```
import pandas as pd
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
import torch
from tqdm import tqdm


# 1. Load the dataset
df = pd.read_csv('spam.csv')

# Sample a few rows for testing
df = df.sample(20, random_state=42).reset_index(drop=True)


# 2. Load the model and tokenizer
model_name = "mistralai/Mistral-7B-Instruct-v0.2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16, device_map="auto")
```

```
Loading checkpoint shards:   0%|          | 0/3 [00:00<?, ?it/s]
Some parameters are on the meta device because they were offloaded to the cpu and disk.
```

```
# 3. Build a classification pipeline
llm_pipeline = pipeline("text-generation", model=model, tokenizer=tokenizer, max_new_tokens=10)

# 4. Prompt engineering: design a clear and constraining prompt
def classify_email(email_text):
    prompt = f"""Classify the following email as either "spam" or "ham" (not spam).
Only respond with one word: spam or ham.

Email:
\"\"\"{email_text}\"\"\"
Answer:"""
    response = llm_pipeline(prompt)[0]['generated_text']
    # Extract the last word from the response
    answer = response.strip().split("Answer:")[-1].strip().lower()
    if "spam" in answer:
        return "spam"
    elif "ham" in answer:
        return "ham"
    else:
        return "unknown" # long response
```

```
Device set to use cpu
```

```
from tqdm import tqdm
tqdm.pandas()

# Initialize the new column with None or some default value
df['llm_pred'] = None

# Apply classify_email only on the first row (index 0)
df.loc[0, 'llm_pred'] = classify_email(df.loc[0, 'target'])
```

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
```

```
print("Prediction for first row:", df.loc[0, 'llm_pred'])
print("Actual for first row:", df.loc[0, 'target'])
```

```
Prediction for first row: ham
Actual for first row: ham
```

```
Commencez à coder ou à générer avec l'IA.
```