Python Transformers, Tokenization, XGBoost with Optuna Fine-Tuning

1. Transformers and Tokenization

### What are Transformers?

Transformers are deep learning models used for NLP tasks like sentiment analysis, text classification, and text generation.

They use self-attention mechanisms to process text data efficiently.

### Tokenization Example:

```
# Importing necessary libraries
from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline
import torch

# Load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")

# Tokenize a sentence
sentence = "I love Python programming!"
tokens = tokenizer(sentence)
print("Tokens:", tokens)
```

2. XGBoost with Optuna Fine-Tuning

### XGBoost with Optuna for Hyperparameter Tuning

```
import xgboost as xgb

from sklearn.datasets import load_diabetes

from sklearn.model_selection import train_test_split

import optuna


# Load dataset

data = load_diabetes()

X_train,  X_test,  y_train,  y_test  =  train_test_split(data.data,  data.target,

test_size=0.2, random_state=42)


# Objective function for Optuna

def objective(trial):

    params = {

        "objective": "reg:squarederror",

        "eval_metric": "rmse",

        "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.3),

        "max_depth": trial.suggest_int("max_depth", 3, 10),

        "n_estimators": trial.suggest_int("n_estimators", 50, 300),

        "subsample": trial.suggest_float("subsample", 0.5, 1.0),

        "colsample_bytree": trial.suggest_float("colsample_bytree", 0.5, 1.0)

    }


    dtrain = xgb.DMatrix(X_train, label=y_train)

    dtest = xgb.DMatrix(X_test, label=y_test)
```

```python
    model = xgb.train(params, dtrain, num_boost_round=100)

    preds = model.predict(dtest)


    rmse = ((preds - y_test) ** 2).mean() ** 0.5

    return rmse


# Optuna tuning

study = optuna.create_study(direction="minimize")

study.optimize(objective, n_trials=20)


# Best parameters

print("Best parameters:", study.best_params)
```