예측 모델

게임 플레이 예측

Easy and simple presentation

04

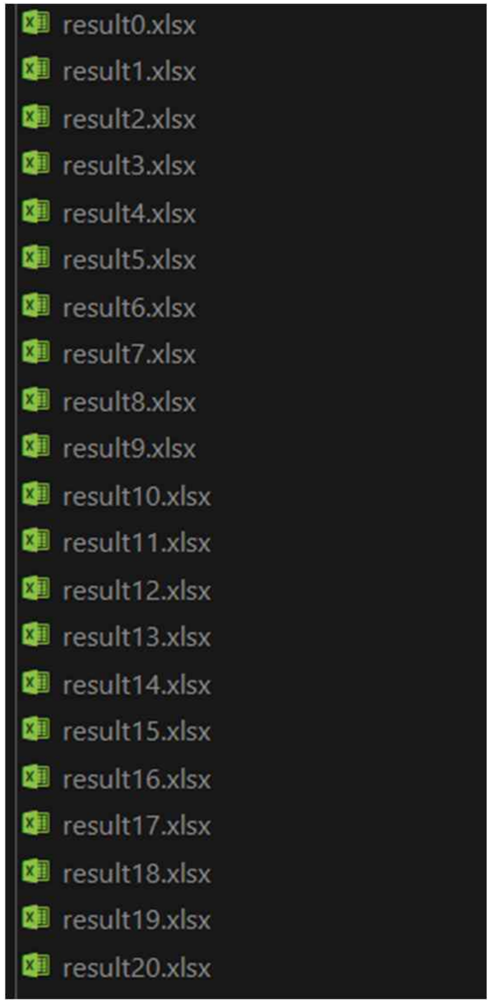
오버워치 댓글 예측기

김강환

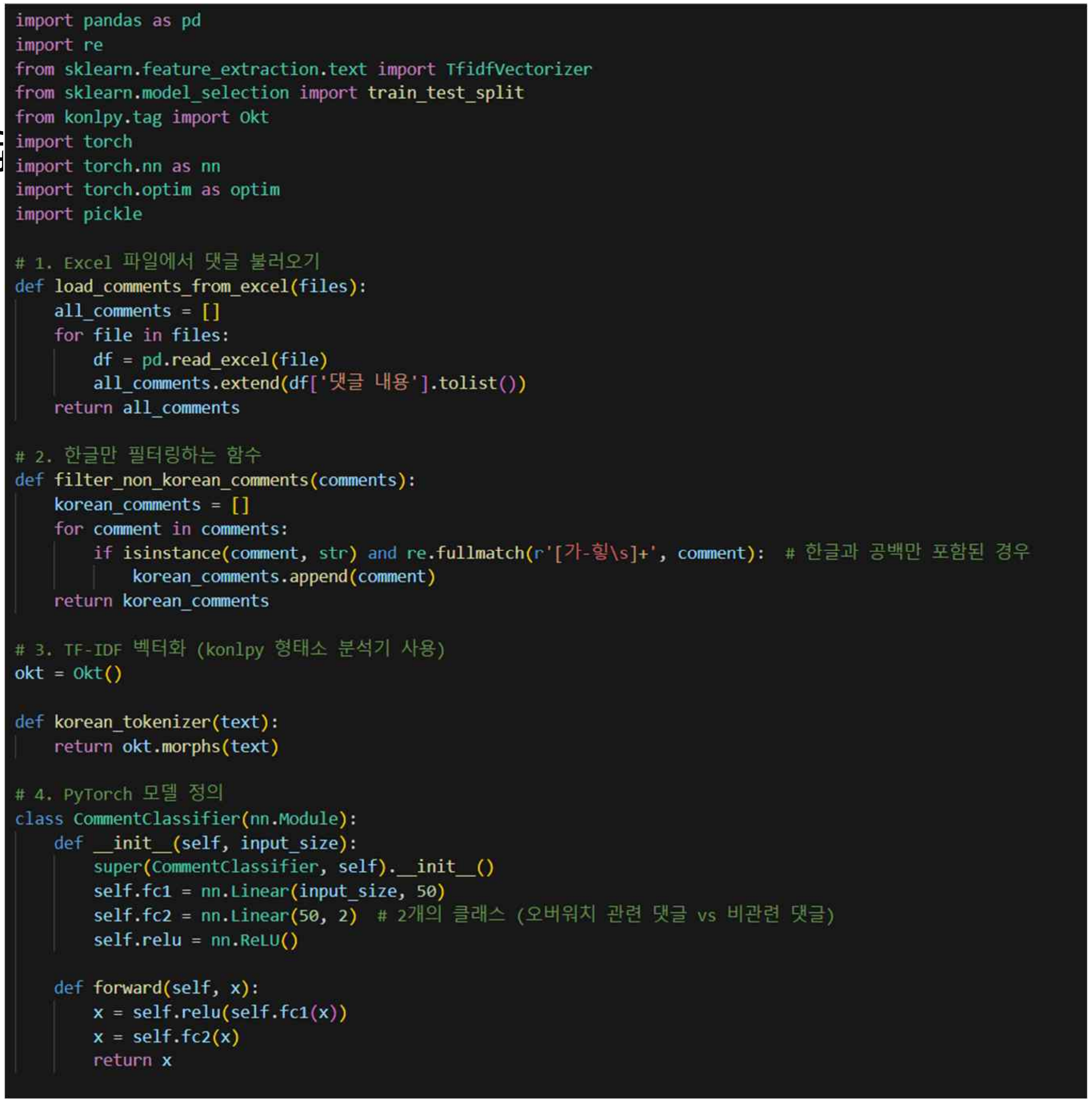
1

크롤링

셀레니움+ 뷰티풀숩

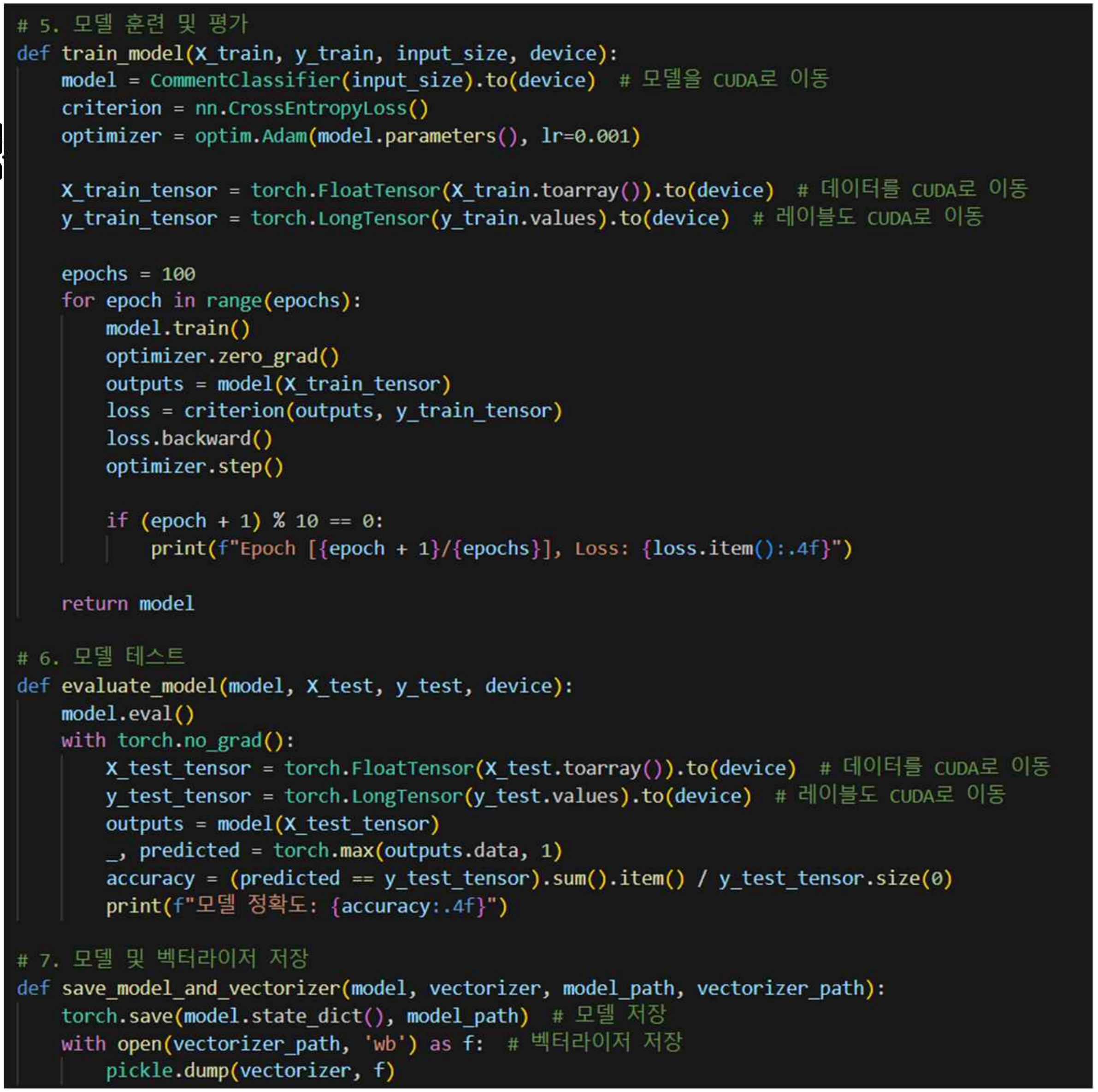

result0.xlsx
result1.xlsx
result2.xlsx
result3.xlsx
result4.xlsx
result5.xlsx
result6.xlsx
result7.xlsx
result8.xlsx
result9.xlsx
result10.xlsx
result11.xlsx
result12.xlsx
result13.xlsx
result14.xlsx
result15.xlsx
result16.xlsx
result17.xlsx
result18.xlsx
result19.xlsx
result20.xlsx

2 학습



```python
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from konlpy.tag import Okt
import torch
import torch.nn as nn
import torch.optim as optim
import pickle

# 1. Excel 파일에서 댓글 불러오기
def load_comments_from_excel(files):
    all_comments = []
    for file in files:
        df = pd.read_excel(file)
        all_comments.extend(df['댓글 내용'].tolist())
    return all_comments

# 2. 한글만 필터링하는 함수
def filter_non_korean_comments(comments):
    korean_comments = []
    for comment in comments:
        if isinstance(comment, str) and re.fullmatch(r'[가-힣\s]+', comment):  # 한글과 공백만 포함된 경우
            korean_comments.append(comment)
    return korean_comments

# 3. TF-IDF 벡터화 (konlpy 형태소 분석기 사용)
okt = Okt()

def korean_tokenizer(text):
    return okt.morphs(text)

# 4. PyTorch 모델 정의
class CommentClassifier(nn.Module):
    def __init__(self, input_size):
        super(CommentClassifier, self).__init__()
        self.fc1 = nn.Linear(input_size, 50)
        self.fc2 = nn.Linear(50, 2)  # 2개의 클래스 (오버워치 관련 댓글 vs 비관련 댓글)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

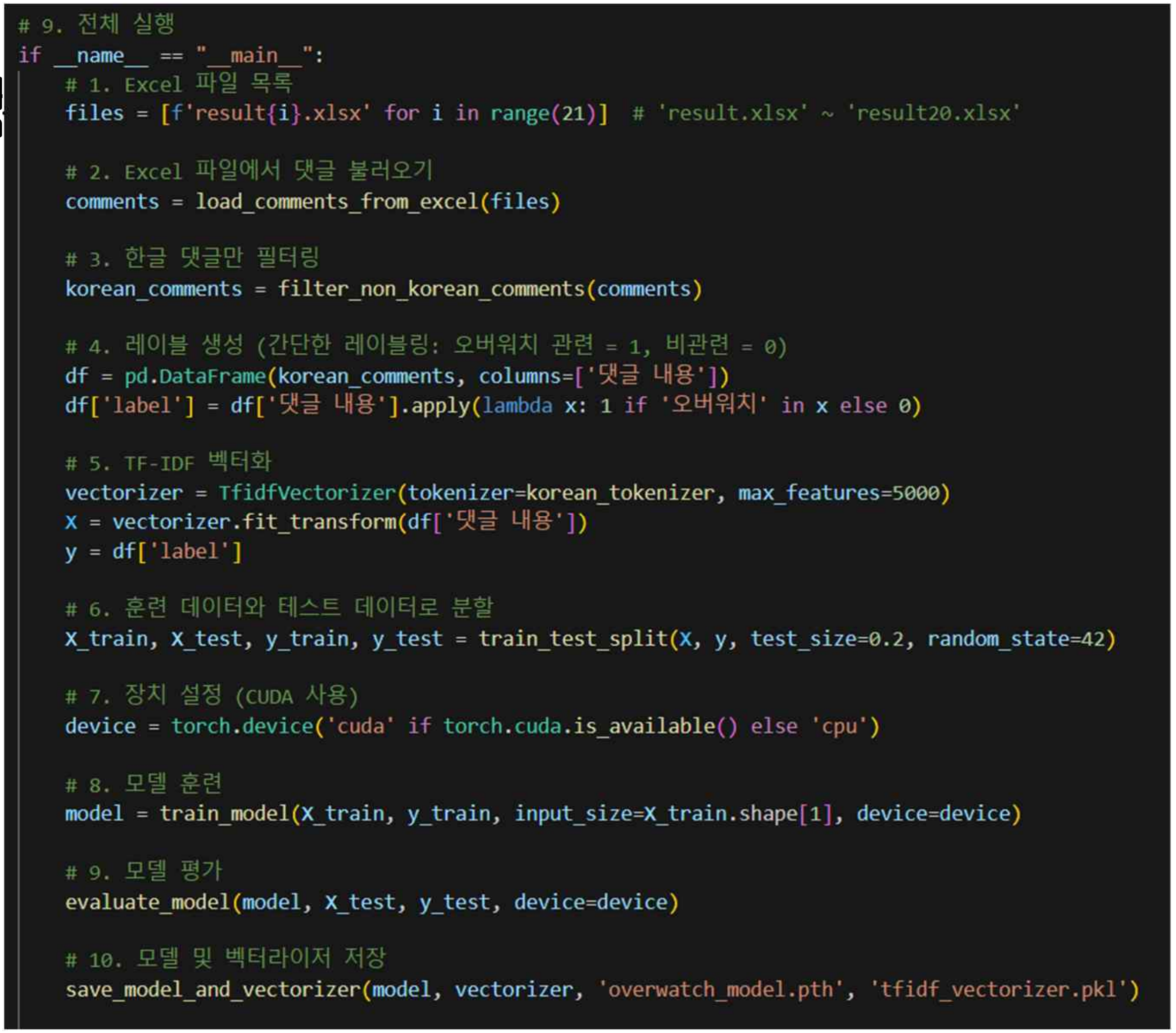
2



```python
# 5. 모델 훈련 및 평가
def train_model(X_train, y_train, input_size, device):
    model = CommentClassifier(input_size).to(device)  # 모델을 CUDA로 이동
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    X_train_tensor = torch.FloatTensor(X_train.toarray()).to(device)  # 데이터를 CUDA로 이동
    y_train_tensor = torch.LongTensor(y_train.values).to(device)  # 레이블도 CUDA로 이동

    epochs = 100
    for epoch in range(epochs):
        model.train()
        optimizer.zero_grad()
        outputs = model(X_train_tensor)
        loss = criterion(outputs, y_train_tensor)
        loss.backward()
        optimizer.step()

        if (epoch + 1) % 10 == 0:
            print(f"Epoch [{epoch + 1}/{epochs}], Loss: {loss.item():.4f}")

    return model

# 6. 모델 테스트
def evaluate_model(model, X_test, y_test, device):
    model.eval()
    with torch.no_grad():
        X_test_tensor = torch.FloatTensor(X_test.toarray()).to(device)  # 데이터를 CUDA로 이동
        y_test_tensor = torch.LongTensor(y_test.values).to(device)  # 레이블도 CUDA로 이동
        outputs = model(X_test_tensor)
        _, predicted = torch.max(outputs.data, 1)
        accuracy = (predicted == y_test_tensor).sum().item() / y_test_tensor.size(0)
        print(f"모델 정확도: {accuracy:.4f}")

# 7. 모델 및 벡터라이저 저장
def save_model_and_vectorizer(model, vectorizer, model_path, vectorizer_path):
    torch.save(model.state_dict(), model_path)  # 모델 저장
    with open(vectorizer_path, 'wb') as f:  # 벡터라이저 저장
        pickle.dump(vectorizer, f)
```

2

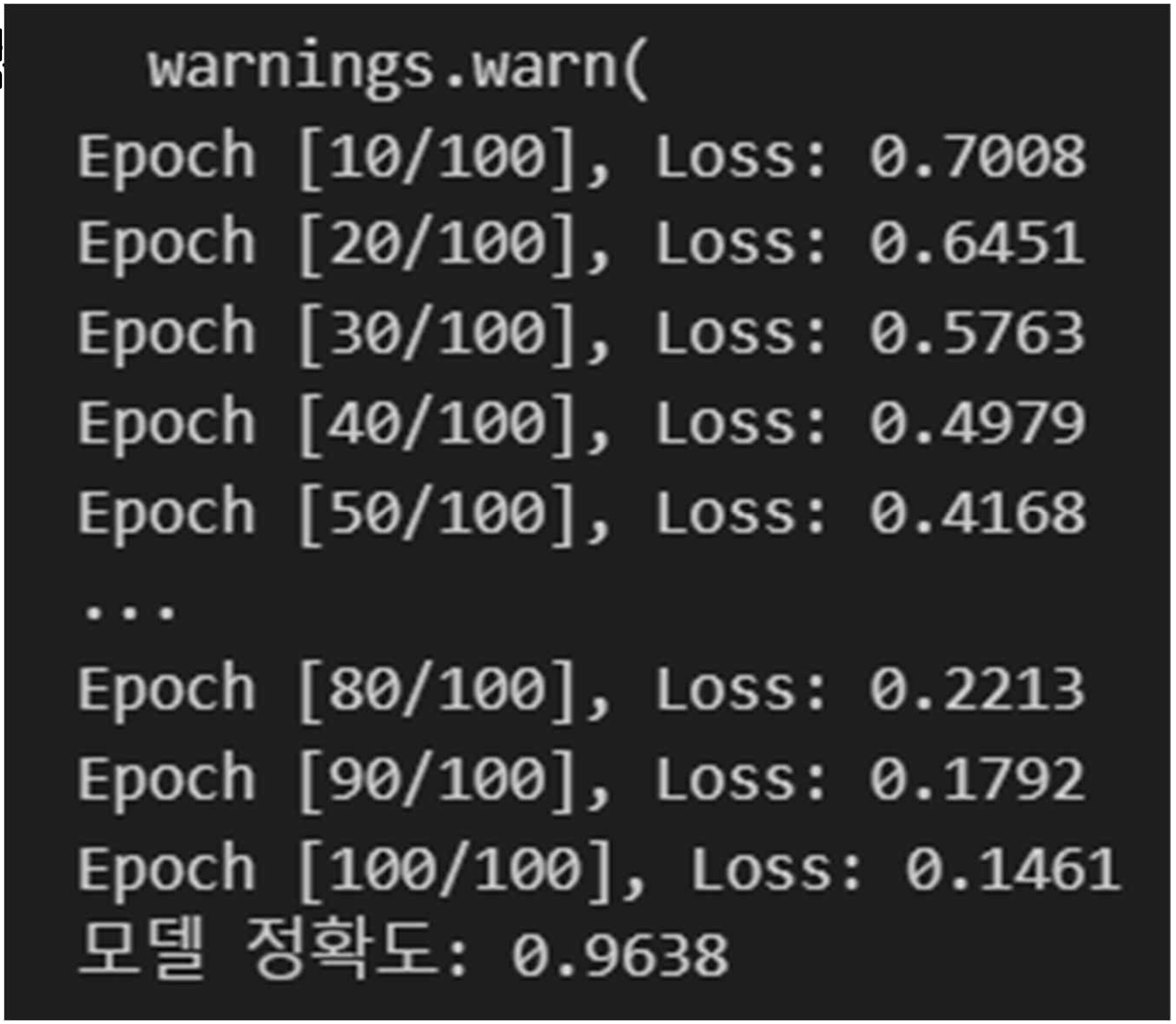
학



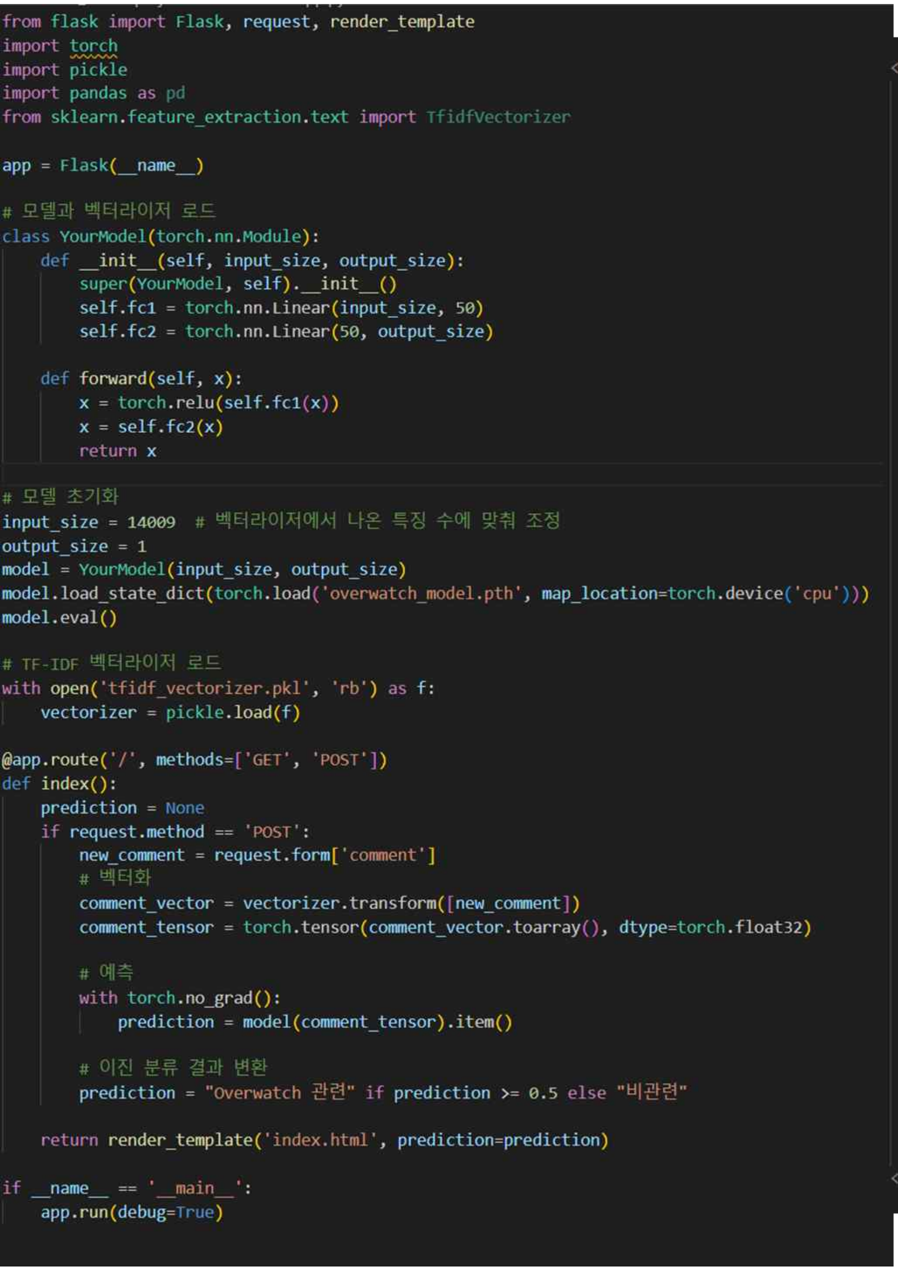
```python
# 9. 전체 실행
if __name__ == "__main__":
    # 1. Excel 파일 목록
    files = [f'result{i}.xlsx' for i in range(21)]  # 'result.xlsx' ~ 'result20.xlsx'

    # 2. Excel 파일에서 댓글 불러오기
    comments = load_comments_from_excel(files)

    # 3. 한글 댓글만 필터링
    korean_comments = filter_non_korean_comments(comments)

    # 4. 레이블 생성 (간단한 레이블링: 오버워치 관련 = 1, 비관련 = 0)
    df = pd.DataFrame(korean_comments, columns=['댓글 내용'])
    df['label'] = df['댓글 내용'].apply(lambda x: 1 if '오버워치' in x else 0)

    # 5. TF-IDF 벡터화
    vectorizer = TfidfVectorizer(tokenizer=korean_tokenizer, max_features=5000)
    X = vectorizer.fit_transform(df['댓글 내용'])
    y = df['label']

    # 6. 훈련 데이터와 테스트 데이터로 분할
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # 7. 장치 설정 (CUDA 사용)
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # 8. 모델 훈련
    model = train_model(X_train, y_train, input_size=X_train.shape[1], device=device)

    # 9. 모델 평가
    evaluate_model(model, X_test, y_test, device=device)

    # 10. 모델 및 벡터라이저 저장
    save_model_and_vectorizer(model, vectorizer, 'overwatch_model.pth', 'tfidf_vectorizer.pkl')
```

2

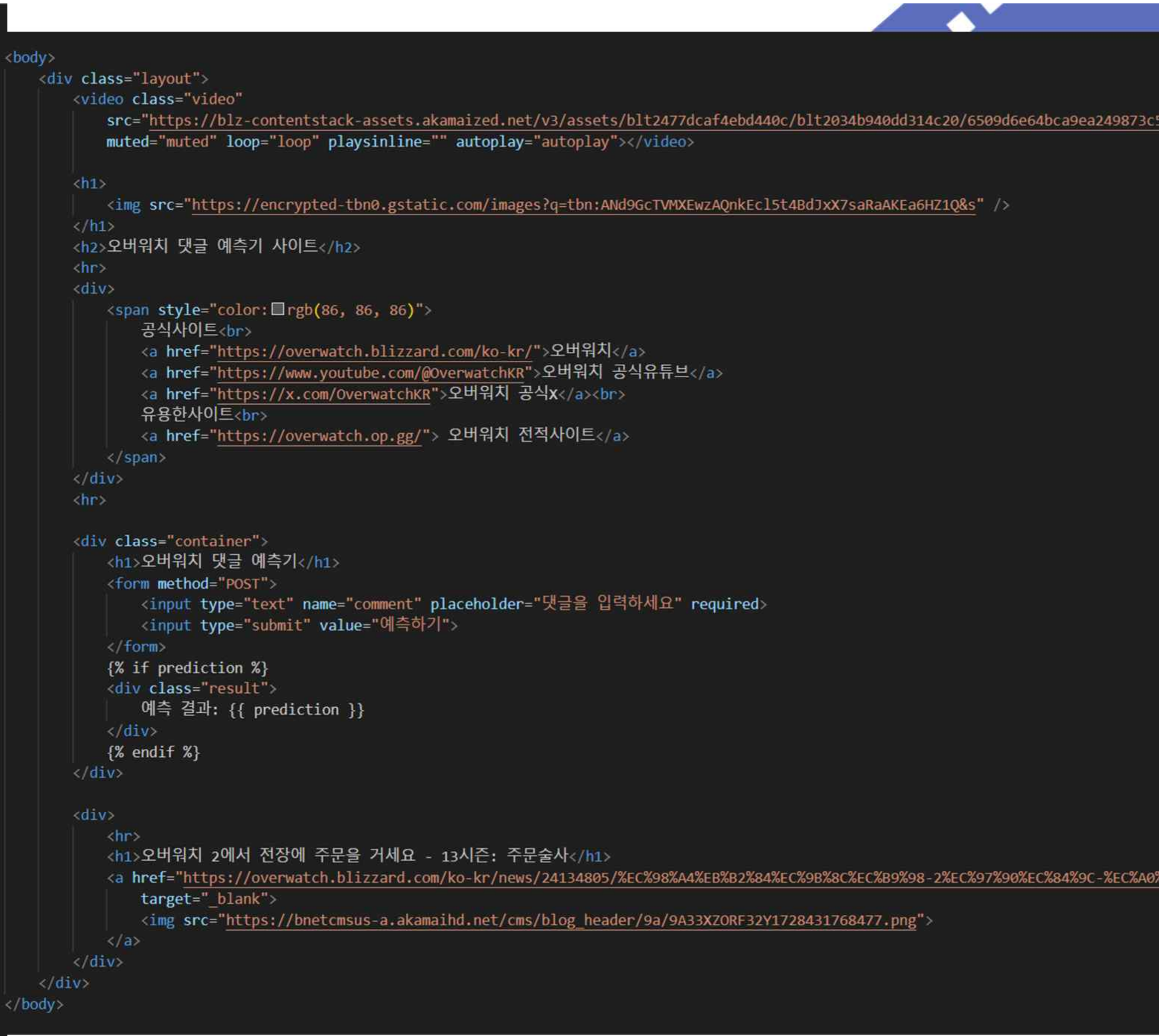
학



```
  warnings.warn(
Epoch [10/100], Loss: 0.7008
Epoch [20/100], Loss: 0.6451
Epoch [30/100], Loss: 0.5763
Epoch [40/100], Loss: 0.4979
Epoch [50/100], Loss: 0.4168
...
Epoch [80/100], Loss: 0.2213
Epoch [90/100], Loss: 0.1792
Epoch [100/100], Loss: 0.1461
모델 정확도: 0.9638
```



```python
from flask import Flask, request, render_template
import torch
import pickle
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

app = Flask(__name__)

# 모델과 벡터라이저 로드
class YourModel(torch.nn.Module):
    def __init__(self, input_size, output_size):
        super(YourModel, self).__init__()
        self.fc1 = torch.nn.Linear(input_size, 50)
        self.fc2 = torch.nn.Linear(50, output_size)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# 모델 초기화
input_size = 14009  # 벡터라이저에서 나온 특징 수에 맞춰 조정
output_size = 1
model = YourModel(input_size, output_size)
model.load_state_dict(torch.load('overwatch_model.pth', map_location=torch.device('cpu')))
model.eval()

# TF-IDF 벡터라이저 로드
with open('tfidf_vectorizer.pkl', 'rb') as f:
    vectorizer = pickle.load(f)

@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    if request.method == 'POST':
        new_comment = request.form['comment']
        # 벡터화
        comment_vector = vectorizer.transform([new_comment])
        comment_tensor = torch.tensor(comment_vector.toarray(), dtype=torch.float32)

        # 예측
        with torch.no_grad():
            prediction = model(comment_tensor).item()

        # 이진 분류 결과 변환
        prediction = "Overwatch 관련" if prediction >= 0.5 else "비관련"

    return render_template('index.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```



```html
<body>
    <div class="layout">
        <video class="video"
            src="https://blz-contentstack-assets.akamaized.net/v3/assets/blt2477dcaf4ebd440c/blt2034b940dd314c20/6509d6e64bca9ea249873c5
            muted="muted" loop="loop" playsinline="" autoplay="autoplay"></video>

        <h1>
            <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTVMXEwzAQnkEcl5t4BdJxX7saRaAKEa6HZ1Q&s" />
        </h1>
        <h2>오버워치 댓글 예측기 사이트</h2>
        <hr>
        <div>
            <span style="color: rgb(86, 86, 86)">
                공식사이트<br>
                <a href="https://overwatch.blizzard.com/ko-kr/">오버워치</a>
                <a href="https://www.youtube.com/@OverwatchKR">오버워치 공식유튜브</a>
                <a href="https://x.com/OverwatchKR">오버워치 공식x</a><br>
                유용한사이트<br>
                <a href="https://overwatch.op.gg/"> 오버워치 전적사이트</a>
            </span>
        </div>
        <hr>

        <div class="container">
            <h1>오버워치 댓글 예측기</h1>
            <form method="POST">
                <input type="text" name="comment" placeholder="댓글을 입력하세요" required>
                <input type="submit" value="예측하기">
            </form>
            {% if prediction %}
            <div class="result">
                예측 결과: {{ prediction }}
            </div>
            {% endif %}
        </div>

        <div>
            <hr>
            <h1>오버워치 2에서 전장에 주문을 거세요 - 13시즌: 주문술사</h1>
            <a href="https://overwatch.blizzard.com/ko-kr/news/24134805/%EC%98%A4%EB%B2%84%EC%9B%8C%EC%B9%98-2%EC%97%90%EC%84%9C-%EC%A0%
                target="_blank">
                <img src="https://bnetcmsus-a.akamaihd.net/cms/blog_header/9a/9A33XZORF32Y1728431768477.png">
            </a>
        </div>
    </div>
</body>
```