محمد جواد رنجبر

۸۱۰۱۰۱۱۷۳

پروژه امتیازی درس پردازش گفتار:

تشخیص احساسات زبان فارسی با استفاده از یادگیری عمیق

دکتر ویسی

بهار ۱۴۰۳

Contents

گزارش فایل کد ساده

لود کردن کتابخانه‌ها و دانلود داده‌ها

ابتدا کتابخانه‌های مورد نیاز را نصب کرده و داده‌ها را از منبع مورد نظر دانلود می‌کنیم:

```
[ ] pip install transformers datasets torch accelerate librosa numpy scikit-learn keras

[ ] !kaggle datasets download -d mansourehk/shemo-persian-speech-emotion-detection-database

[ ] !unzip shemo-persian-speech-emotion-detection-database.zip
```

تصویر ۱ نصب کتابخانه و مجموعه داده

همچنین داده‌ها را از حالت زیپ خارج می‌کنیم.

تمیز سازی و نمایش داده‌ها:

حال داده‌ها به دو دسته زن و مرد تقسیم شده‌اند در این بخش ما به جنسیت آن‌ها کار ندارم و فقط قصد داریم یک مدل عمومی برای تشخیص احساسات بسازیم، بنابراین داده‌ها را با یکدیگر ترکیب می‌کنیم و یک دیتافریم از کل داده‌ها با احساسات مربوط به آن‌ها نگه می‌داریم.

همچنین مدت زمان هر داده را نیز استخراج می‌کنیم و دیتافریم می‌گذاریم.

```
[ ]
    import librosa

    # Function to calculate duration in seconds for an audio file
    def get_duration(file_path):
        audio, sr = librosa.load(file_path)
        duration_sec = librosa.get_duration(y=audio, sr=sr)
        return duration_sec

    # Assuming 'df' contains a column named 'path' with file paths
    # Add a new column 'duration_sec' to store voice durations
    df['duration_sec'] = df['path'].apply(lambda x: get_duration(x))
```

تصویر ۲ به دست آوردن طول داده‌ها

نمایش داده‌ها:

ابتدا دیتا فریم به شکل زیر می‌باشد:

```
[ ] df.head()
```

	gender	speaker_code	emotion	utterance_number	path	duration_sec
0	M	06	N	04.wav	/content/dataset/male/M06N04.wav	1.082676
1	M	25	N	15.wav	/content/dataset/male/M25N15.wav	6.294694
2	M	16	W	02.wav	/content/dataset/male/M16W02.wav	0.979138
3	M	05	N	18.wav	/content/dataset/male/M05N18.wav	3.016508
4	M	05	N	02.wav	/content/dataset/male/M05N02.wav	1.577052

تصویر ۳ تعداد از داده‌ها

همچنین داده‌ها به شکل زیر می‌باشد:

```
df.describe(include='all')
```

	gender	speaker_code	emotion	utterance_number	path	duration_sec
count	3000	3000	3000	3000	3000	3000.000000
unique	2	56	6	87	3000	NaN
top	M	12	A	01.wav	/content/dataset/female/F07A44.wav	NaN
freq	1737	208	1059	341	1	NaN
mean	NaN	NaN	NaN	NaN	NaN	4.111807
std	NaN	NaN	NaN	NaN	NaN	3.414900
min	NaN	NaN	NaN	NaN	NaN	0.358639
25%	NaN	NaN	NaN	NaN	NaN	1.955113
50%	NaN	NaN	NaN	NaN	NaN	3.079637
75%	NaN	NaN	NaN	NaN	NaN	5.104093
max	NaN	NaN	NaN	NaN	NaN	33.328027

تصویر ٤ ویژگی‌های اماری دیتاست

دارای ۳۰۰۰ داده هستیم که به صورت میانگین ۴ ثانیه هستند. همچنین توزیع جنسیت زن و مرد به شکل زیر می‌باشد:

```
[ ] # Value counts for the 'gender' column
    gender_counts = df['gender'].value_counts()

    # Display the value counts for gender
    print(gender_counts)
```

```
gender
M    1737
F    1263
Name: count, dtype: int64
```

تصویر ٥ توزیع جنسیت

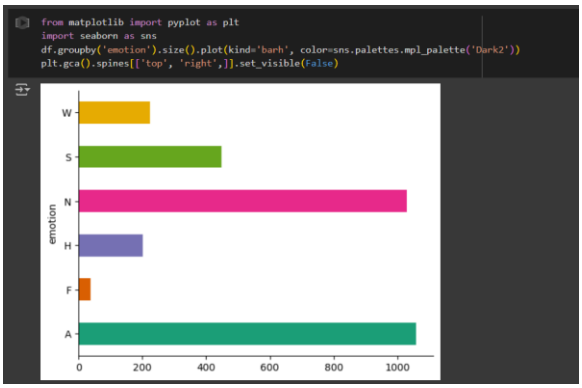
همچنین توزیع احساسات مختلف به شکل زیر می‌باشد:

```
[ ] # Value counts for the 'gender' column
    gender_counts = df['emotion'].value_counts()

    # Display the value counts for gender
    print(gender_counts)
```

```
emotion
A    1059
N    1028
S     449
W     225
H     201
F      38
Name: count, dtype: int64
```

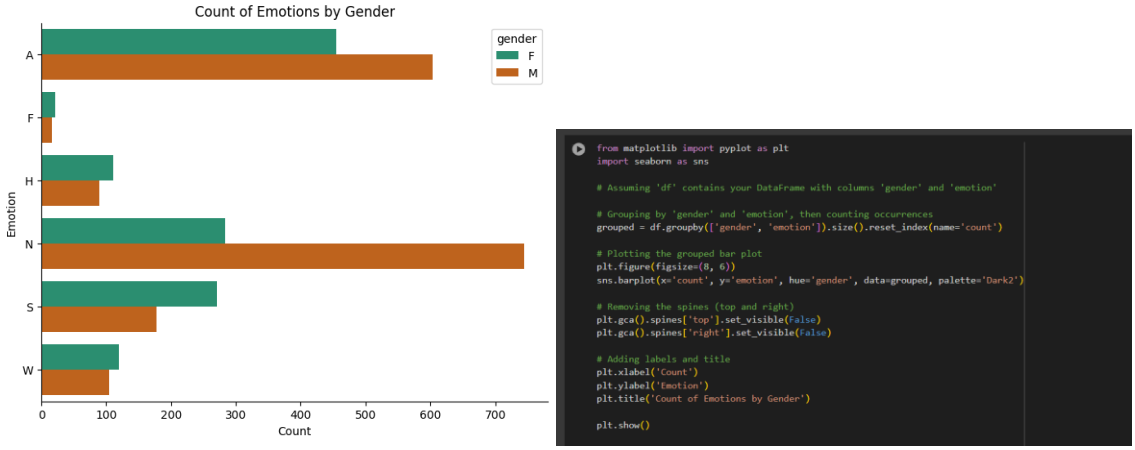
تصویر ٦ توزیع کلاس‌ها

همچنین هیستگورام توزیع داده‌ها به شرح زیر می‌باشد:



تصویر ۷ هیستوگرام داده‌ها

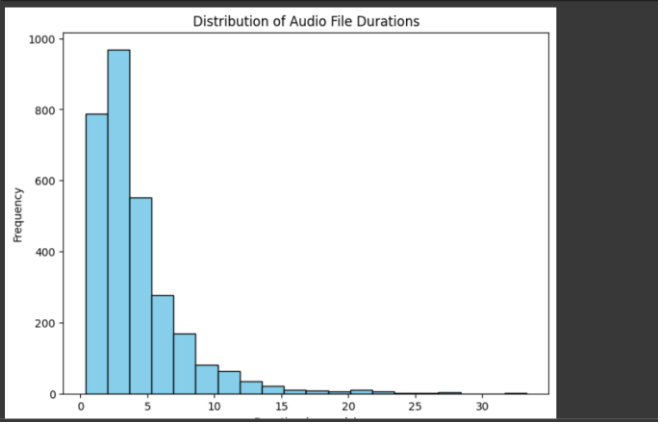
همچنین توزیع احساسات برای جنسیت‌های مختلف به شکل زیر به دست می‌آید:



تصویر 8 هیستوگرام برای هر کلاس

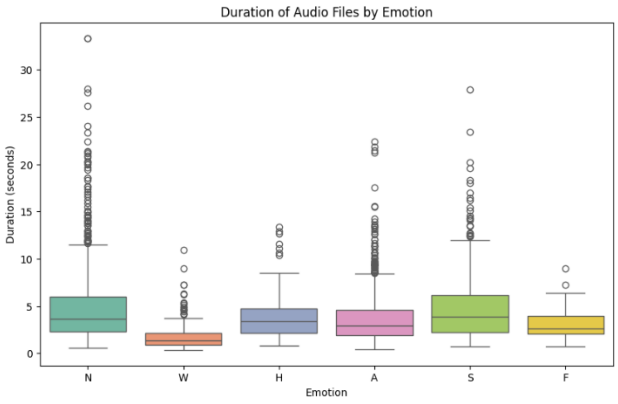
همچنین هیستوگرام مدت زمان صوت‌ها به شکل زیر می‌باشد:

```python
import matplotlib.pyplot as plt

# Plotting a histogram of durations
plt.figure(figsize=(8, 6))
plt.hist(df['duration_sec'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Duration (seconds)')
plt.ylabel('Frequency')
plt.title('Distribution of Audio File Durations')
plt.show()
```



تصویر ۹ هیستوگرام طول صوت‌ها

همچین باکس پلات مربوط به احساسات و مدت زمان آن‌ها به شکل زیر می‌باشد:




```python
import seaborn as sns

# Boxplot of duration by emotion
plt.figure(figsize=(10, 6))
sns.boxplot(x='emotion', y='duration_sec', data=df, palette='Set2')
plt.xlabel('Emotion')
plt.ylabel('Duration (seconds)')
plt.title('Duration of Audio Files by Emotion')
plt.show()
```

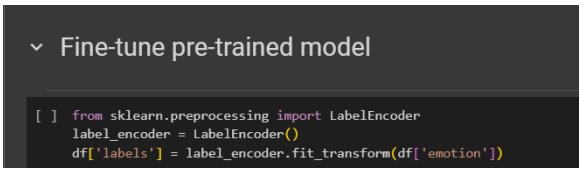
تصویر 10 باکس پلات داده‌ها

همچنین برای جنسیت‌های مختلف به شکل زیر می‌باشد:

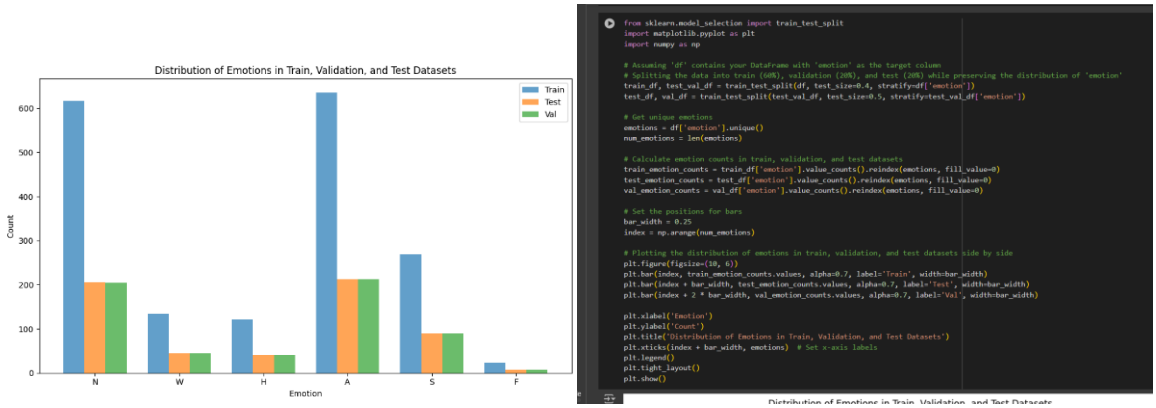
تصویر ۱۱ باکس پلات جنسیت‌ها

آماده‌سازی داده‌ها برای آموزش:

ابتدا داده‌ها را از حروف انگلیسی به اعداد تبدیل می‌کنیم تا مدل بتواند آن‌ها را پیش‌بینی کند:



حال داده‌ها را به سه دسته آموزش، ارزیابی و آزمون تقسیم می‌کنیم، در این تقسیم توزیع داده‌ها در کلاس‌های مختلف حفظ می‌شود:



تصویر 12 داده‌های آموزش، ارزیابی و آزمون

تعریف تابع map_to_array

ابتدا تابع map_to_arrayتعریف شده که وظیفه‌اش بارگذاری فایل‌های صوتی و تبدیل آن‌ها به آرایه‌های numpy است. از کتابخانه‌ی librosaبرای این منظور استفاده شده است.

تبدیل داده‌ها به فرمت Dataset

داده‌های آموزشی(train_df) ، آزمایشی (test_df) و اعتبارسنجی (val_df) که در قالبDataFrame های pandas هستند، به فرمت Datasetتبدیل شده و سپس تابع map_to_arrayروی آن‌ها اعمال می‌شود.

```
def map_to_array(example):
    speech, _ = librosa.load(example["path"], sr=16000, mono=True)
    example["speech"] = speech
    return example

# Assuming train_df, test_df, val_df are pandas DataFrames with 'path' and 'labels' columns
train_data = Dataset.from_pandas(train_df).map(map_to_array)
test_data = Dataset.from_pandas(test_df).map(map_to_array)
val_data = Dataset.from_pandas(val_df).map(map_to_array)
```

تصویر ۱۳ بارگزاری داده‌ها

تعریف تابع پیش‌پردازش

یک تابع پیش‌پردازش تعریف شده که ویژگی‌های صوتی را استخراج کرده و برچسب‌ها را اضافه می‌کند. از feature_extractor که با استفاده از مدل پیش‌آماده‌ی facebook/hubert-base-ls960ایجاد شده، استفاده می‌شود.

اعمال تابع پیش‌پردازش

با استفاده از feature_extractor، تابع پیش‌پردازش روی داده‌های آموزشی، آزمایشی و اعتبارسنجی اعمال می‌شود.

```
# Apply preprocessing
feature_extractor = Wav2Vec2FeatureExtractor.from_pretrained("facebook/hubert-base-ls960")

train_encodings = train_data.map(preprocess_function, remove_columns="speech", batched=True)
test_encodings = test_data.map(preprocess_function, remove_columns="speech", batched=True)
val_encodings = val_data.map(preprocess_function, remove_columns="speech", batched=True)

train_dataset = train_encodings.with_format("torch")
test_dataset = test_encodings.with_format("torch")
val_dataset = val_encodings.with_format("torch")
```

تصویر ۱۴ پیش‌پردازش داده‌ها

تبدیل داده‌ها به فرمتPyTorch

.تبدیل می‌شوند تا برای استفاده در مدل‌های یادگیری عمیق آماده شوند PyTorch در نهایت، داده‌های کدگذاری شده به فرمت

```
train_dataset = train_encodings.with_format("torch")
test_dataset = test_encodings.with_format("torch")
val_dataset = val_encodings.with_format("torch")
```

تصویر 15 تبدیل داده‌ها به فرمت torch

تعریف مدل و آماده‌سازی محیط

در این کد، مدل Wav2Vec2ForSequenceClassificationاز کتابخانه‌ی transformersبرای انجام وظیفه‌ی طبقه‌بندی توالی‌های صوتی استفاده شده است. مدل بر پایه‌ی facebook/hubert-base-ls960است و برای طبقه‌بندی ۶ دسته تنظیم شده است.

```
num_labels=6
# Define the model
model = Wav2Vec2ForSequenceClassification.from_pretrained("facebook/hubert-base-ls960", num_labels=num_labels)
```

تصویر ۱۶ لود کردن مدل

تعریف Callback سفارشی برای ثبت معیارها

برای ثبت معیارهای آموزشی و ارزیابی، یک Callback سفارشی به نام LoggingCallbackتعریف شده است. این کلاس معیارهای مانند train_loss، eval_loss، train_accو eval_accرا ذخیره می‌کند.

```python
# Custom callback to log metrics
class LoggingCallback(TrainerCallback):
    def __init__(self):
        self.train_loss = []
        self.eval_loss = []
        self.train_acc = []
        self.eval_acc = []

    def on_log(self, args, state, control, logs=None, **kwargs):
        if "loss" in logs:
            self.train_loss.append(logs["loss"])
        if "eval_loss" in logs:
            self.eval_loss.append(logs["eval_loss"])
        if "eval_accuracy" in logs:
            self.eval_acc.append(logs["eval_accuracy"])
        if "accuracy" in logs:
            self.train_acc.append(logs["accuracy"])
```

تصویر ۱۷ تعریف call back

تعریف معیار دقت

برای محاسبه‌ی معیار دقت از کتابخانه‌ی datasetsاستفاده شده است.

```python
# Define accuracy metric
accuracy_metric = load_metric("accuracy")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return accuracy_metric.compute(predictions=predictions, references=labels)
```

تصویر ۱۸ تعریف دقت

تنظیمات آموزشی

پارامترهای آموزشی با استفاده از کلاس TrainingArgumentsتنظیم شده‌اند. این پارامترها شامل تنظیمات مربوط به تعداد اپوک‌ها، اندازه‌ی بچ‌ها و مکان ذخیره‌سازی نتایج و لاگ‌ها هستند.

```
# Training arguments
training_args = TrainingArguments(
    output_dir="./results",
    eval_strategy="epoch",
    per_device_train_batch_size=64,
    per_device_eval_batch_size=64,
    num_train_epochs=10,
    logging_dir="./logs",
    logging_steps=10,
)
```

تصویر ۱۹ تعریف آرگمان‌های آموزش

تعریفTrainer

کلاس Trainerبرای مدیریت آموزش و ارزیابی مدل استفاده شده است. این کلاس مدل، داده‌های آموزشی و اعتبارسنجی، توکنایزر، معیارهای محاسبه‌شده و Callback سفارشی را دریافت می‌کند.

```
# Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=feature_extractor,
    compute_metrics=compute_metrics,
    callbacks=[logging_callback],
)
```

تصویر ۲۰ ساخت trainer

آموزش مدل

مدل با استفاده از trainer.train()آموزش داده می‌شود و سپس با trainer.evaluate(test_dataset)ارزیابی می‌شود.

رسم نمودارهای از دست دادن و دقت

در نهایت، نمودارهای از دست دادن و دقت برای دوره‌های آموزشی و ارزیابی رسم می‌شوند.

```python
# Plotting loss and accuracy
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(logging_callback.train_loss, label="Train Loss")
plt.plot(logging_callback.eval_loss, label="Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training and Validation Loss")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(logging_callback.train_acc, label="Train Accuracy")
plt.plot(logging_callback.eval_acc, label="Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Validation Accuracy")
plt.legend()

plt.show()
```

تصویر ۲۱ نمایش عملکرد مدل در حین آموزش

پیش‌بینی بر روی مجموعه‌ی تست

مدل آموزش‌دیده با استفاده از trainer.predict بر روی مجموعه‌ی تست اعمال می‌شود و پیش‌بینی‌ها استخراج می‌شوند.

```python
# Make predictions on the test set
predictions = trainer.predict(test_dataset)
preds = np.argmax(predictions.predictions, axis=-1)
```

تصویر ۲۲ پیش‌بینی داده‌های آزمون

تولید گزارش طبقه‌بندی

گزارش طبقه‌بندی که شامل معیارهایی مثل دقت، بازخوانی و امتیاز F1 برای هر کلاس است، با استفاده از classification_report از کتابخانه‌ی sklearn تولید می‌شود.

```python
# Generate classification report
report = classification_report(test_dataset["labels"], preds)
print("Classification Report:\n", report)
```

تصویر ۲۳ گزارش طبقه‌بندی

تولید ماتریس درهمریختگی

ماتریس درهمریختگی برای نمایش تعداد نمونه‌های درست و نادرست طبقه‌بندی شده برای هر کلاس تولید می‌شود و با استفاده از ConfusionMatrixDisplay از کتابخانه‌ی sklearn رسم می‌شود.

```python
# Generate confusion matrix
cm = confusion_matrix(test_dataset["labels"], preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.show()
```

تصویر ۲۴ ماتریس درهمریختگی

تمام بخش‌های بالا برای مدل‌های دیگر تکرار شده است، تنها فرق این است که نام مدل‌های دیگر برای دانلود داده شده‌اند.

مدل‌های کلاسیک:

برای این مدل‌ها باید از روش‌های استخراج ویژگی مربوط به صوت استفاده کنیم بنابراین داریم:

تعریف تابع extract_features

تابع extract_featuresکه سه پارامتر mfcc، chromaو melدارد که تعیین می‌کنند کدام ویژگی‌ها استخراج شوند. ورودی این تابع مسیر فایل صوتی (file_path) است.

بارگذاری فایل صوتی: فایل صوتی با استفاده از librosa.loadبارگذاری می‌شود که به طور خودکار نرخ نمونه‌برداری (sampling rate)را نیز برمی‌گرداند.

استخراج MFCC: اگر پارامتر mfccبرابر با Trueباشد، ویژگی‌های MFCC با استفاده از librosa.feature.mfccاستخراج شده و میانگین آن‌ها محاسبه و به لیست ویژگی‌ها اضافه می‌شود.

استخراج کرومای استروف: اگر پارامتر chromaبرابر با Trueباشد، ویژگی‌های کرومای استروف با استفاده از librosa.feature.chroma_stftاستخراج شده و میانگین آن‌ها محاسبه و به لیست ویژگی‌ها اضافه می‌شود.

استخراج مل‌سپکتروگرام: اگر پارامتر melبرابر با Trueباشد، ویژگی‌های مل‌سپکتروگرام با استفاده از librosa.feature.melspectrogramاستخراج شده و میانگین آن‌ها محاسبه و به لیست ویژگی‌ها اضافه می‌شود.

.

```python
import librosa
import numpy as np

def extract_features(file_path, mfcc=True, chroma=True, mel=True):
    with open(file_path, 'rb') as f:
        try:
            audio, sr = librosa.load(f)
            features = []
            if mfcc:
                mfccs = np.mean(librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13).T, axis=0)
                features.extend(mfccs)
            if chroma:
                chroma = np.mean(librosa.feature.chroma_stft(y=audio, sr=sr).T, axis=0)
                features.extend(chroma)
            if mel:
                mel = np.mean(librosa.feature.melspectrogram(y=audio, sr=sr).T, axis=0)
                features.extend(mel)
        except Exception as e:
            print(f"Error encountered while parsing file: {file_path}")
            return None

    return features
```

تصویر ۲۵ تابع استخراج ویژگی

حال با استفاده از دیتافریم ویژگی‌های مربوط به همه‌ی صوت‌ها را استخراج می‌کنیم:

```python
[ ]  # Load data for training, validation, and testing
    def load_data_from_df(df):
        X = []
        y = []
        for index, row in df.iterrows():
            file_path = row['path']  # Assuming 'file_path' is the column name for audio file paths
            emotion_label = row['emotion']  # Assuming 'emotion' is the column name for emotion labels
            features = extract_features(file_path)
            if features is not None:
                X.append(features)
                y.append(emotion_label)
        return X, y
    X_train, y_train = load_data_from_df(train_df)
    X_val, y_val = load_data_from_df(val_df)
    X_test, y_test = load_data_from_df(test_df)
```

تصویر ۲۶ استخراج ویژگی از داده‌ها

حال مدل‌های مختلفی را برای این کار آموزش می‌دهیم:

مدل SVM

تعریف و آموزش مدل SVM

ابتدا مدل SVM با هسته‌ی خطی (linear kernel) و حالت تصادفی ۴۲ تعریف می‌شود. سپس مدل با استفاده از داده‌های آموزشی آموزش داده می‌شود.

```python
# Initialize SVM classifier
svm_model = SVC(kernel='linear', random_state=42)

# Train SVM model
svm_model.fit(X_train, y_train)
```

تصویر ۲۷ آموزش مدل SVM

پیش‌بینی و ارزیابی مدل

مدل آموزش‌دیده بر روی مجموعه‌ی تست اعمال شده و پیش‌بینی‌ها محاسبه می‌شوند. سپس دقت تست و گزارش طبقه‌بندی محاسبه و نمایش داده می‌شوند.

```python
# Predict on test set
y_pred_test = svm_model.predict(X_test)

# Evaluate test accuracy
test_accuracy = accuracy_score(y_test, y_pred_test)
print(f"Test Accuracy: {test_accuracy}")
```

تصویر ۲۸ پیش‌بینی مدل SVM

تولید و نمایش ماتریس درهمریختگی

ماتریس درهمریختگی گمی برای نمایش تعداد نمونه‌های درست و نادرست طبقه‌بندی شده برای هر کلاس تولید و با استفاده از seabornرسم می‌شود.

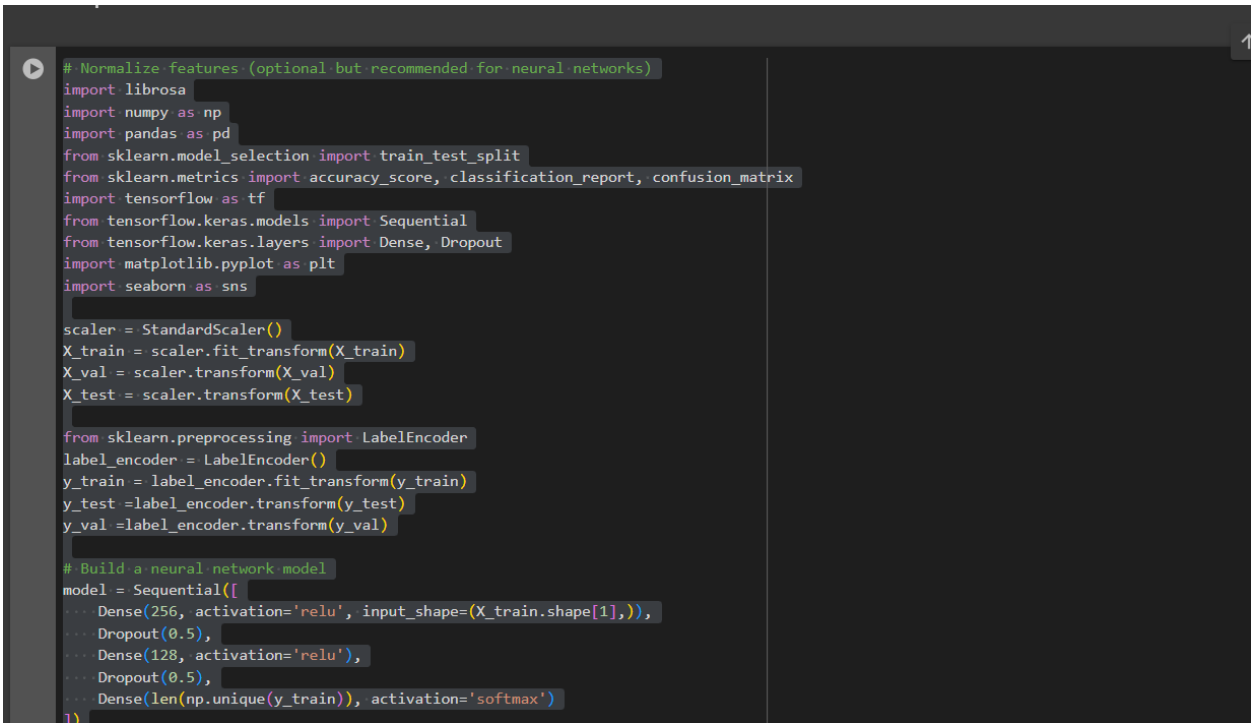
```python
# Evaluate test accuracy
test_accuracy = accuracy_score(y_test, y_pred_test)
print(f"Test Accuracy: {test_accuracy}")

# Generate classification report
print("Classification Report:")
print(classification_report(y_test, y_pred_test))

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred_test)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=svm_model.classes_, yticklabels=svm_model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

تصویر ۲۹ نمایش ماتریس درهمریختگی

برای مدل‌های کلاسیک دیگر نیز همین فرایند کد تکرار می‌شود.

مدل عمیق:

برای عمیق ابتدا، یک شبکه‌ی عصبی ساده چند لایه می‌سازیم و سپس مانند مدل‌های بالا آن را آموزش داده و عملکرد آن را روی داده‌های مختلف بررسی می‌کنیم.



```python
# Normalize features (optional but recommended for neural networks)
import librosa
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import seaborn as sns

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test =label_encoder.transform(y_test)
y_val =label_encoder.transform(y_val)

# Build a neural network model
model = Sequential([
    Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(np.unique(y_train)), activation='softmax')
])
```

تصویر ۳۰ مدل MLP

حالت oversample کردن

```python
# Oversample the training dataset only
ros = RandomOverSampler()

train_features, train_labels = train_df.drop(columns=['emotion']), train_df['emotion']
train_features_resampled, train_labels_resampled = ros.fit_resample(train_features, train_labels)
train_df = pd.concat([train_features_resampled, train_labels_resampled], axis=1)
```

تصویر ۳۱ oversample کردن داده‌ها

تفاوت این کد فقط در بخش oversample کردن داده‌های آموزش هست که با کد بالا انجام شده است.

آموزش سه کلاسه

```
7
8  # Calculate the number of samples for each emotion class
9  emotion_counts = df['emotion'].value_counts()
10
11 # Identify the three classes with the least number of samples
12 least_classes = emotion_counts.nsmallest(3).index
13
14 # Filter out these classes from the dataset
15 df_filtered = df[~df['emotion'].isin(least_classes)]
16
```

تصویر ۳۲ حذف کلاس‌های اقلیت

کلاسهایی که کمترین تعداد داده را داشتند حذف کردیم.

Undersample کردن

داده‌های ترین را undersample می‌کنیم:

```
# Undersample the training dataset only
rus = RandomUnderSampler()

train_features, train_labels = train_df.drop(columns=['emotion']), train_df['emotion']
train_features_resampled, train_labels_resampled = rus.fit_resample(train_features, train_labels)
train_df = pd.concat([train_features_resampled, train_labels_resampled], axis=1)

# Get unique emotions
```

تصویر ۳۳ under sample کردن داده‌ها

آموزش با دیگر مجموعه داده‌ها

فرمت هرکدام از این مجموعه داده‌ها متفاوت است بنابراین ابتدا هرکدام را باز می‌کنیم و با فرمت یکسان برای آموزش آماده می‌کنیم، دقت شود داده‌های تست، همچنان مجموعه داده اصلی هستند.