```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from keras.losses import BinaryCrossentropy
```

```
data = pd.read_csv('spotify_preprocessed.csv')
```

```
# Shuffle the data
data = data.sample(frac=1).reset_index(drop=True)
```

```
# Split the data into features and labels
X = data.drop(columns=['target'])
y = data['target']
```

```
# Split the data into training and test set
X_train_full, X_test, y_train_full, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```

```
# Split the training set into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train_full, y_train_full, test_size=0.2, random_state=42)
```

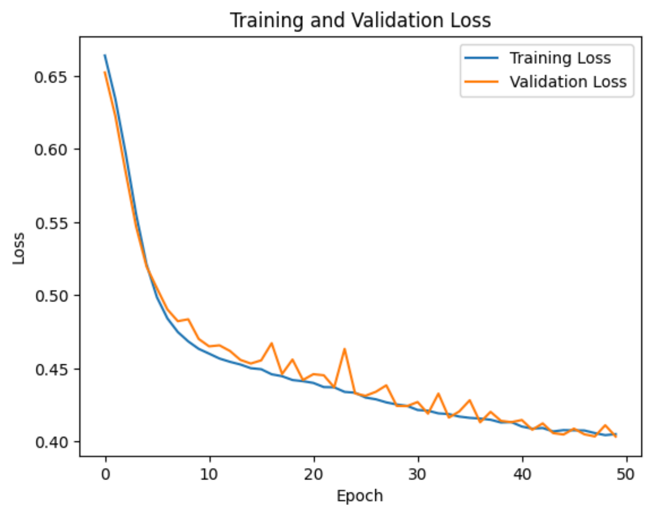
```
# Build the neural network model
def build_model():
    model = Sequential()
    model.add(Dense(32, input_dim=X.shape[1], activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    return model

model = build_model()
```

```
# Compile the model
model.compile(loss=BinaryCrossentropy(),
              optimizer=SGD(learning_rate=0.01),
              metrics=['accuracy'])
```

```
# Train the model
history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=16,
                    validation_data=(X_val, y_val))
```

```
Epoch 43/50
288/288 [==============================] - 1s 3ms/step - loss: 0.4094 - accuracy: 0.8150 - val_loss: 0.4125 - val_accuracy: 0.8203
Epoch 44/50
288/288 [==============================] - 1s 4ms/step - loss: 0.4070 - accuracy: 0.8146 - val_loss: 0.4059 - val_accuracy: 0.8273
Epoch 45/50
288/288 [==============================] - 1s 4ms/step - loss: 0.4079 - accuracy: 0.8111 - val_loss: 0.4049 - val_accuracy: 0.8281
Epoch 46/50
288/288 [==============================] - 1s 4ms/step - loss: 0.4078 - accuracy: 0.8152 - val_loss: 0.4089 - val_accuracy: 0.8177
Epoch 47/50
288/288 [==============================] - 1s 4ms/step - loss: 0.4077 - accuracy: 0.8133 - val_loss: 0.4051 - val_accuracy: 0.8264
Epoch 48/50
288/288 [==============================] - 1s 4ms/step - loss: 0.4059 - accuracy: 0.8131 - val_loss: 0.4036 - val_accuracy: 0.8255
Epoch 49/50
288/288 [==============================] - 1s 4ms/step - loss: 0.4045 - accuracy: 0.8124 - val_loss: 0.4113 - val_accuracy: 0.8151
Epoch 50/50
288/288 [==============================] - 2s 5ms/step - loss: 0.4052 - accuracy: 0.8174 - val_loss: 0.4035 - val_accuracy: 0.8325
```

```
# Plot the training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



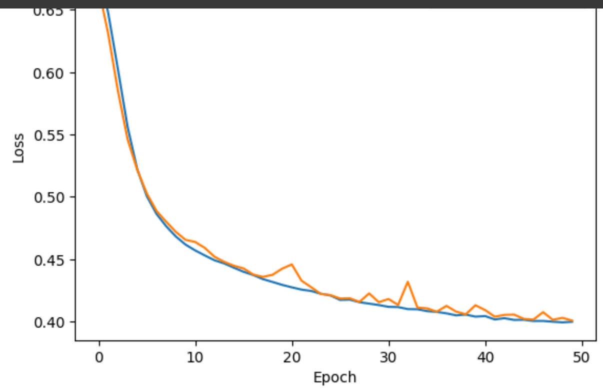
```
# Evaluate the model on the training and validation sets
train_loss, train_accuracy = model.evaluate(X_train, y_train, verbose=0)
val_loss, val_accuracy = model.evaluate(X_val, y_val, verbose=0)
```

```
print(f'Training Loss: {train_loss}, Training Accuracy: {train_accuracy}')
print(f'Validation Loss: {val_loss}, Validation Accuracy: {val_accuracy}')
```

```
Training Loss: 0.3991946578025818, Training Accuracy: 0.8174120783805847
Validation Loss: 0.40354645252227783, Validation Accuracy: 0.8324652910232544
```

```
# Test different model designs
def experiment_with_models():
    configurations = [
        {"layers": [32, 32], "epochs": 50, "batch_size": 16, "learning_rate": 0.01},
        {"layers": [64, 32], "epochs": 50, "batch_size": 16, "learning_rate": 0.01},
        {"layers": [32, 32, 16], "epochs": 50, "batch_size": 16, "learning_rate": 0.01},
        {"layers": [64, 64], "epochs": 50, "batch_size": 16, "learning_rate": 0.01},
        {"layers": [32, 16], "epochs": 50, "batch_size": 16, "learning_rate": 0.01},
    ]

    results = []

    for config in configurations:
        model = Sequential()
        for units in config["layers"]:
            if model.layers:
                model.add(Dense(units, activation='relu'))
            else:
                model.add(Dense(units, input_dim=X.shape[1], activation='relu'))
        model.add(Dense(1, activation='sigmoid'))

        model.compile(loss='binary_crossentropy',
                      optimizer=SGD(learning_rate=config["learning_rate"]),
                      metrics=['accuracy'])

        history = model.fit(X_train, y_train,
                            epochs=config["epochs"],
                            batch_size=config["batch_size"],
                            validation_data=(X_val, y_val),
                            verbose=0)

        train_loss, train_accuracy = model.evaluate(X_train, y_train, verbose=0)
        val_loss, val_accuracy = model.evaluate(X_val, y_val, verbose=0)

        results.append({
            "Configuration": config,
            "Training Loss": train_loss,
            "Training Accuracy": train_accuracy,
            "Validation Loss": val_loss,
            "Validation Accuracy": val_accuracy
        })

        # Plot the training and validation loss for each configuration
        plt.figure()
        plt.plot(history.history['loss'], label='Training Loss')
        plt.plot(history.history['val_loss'], label='Validation Loss')
        plt.title(f'Training and Validation Loss for config {config}')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.legend()
        plt.show()
    return pd.DataFrame(results)
```

```
results_df = experiment_with_models()
print(results_df)
```



Training and Validation Loss for config {'layers': [32, 16], 'epochs': 50, 'batch_size': 16, 'learning_rate': 0.01}