Project Idea: Enhancing Access to Quality Education (SDG 4)

Project Explanation

The aim of this project is to enhance access to quality education by fine-tuning a pre-trained large language model (LLM) to better support educational tasks such as automated essay scoring, personalized feedback, and content recommendation. By improving the model's performance in these areas, we can help educators and students achieve better educational outcomes, particularly in under-resourced areas. This aligns with Sustainable Development Goal 4 (SDG 4), which aims to ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

Steps to Complete the Project

i. Install the Necessary Libraries and Tools

First, we need to install the necessary libraries for finetuning large language models. This includes the Hugging Face Transformers library and other dependencies.

```
python
!pip install transformers datasets matplotlib seaborn
```

ii. Exploratory Data Analysis (EDA)

Load and analyze the dataset to understand its structure and characteristics.

```
python
from datasets import load_dataset
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset (for example, an essay scoring dataset)
dataset = load_dataset('path/to/dataset')

# Display basic information about the dataset
print(dataset)
```

```
# Visualize the dataset (e.g., distribution of scores)
sns.histplot(dataset['train']['score'])
plt.title('Distribution of Essay Scores')
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.show()
```

iii. Dataset Preparation

Preprocess the dataset to make it suitable for the specific task of essay scoring.

python

```
from transformers import AutoTokenizer

# Load the tokenizer
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')

# Tokenize the essays
def tokenize_function(examples):
    return tokenizer(examples['essay'], truncation=True, padding='max_length')

tokenized_dataset = dataset.map(tokenize_function, batched=True)
```

iv. Model Selection

Choose and load a pre-trained language model.

python

```
from transformers import AutoModelForSequenceClassification

# Load a pre-trained model
```

```
model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=1)
```

v. Finetuning Process

Define the training arguments and fine-tune the pre-trained model on the prepared dataset.

```
python
from transformers import TrainingArguments, Trainer

training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset['train'],
    eval_dataset=tokenized_dataset['test'],
)

trainer.train()
```

vi. Evaluation

Evaluate the fine-tuned model on a test set and compare its performance before and after the fine-tuning process.

python

```
from sklearn.metrics import mean_squared_error

# Evaluate the model
predictions = trainer.predict(tokenized_dataset['test'])
predicted_scores = predictions.predictions.flatten()
true_scores = tokenized_dataset['test']['score']

# Calculate and print the mean squared error
mse = mean_squared_error(true_scores, predicted_scores)
print(f'Mean Squared Error: {mse}')

# Compare performance before and after fine-tuning (for example, using validation set)
# (Assuming you have initial model predictions before fine-tuning)
initial_predictions = ...  # Load or compute initial predictions
initial_mse = mean_squared_error(true_scores, initial_predictions)
print(f'Initial Mean Squared Error: {initial_mse}')
print(f'Improvement: {initial_mse - mse}')
```

Summary

By implementing these steps, our goal is to optimize a large language model to better assist with educational tasks, which will contribute to SDG 4. The project's success will be assessed based on the improvement in model performance, measured using evaluation metrics such as mean squared error.