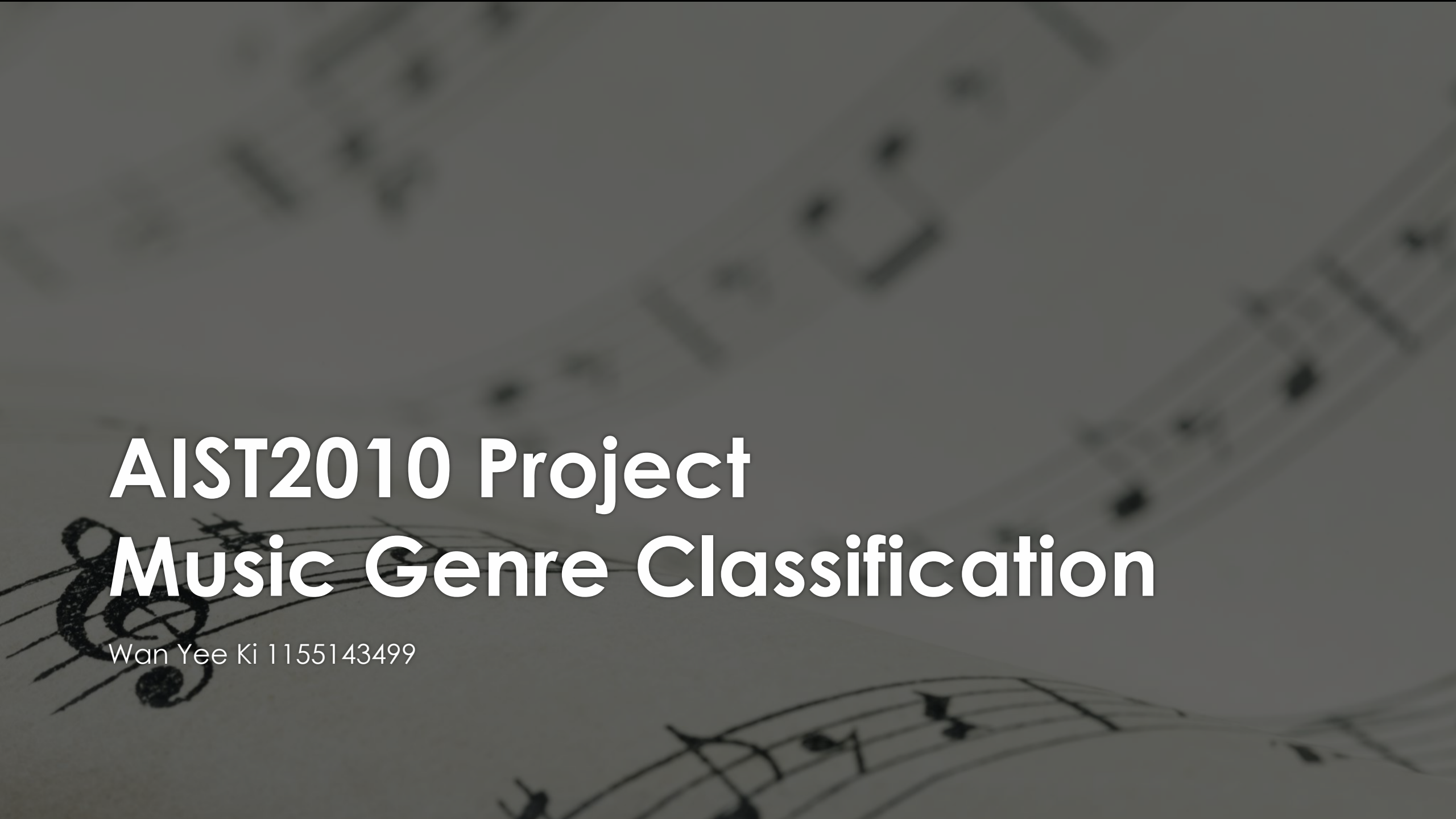
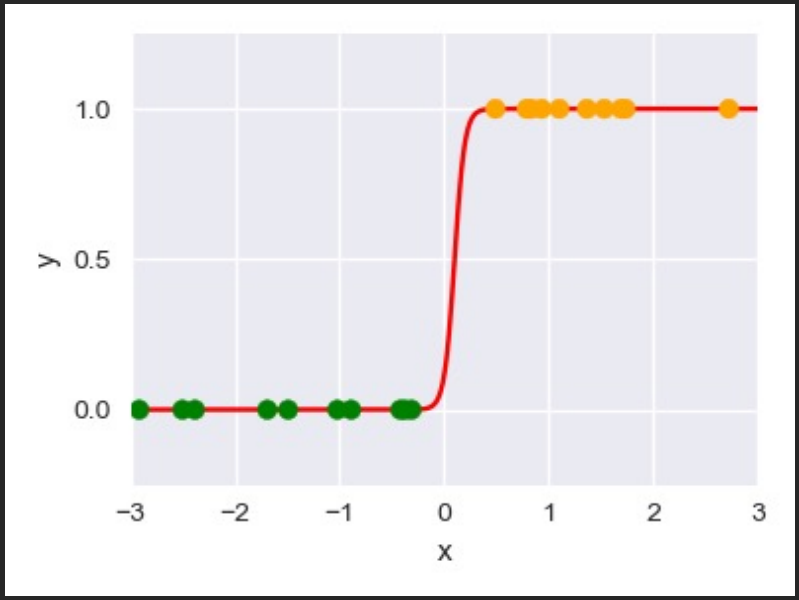


AIST2010 Project
Music Genre Classification

Wan Yee Ki 1155143499

Model explanation

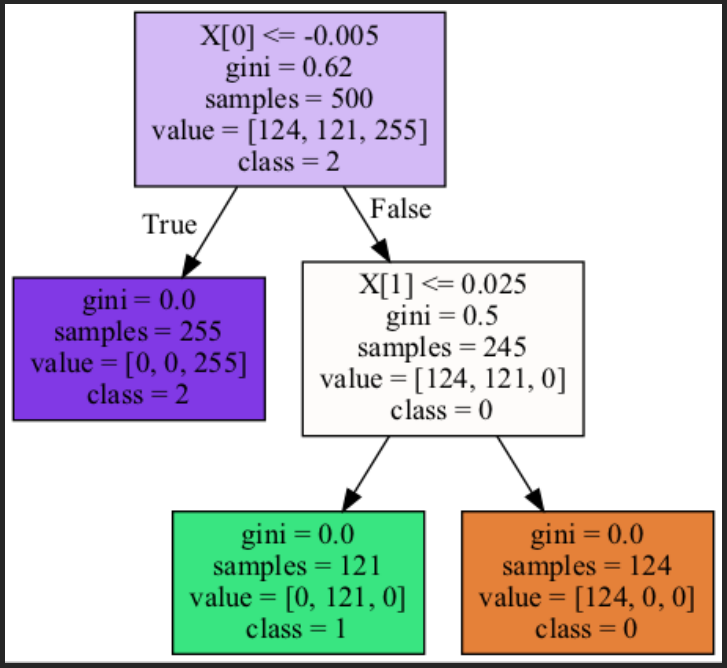
- Logistic Regression
  - Estimating probabilities using a logistic function by the given features




```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from scipy.special import expit
plt.style.use('seaborn')

y = []
X = np.random.uniform(-3, 3, size=(20,1))
for i in X:
    if i > 0:
        y.append(1)
    else:
        y.append(0)

clf = LogisticRegression(C=1e5)
clf.fit(X, y)

plt.figure(1, figsize=(4, 3))
plt.clf()
plt.scatter(X.ravel(), y, c=['orange' if l == 1 else 'green' for l in y], zorder=10)
X_test = np.linspace(-5, 10, 1000)

curve = expit(X_test * clf.coef_ + clf.intercept_).ravel()
plt.plot(X_test, curve, color="red")


plt.ylabel("y")
plt.xlabel("x")
plt.yticks([0, 0.5, 1])
plt.ylim(-0.25, 1.25)
plt.xlim(-3, 3)
plt.tight_layout()
plt.savefig("logistic_regression.png")
```

Model explanation

- Random Forest Classifier
  - Several decision trees are built
  - By averaging the parameters, it improves the predictive accuracy

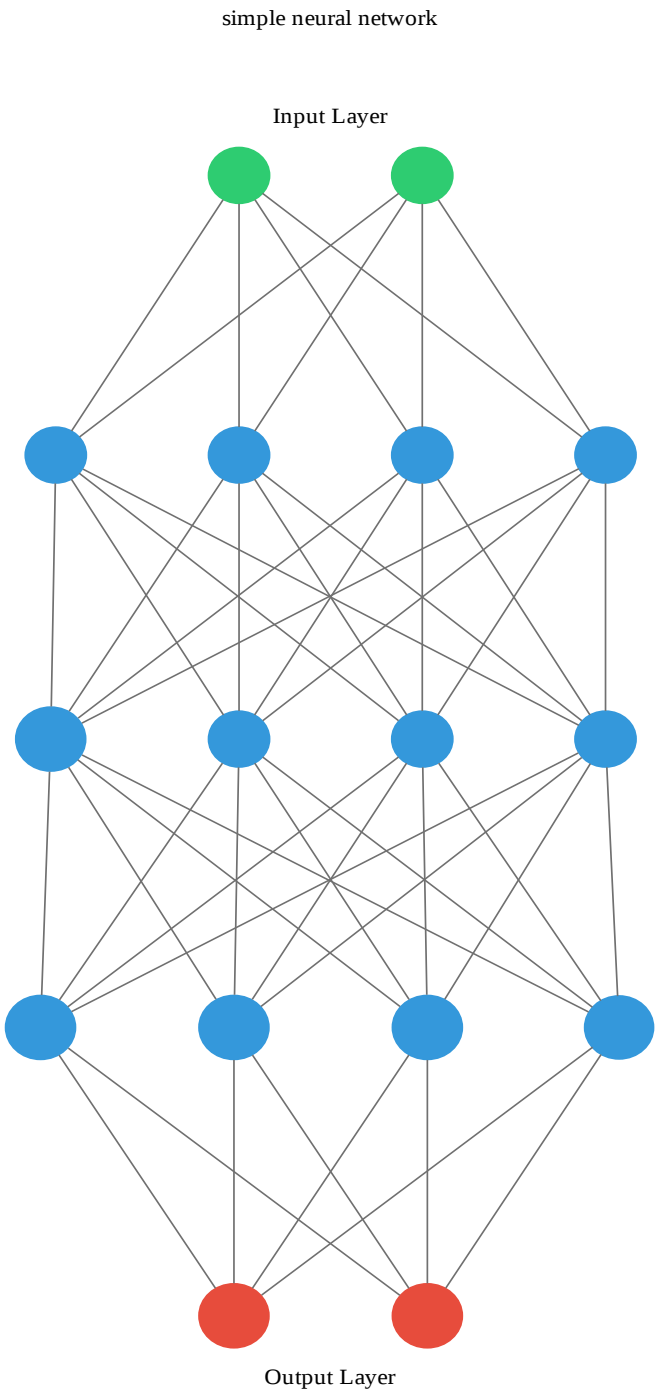



```python
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt
import graphviz

x = np.random.uniform(-10, 10, size=(500, 2))
y = []

for i in x:
    if i[0] > 0:
        if i[1] > 0:
            y.append(0)
        else:
            y.append(1)
    else:
        y.append(2)

clf = DecisionTreeClassifier(max_depth=5)
clf.fit(x, y)

dot_data = tree.export_graphviz(
    clf, filled=True, label='all', class_names=['0', '1', '2'])
graph = graphviz.Source(dot_data, format="png")
graph.render("decision_tree")

```

Model explanation

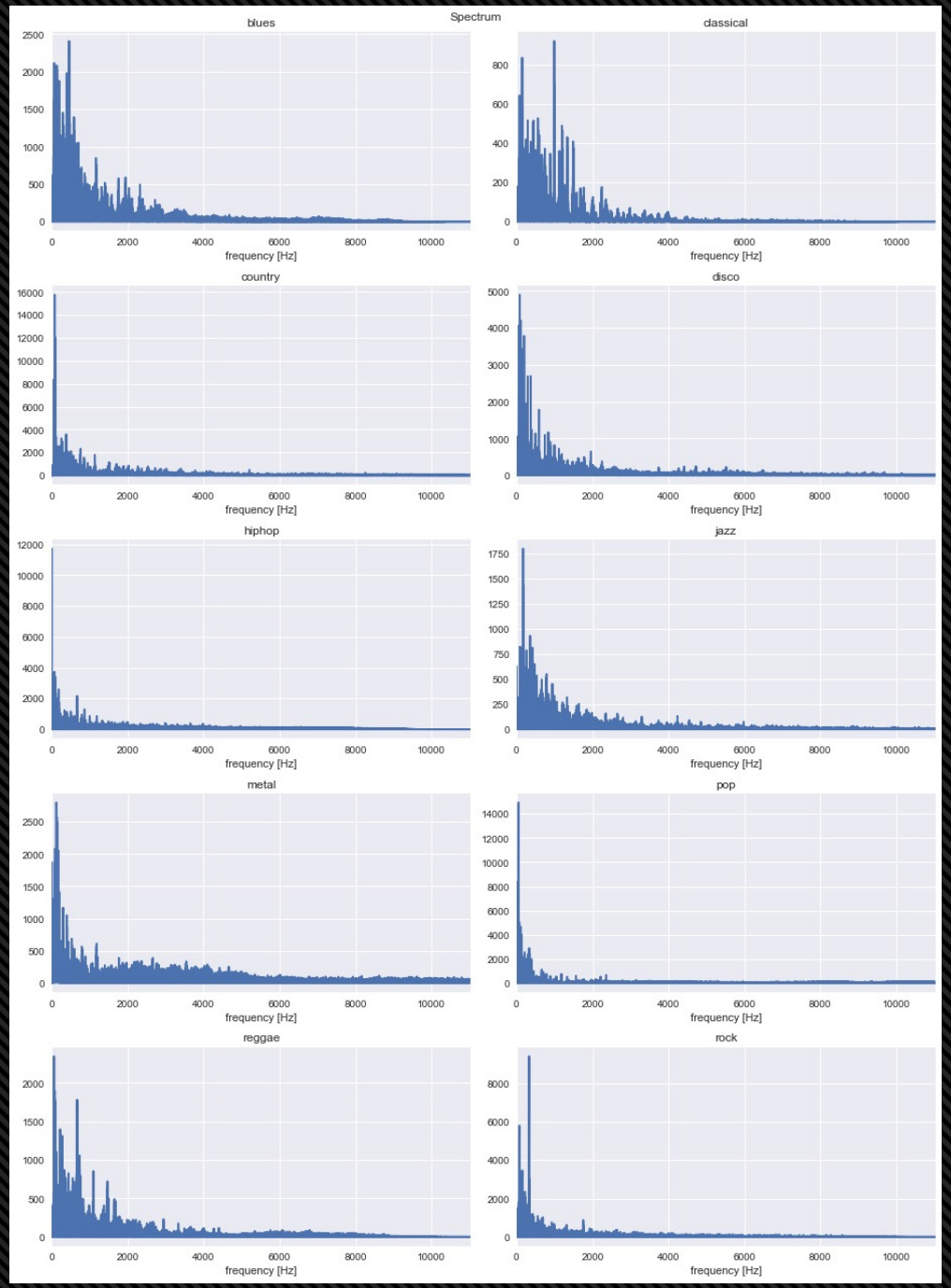
- Simple Neural Network
  - A connected layer
  - Each wire represents a weight
  - Each node represents an activation function

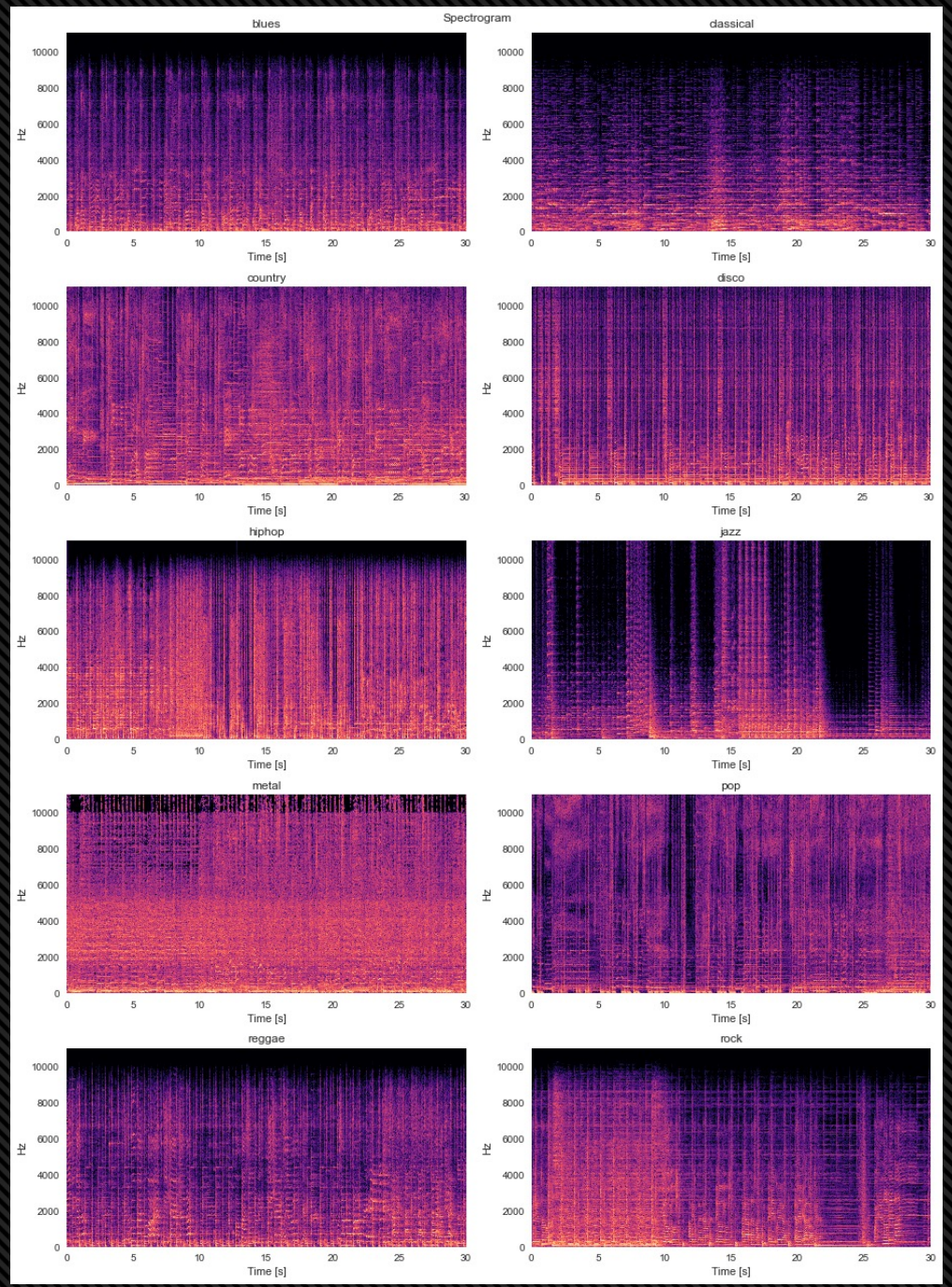
```
model = tf.keras.models.Sequential()

model.add(
    tf.keras.layers.Dense(2, input_shape=(2,)))
model.add(
    tf.keras.layers.Dense(4, activation='relu'))
model.add(
    tf.keras.layers.Dense(4, activation='relu'))
model.add(
    tf.keras.layers.Dense(2, activation='softmax'))

optimiser = tf.keras.optimizers.Adam()
model.compile(
    optimizer=optimiser,
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

model.fit(x, y, epochs=10, batch_size=32)
```



GTZAN Dataset

- Include 100 audio files for each of the following types:
  - blues
  - classical
  - country
  - disco
  - hiphop
  - jazz
  - metal
  - pop
  - reggae
  - rock


Spectrum
blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
frequency [Hz]


Spectrogram
blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
Hz
Time [s]

Approach 1
Extract frequencies with the 2000 highest magnitude

1. Logistic Regression
- Training accuracy: 100%
- Testing accuracy: 12.6%

2. Random Forest Classifier
- Training accuracy: 100%
- Testing accuracy: 35.6%

3. Simple Neural Network
- Training accuracy: 29.6%
- Testing accuracy: 15.6%

4. CNN
- Training accuracy: 100%
- Testing accuracy: 17%

Extracting Spectral features

1. chroma stft mean
2. chroma stft variance
3. rms mean
4. rms variance
5. spectral centroid mean
6. spectral centroid variance
7. spectral bandwidth mean
8. spectral bandwidth variance
9. spectral rolloff mean
10. spectral rolloff variance
11. zero crossing rate mean
12. zero crossing rate variance
13. tempo
14. Mel-frequency cepstral coefficients (20)

Approach 2
Extract spectral features

1. Logistic Regression
- Training accuracy: 47.6%
- Testing accuracy: 44.6%

2. Random Forest Classifier
- Training accuracy: 99.8%
- Testing accuracy: 68.6%

3. Simple Neural Network
- Training accuracy: 99.5%
- Testing accuracy: 72.6%

Approach 3
Extract magnitude among frequency 0 – 11025Hz

1. Logistic Regression
- Training accuracy: 100%
- Testing accuracy: 54%

2. Random Forest Classifier
- Training accuracy: 100%
- Testing accuracy: 55.6%

3. Simple Neural Network
- Training accuracy: 72.4%
- Testing accuracy: 48.3%

4. CNN
- Training accuracy: 93.3%
- Testing accuracy: 53%

Make A Largest Dataset

1. Reverb / Chorus
2. Distortion
3. Pitch shift
4. Time stretch
5. Noise

Approach 4
Extract magnitude among frequency 0 – 11025Hz on a larger dataset

CNN

- Training accuracy: 99%
- Testing accuracy: 98%