

Scikit-LLM Cheat Sheet

Overview & Installation

Scikit-LLM wraps Large Language Models into standard scikit-learn `Estimator` and `Transformer` interfaces (`fit`, `predict`, `fit_transform`), allowing direct use in scikit-learn `Pipeline` and cross-validation workflows.

- **Install:** `pip install scikit-llm`
- **Execution Model:** `fit()` registers candidate labels or prompt examples; actual inference and token consumption occur during `predict()`.

Global Configuration (SKLLM-Config)

Set credentials once globally for all estimators in the session.

```
from skllm.config import SKLLMConfig
```

```
# OpenAI API Key & Organization
SKLLMConfig.set_openai_key("<YOUR_KEY>")
SKLLMConfig.set_openai_org("<YOUR_ORG_ID>")
```

```
# Custom / Local OpenAI-compatible endpoint
SKLLMConfig.set_gpt_url("http://localhost:8000/v1")
```

Demo Datasets

Small sample datasets to verify prompts before expending tokens.

```
from skllm.datasets import (
    get_classification_dataset,
```

```
get_multilabel_classification_dataset
)
```

```
# labels: positive, negative, neutral
X, y = get_classification_dataset()
```

Zero-Shot Classification

ZeroShotGPTClassifier (Single-Label)

Classify text with candidate labels without training data.

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

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

MultiLabelZeroShotGPTClassifier (Multi-Label)

Assign multiple labels per text sample with a maximum limit.

```
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)
```

Few-Shot Classification

FewShotGPTClassifier (Static In-Context)

Includes all training samples directly in the prompt context.

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

```
clf =
FewShotGPTClassifier(model="gpt-4o")
clf.fit(X_train, y_train) # Keep ~10
samples per class
labels = clf.predict(X_test)
```

DynamicFewShotGPTClassifier (Semantic Retrieval)

Retrieves the most semantically relevant examples per class for each sample to avoid context-window limits.

```
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)
```

Chain-of-Thought Classifier

CoTGPTClassifier (Label + Explanation)

Requests reasoning alongside classification labels.

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

clf =
CoTGPTClassifier(model="gpt-4o").fit(X,
y)
pred = clf.predict(X)
```

```
labels = pred[:, 0] # Predicted
class
reasoning = pred[:, 1] # Explanatory
text
```

Fine-Tuning (Tunable GPTClassifier)

Fine-tunes the base model in the cloud provider's infrastructure.

```
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)
```

Vectorization & Embeddings (GPTVectorizer)

Embeds arbitrary-length text into fixed-dimension vectors as a standard scikit-learn transformer.

```
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 Pipeline Integration

Combine LLM embeddings with downstream traditional estimators.

```
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)
```

Text2Text Transformers

GPTSummarizer (Document Summarization)

```
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 (Translation Transformer)

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

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

GPTExplainableNER (Entity Extraction)

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

```
entities = {
    "PERSON": "A name of an
individual.",
    "ORG": "A name of a company or
organization.",
}
ner =
GPTExplainableNER(entities=entities)
tagged = ner.fit_transform(X)
```

Backend Namespaces

Prefixing `model` swaps between cloud and local providers.

```
# Azure deployment
ZeroShotGPTClassifier(model="azure::my-
deployment")
```

```
# Local quantized GGUF
ZeroShotGPTClassifier(model="gguf::llama3-8b-
q4")
```

```
# Custom OpenAI-compatible URL
ZeroShotGPTClassifier(model="custom_url::my-
model")
```

Model Evaluation

Use scikit-learn metrics and cross-validation natively.

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

```
print(classification_report(y_test,
labels))
```

```
# Re-queries API for each fold
scores = cross_val_score(clf, X, y,
cv=3)
```