

개요 및 설치 (Overview & Install)

Scikit-LLM은 대형 언어 모델(LLM)을 scikit-learn의 표준 **Estimator** 및 **Transformer** 인터페이스(**fit**, **predict**, **fit_transform**)로 감싸 파이프라인 및 교차 검증에 바로 결합할 수 있도록 지원하는 라이브러리입니다.

- 설치: `pip install scikit-llm`
- 핵심 개념: **fit()**은 주로 후보 레이블이나 예시를 등록하고, 실제 추론 및 토큰 소비는 **predict()** 실행 시점에 API 호출로 발생합니다.

전역 설정 (SKLLMConfig)

자격 증명 및 API 키를 전역으로 1회 설정하여 모든 추론기에서 공유합니다.

```
from skllm.config import SKLLMConfig

# OpenAI 키 및 조직 설정
SKLLMConfig.set_openai_key("<YOUR_KEY>")
SKLLMConfig.set_openai_org("<YOUR_ORG_ID>")

# 로컬/커스텀 OpenAI 호환 엔드포인트 설정
SKLLMConfig.set_gpt_url("http://localhost:8000/v1")
```

내장 데모 데이터셋 (Datasets)

토큰을 사용하기 전 프롬프트와 파이프라인 동작을 검증할 수 있는 샘플 데이터셋입니다.

```
from skllm.datasets import (
    get_classification_dataset,

    get_multilabel_classification_dataset
)

# X: 리뷰 텍스트 리스트, y: 레이블
('positive', 'negative', 'neutral')
X, y = get_classification_dataset()
```

제로샷 분류 (Zero-Shot Classification)

ZeroShotGPTClassifier (단일 레이블)

학습 데이터 없이 후보 레이블 목록만으로 텍스트를 분류합니다.

```
from skllm.models.gpt.classification.zero_shot
```

```
import ZeroShotGPTClassifier
```

```
clf = ZeroShotGPTClassifier(model="gpt-4o")
clf.fit(None, ["positive", "negative", "neutral"])
labels = clf.predict(X)
# 결과: ['positive', 'neutral', 'negative']
```

MultiLabelZeroShotGPTClassifier (다중 레이블)

하나의 텍스트에 여러 레이블을 동시에 할당합니다.

```
from skllm.models.gpt.classification.zero_shot
import (
    MultiLabelZeroShotGPTClassifier
)
```

```
clf = MultiLabelZeroShotGPTClassifier(max_labels=3)
clf.fit(None, [{"Quality", "Price", "Delivery", "Service"}])
labels = clf.predict(X)
```

퓨샷 분류 (Few-Shot Classification)

FewShotGPTClassifier (정적 퓨샷)

학습 데이터 전체를 프롬프트 내 예시로 포함하여 분류합니다.

```
from skllm.models.gpt.classification.few_shot
import FewShotGPTClassifier
```

```
clf = FewShotGPTClassifier(model="gpt-4o")
clf.fit(X_train, y_train) # 클래스당 약 10개 예시 권장
labels = clf.predict(X_test)
```

DynamicFewShotGPTClassifier (동적 시맨틱 검색)

입력 샘플과 가장 유사한 예시를 클래스별로 N 개씩 동적으로 검색하여 프롬프트에 주입함으로써 컨텍스트 윈도우 한계를 극복합니다.

```
from skllm.models.gpt.classification.few_shot
import DynamicFewShotGPTClassifier
```

```
clf = DynamicFewShotGPTClassifier(n_examples=3)
clf.fit(X_train, y_train)
labels = clf.predict(X_test)
```

추론 과정 설명 (CoT Classifier)

CoTGPTClassifier (Chain-of-Thought)

분류 레이블과 함께 모델의 추론 근거(Explanation)를 동시에 반환합니다.

```
from skllm.models.gpt.classification.zero_shot
import CoTGPTClassifier
```

```
clf = CoTGPTClassifier(model="gpt-4o").fit(X, y)
pred = clf.predict(X) # 2차원 배열 반환
```

```
labels = pred[:, 0] # 예측 레이블
reasoning = pred[:, 1] # 의사결정 추론 근거
```

파인튜닝 (Tunable GPTClassifier)

클라우드 제공자 API를 통해 모델을 직접 지도학습 미세조정합니다.

```
from skllm.models.gpt.classification.tunable
import GPTClassifier
```

```
clf = GPTClassifier(base_model="gpt-3.5-turbo-0613", n_epochs=1)
clf.fit(X_train, y_train)
```

텍스트 임베딩 및 벡터화 (GPTVectorizer)

텍스트를 고정 차원 밀집 벡터로 변환하는 scikit-learn 표준 **Transformer**입니다.

```
from skllm.models.gpt.vectorization
import GPTVectorizer
```

```
vec = GPTVectorizer(model="text-embedding-3-small", batch_size=8)
X_emb = vec.fit_transform(X)
print(X_emb.shape) # (2000, 1536)
```

Scikit-Learn 파이프라인 결합 (Pipeline)

GPTVectorizer를 전처리 단계로 두고 전통적인 머신러닝 분류기와 파이프라인을 구성합니다.

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
```

```
pipe = Pipeline([
    ("emb", GPTVectorizer(model="text-embedding-3-small")),
    ("clf", LogisticRegression(max_iter=1000)),
])
```

```
pipe.fit(X_train, y_train)
predictions = pipe.predict(X_test)
```

텍스트 변환기 (Text2Text Transformers)

GPTSummarizer (문서 요약)

```
from skllm.models.gpt.text2text.summarization
import GPTSummarizer
```

```
summ = GPTSummarizer(model="gpt-4o", max_words=15, focus="pricing")
X_short = summ.fit_transform(X)
```

GPTTranslator (다국어 번역)

```
from skllm.models.gpt.text2text.translation
import GPTTranslator
```

```
t = GPTTranslator(model="gpt-4o", output_language="Korean")
X_ko = t.fit_transform(X)
```

GPTEexplainableNER (개체명 인식 및 설명)

사용자가 직접 정의한 개체명(Entity)을 추출하고 선택 사유를 함께 출력합니다.

```
from skllm.models.gpt.tagging.ner import
GPTEexplainableNER
```

```
entities = {
    "PERSON": "개인 이름 또는 사람",
    "ORG": "회사, 기관 또는 조직 이름",
}
ner =
GPTEexplainableNER(entities=entities)
tagged = ner.fit_transform(X)
```

백엔드 네임스페이스 (Backend Namespaces)

model 인자에 프리픽스를 지정하여 Azure, 로컬 GGUF, 커스텀 엔드포인트를 지정할 수 있습니다.

```
# Azure OpenAI 배포 모델
ZeroShotGPPTClassifier(model="azure::my-
deployment")
```

```
# 로컬 양자화 GGUF 모델
ZeroShotGPPTClassifier(model="gguf::llama3-8b-
q4")
```

```
# 커스텀 OpenAI 호환 URL
ZeroShotGPPTClassifier(model="custom_url::my-
model")
```

모델 평가 및 검증 (Evaluation)

scikit-learn의 표준 평가지표 및 교차 검증 도구를 변환 없이 바로 사용합니다.

```
from sklearn.metrics import
classification_report
from sklearn.model_selection import
cross_val_score
```

```
# 분류 성능 지표 보고서
print(classification_report(y_test,
labels))
```

```
# 교차 검증 (주의: fold 수에 비례하여 API 호출 증가)
scores = cross_val_score(clf, X, y,
cv=3)
```