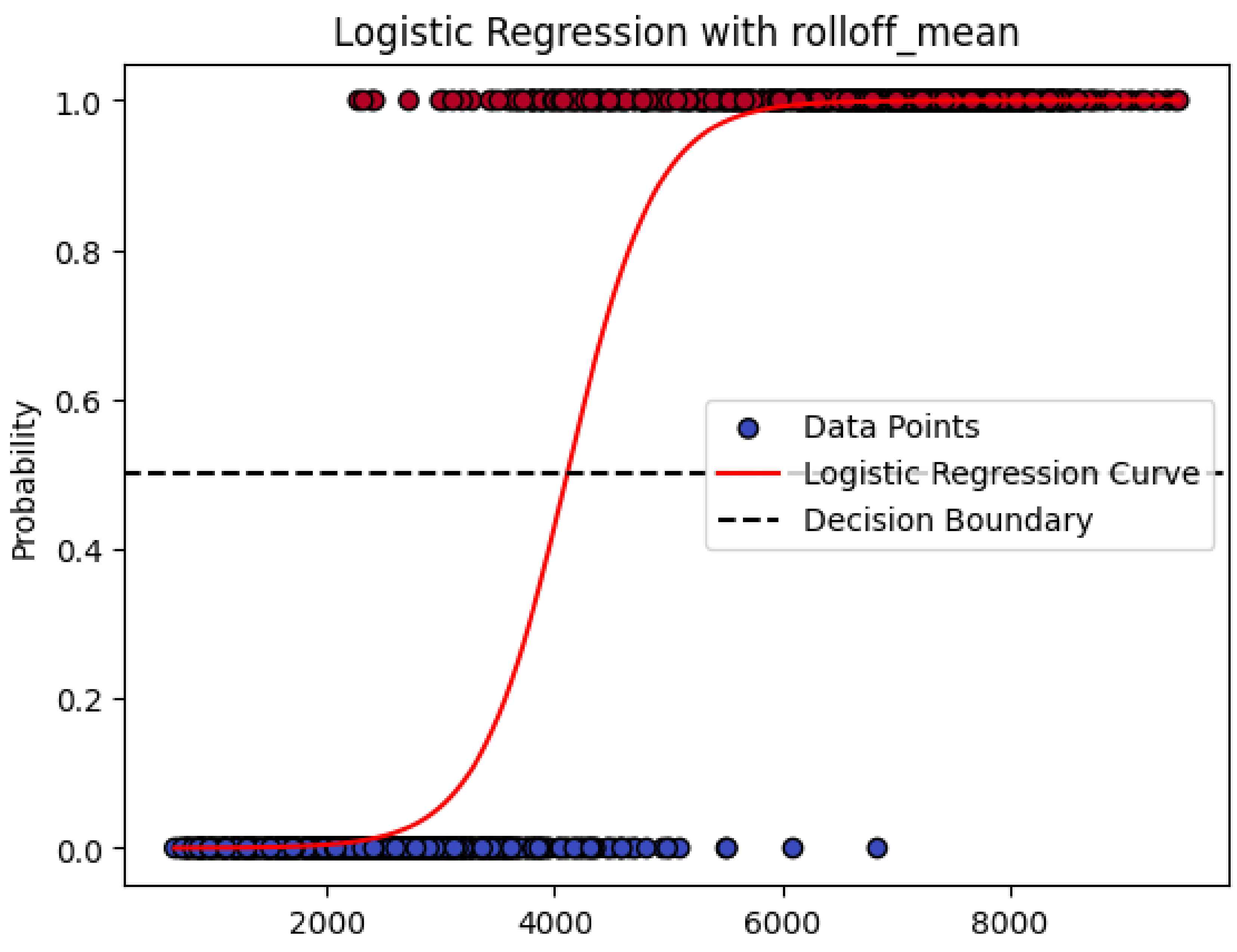

Cross-Entropy Loss vs. Iteration Number
Cross-Entropy Loss
Iteration Number
0.4
0.6
0.8
1.0
1.2
0
10000
20000
30000
40000
50000

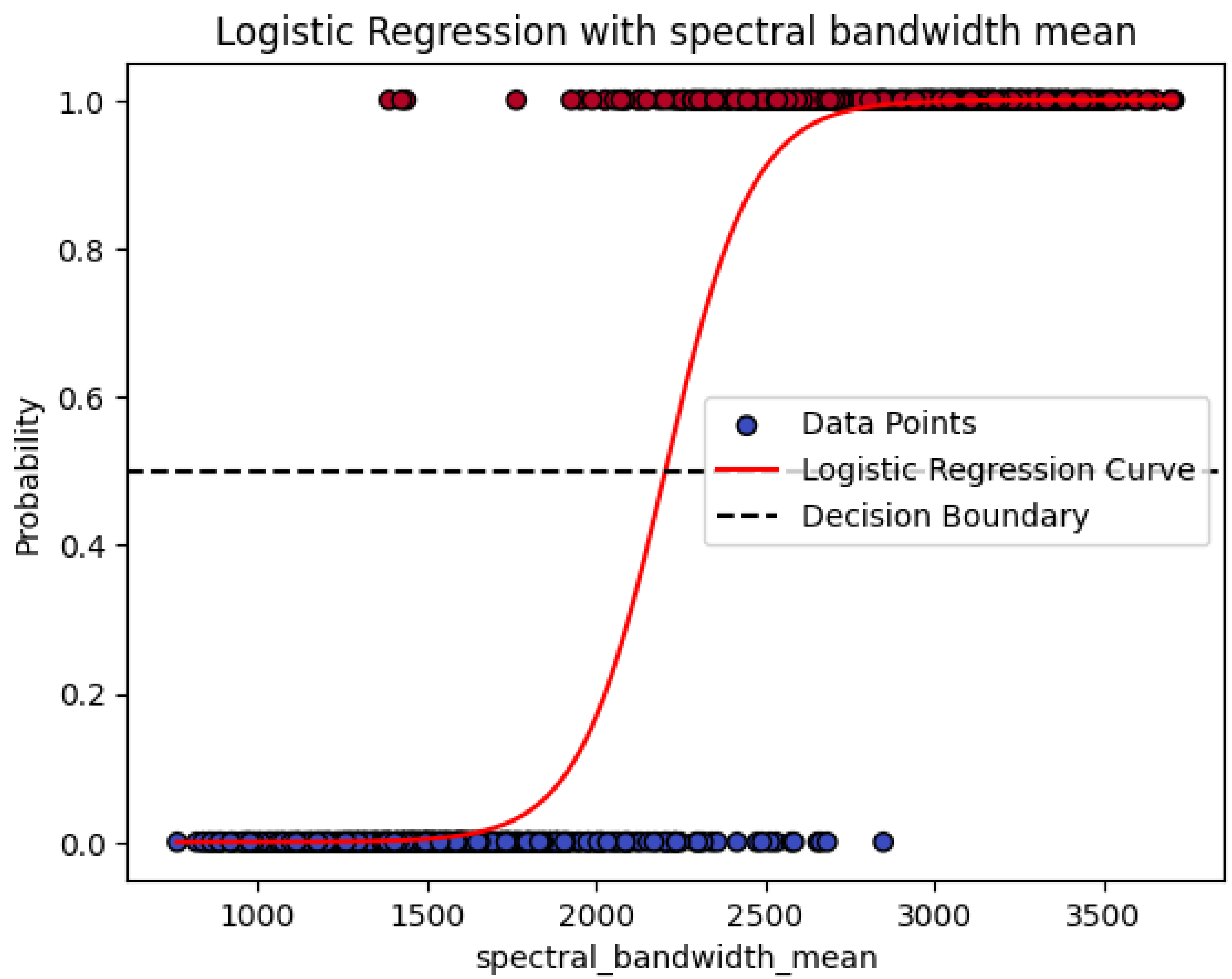

Cross-Entropy Loss vs. Iteration Number
Cross-Entropy Loss
Iteration Number
0.6
0.7
0.8
0.9
1.0
1.1
1.2
0
100
200
300
400
500


Cross-Entropy Loss vs. Iteration Number
Cross-Entropy Loss
Iteration Number
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
10000
20000
30000
40000
50000


Cross-Entropy Loss vs. Iteration Number
Cross-Entropy Loss
Iteration Number
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0
2000
4000
6000
8000
10000
12000
14000
16000

Outcome


Logistic Regression with rolloff_mean
Probability
1.0
0.8
0.6
0.4
0.2
0.0
2000
4000
6000
8000
Data Points
Logistic Regression Curve
Decision Boundary


Logistic Regression with spectral bandwidth mean
Probability
1.0
0.8
0.6
0.4
0.2
0.0
1000
1500
2000
2500
3000
3500
spectral_bandwidth_mean
Data Points
Logistic Regression Curve
Decision Boundary

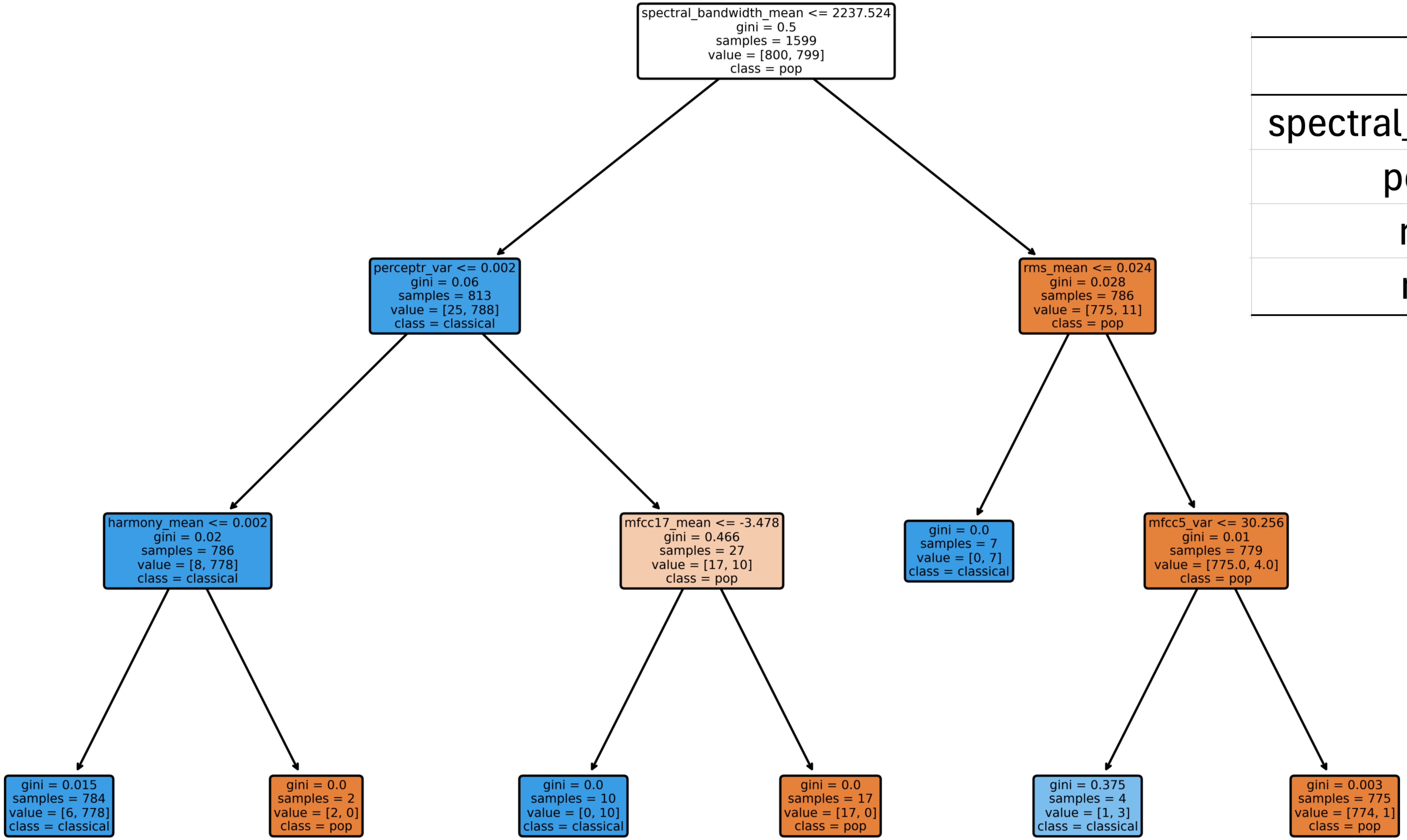
Outcome

Algorithm	Number of iterations	Train accuracy	Test accuracy	Validation accuracy
SGD	/	0.8466	0.85	0.845
Batch Gradient Descent	500	0.759	0.79	0.76
Mini-Batch Gradient Descent	500	0.868	0.87	0.87
Batch Gradient Descent	50000	0.872	0.88	0.87
Mini-Batch Gradient Descent	15000	0.858	0.87	0.865

Decision Tree

- Decision Tree comparation
- Interpretation

Criterion&Max_depth	Accurcy	F1-score	Corss-Val
gini, 1	94.5%	94.2%	96.60%
gini, 3	96.0%	95.8%	97.90%
gini, 5	97.5%	97.4%	96.90%
gini, 10	96.0%	95.8%	96.90%
entropy, 1	94.5%	94.2%	96.60%
entropy, 3	95.5%	95.3%	97.20%
entropy, 5	95.0%	94.7%	96.10%
entropy, 10	95.0%	94.7%	96.20%



Feature	Importance
spectral_bandwidth_mean	0.197
perceptr_var	0.168
rms_mean	0.0972
mfcc5_var	0.0459

Boosting & Bagging

Boosting Result

Firstly , we use GridSearchCV to choose the best hyperparameters, and the hyperparameter tuning is performed using the validation set and train set during this process.

Then ,we fit the model and get the accuracy, precision, recall, f1-score of the test set.

```
Test Set Performance:
Accuracy: 1.0

Classification Report (Test):
              precision    recall  f1-score   support

   classical       1.00      1.00      1.00       100
         pop       1.00      1.00      1.00        99

    accuracy                           1.00       199
   macro avg       1.00      1.00      1.00       199
weighted avg       1.00      1.00      1.00       199
```

eXtreme Gradient Boosting(decision tree model)

```
Test Set Performance:
Accuracy: 0.9949748743718593

Classification Report (Test):
              precision    recall  f1-score   support

   classical       0.99      1.00      1.00       100
         pop       1.00      0.99      0.99        99

    accuracy                           0.99       199
   macro avg       1.00      0.99      0.99       199
weighted avg       1.00      0.99      0.99       199
```

AdaBoost(logistic regression)

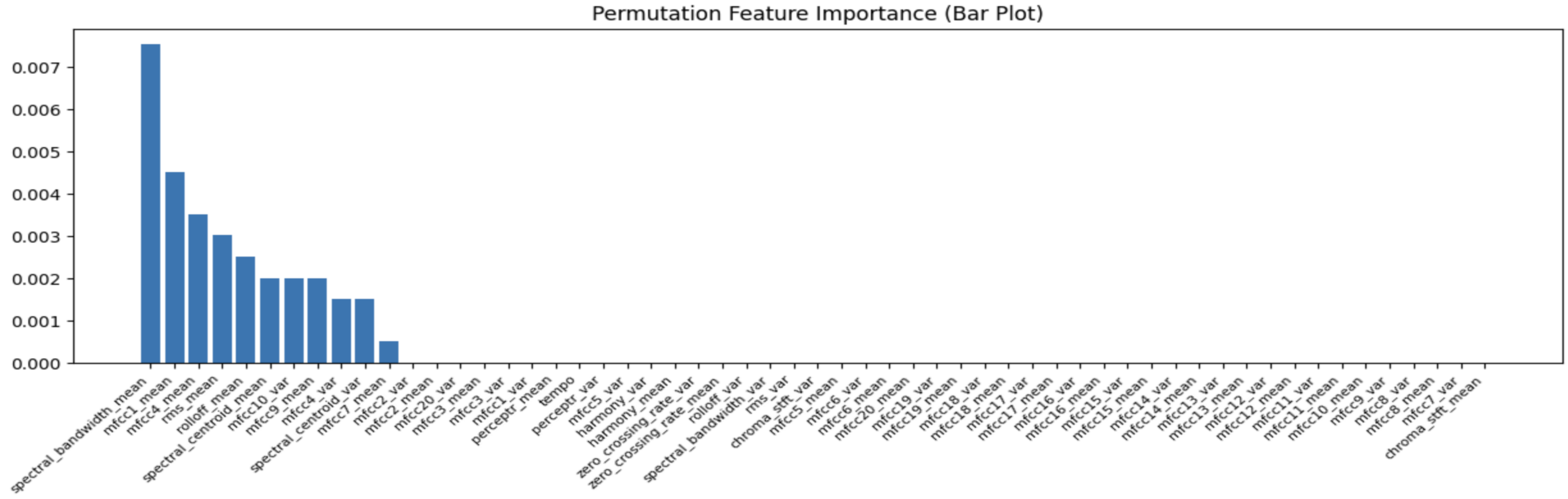
Bagging Classifier

Setting

- Model: BaggingClassifier with Decision tree
- Hyper-parameters:
  - N_estimator: How many models to ensemble
  - Max_depth: max depth of each tree
  - Criterion: gini, entropy, log_loss

Result

- Best parameters:
  - N_esimators = 10
  - Max_depth = 20
  - Criterion = gini
- Accuracies:
  - Validation: **0.97**
  - Test: **0.9899**



Feature Importance

- Spectral bandwidth & mfcc & rolloff
- Perceptual & Rms

Spectral bandwidth & mfcc & rolloff

Classical music typically has a lower spectral bandwidth and rolloff mean.
Pop music tends to have a higher spectral bandwidth and rolloff mean.

Perceptual Variance & rms

Classical music generally shows lower perceptual variance and rms_mean.
Pop music tends to exhibit higher perceptual variance and rms_mean.

Group 16

THANK YOU

Group members:

Rui Gao, Chenfeng Wu, Haorui Wu, Yi Xu, Lin Zhang