```
pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.1.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (17.0.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.6)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.9.0,>=2023.1.0 (from fsspec[http]<=2024.9.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.11.2)
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23.0->data
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (2024.8.3
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1.16
Downloading datasets-3.1.0-py3-none-any.whl (480 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 480.6/480.6 kB 15.8 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 12.2 MB/s eta 0:00:00
Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.3/179.3 kB 17.0 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 14.8 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 18.0 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.9.0 which is incompatible.
Successfully installed datasets-3.1.0 dill-0.3.8 fsspec-2024.9.0 multiprocess-0.70.16 xxhash-3.5.0
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from datasets import Dataset
from transformers import AlbertTokenizer, AlbertForSequenceClassification, Trainer, TrainingArguments
from sklearn.metrics import accuracy_score
import torch
from transformers import EarlyStoppingCallback

# Load the dataset
df = pd.read_csv("bbc_text_cls.csv")

# Ensure column names are consistent
df.columns = df.columns.str.strip().str.lower()

# Verify the column names
if 'text' not in df.columns or 'labels' not in df.columns:
    raise ValueError("The dataset must contain 'text' and 'labels' columns.")

# Step 1: Split into training and test sets (80-20 split) - test set is set aside for final evaluation
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['labels'])
```

```
# Step 2: Further split train_data into training and validation sets (80-20 split of training data)
train_df, val_df = train_test_split(train_df, test_size=0.2, random_state=42, stratify=train_df['labels'])

# Convert to Hugging Face Dataset
train_dataset = Dataset.from_pandas(train_df)
val_dataset = Dataset.from_pandas(val_df)
test_dataset = Dataset.from_pandas(test_df)

# Map label names to integers
label_mapping = {label: idx for idx, label in enumerate(df['labels'].unique())}
train_dataset = train_dataset.map(lambda x: {'labels': label_mapping[x['labels']]})
val_dataset = val_dataset.map(lambda x: {'labels': label_mapping[x['labels']]})
test_dataset = test_dataset.map(lambda x: {'labels': label_mapping[x['labels']]})

# Load the ALBERT tokenizer
tokenizer = AlbertTokenizer.from_pretrained("albert-base-v2")

# Preprocess the text data
def preprocess_function(examples):
    return tokenizer(examples['text'], padding="max_length", truncation=True, max_length=128)

# Tokenize the datasets
train_dataset = train_dataset.map(preprocess_function, batched=True)
val_dataset = val_dataset.map(preprocess_function, batched=True)
test_dataset = test_dataset.map(preprocess_function, batched=True)

# Set the format for PyTorch tensors
train_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
val_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
test_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])

# Load ALBERT model for sequence classification
model = AlbertForSequenceClassification.from_pretrained("albert-base-v2", num_labels=len(label_mapping))

# Freeze all layers except the classification head
for param in model.albert.parameters():
    param.requires_grad = False

# Ensure the function is defined
def check_frozen_layers(model):
    frozen_layers = []
    trainable_layers = []

    for name, param in model.named_parameters():
        if param.requires_grad:
            trainable_layers.append(name)
        else:
            frozen_layers.append(name)

    print("\nTrainable Layers (Unfrozen):")
    for layer in trainable_layers:
        print(f"- {layer}")

    print("\nFrozen Layers:")
    for layer in frozen_layers:
        print(f"- {layer}")

    # Confirmation message
    if all("classifier" in layer for layer in trainable_layers) and len(trainable_layers) == 2:
        print("\nConfirmation: Only the classification head is trainable!")
    else:
        print("\nWarning: Some unexpected layers are trainable!")

# Call this function to check frozen layers after model initialization
check_frozen_layers(model)


# Define training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",              # Evaluate on the validation set after each epoch
    save_strategy="epoch",
    learning_rate=5e-4,                       # Higher learning rate since only the classification head is trainable
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=50,                      # Train for more epochs
    logging_dir="./logs",
    logging_steps=50,
    save_total_limit=2,
    load_best_model_at_end=True               # Load the best model based on validation loss
)

# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)]  # Adjust early stopping patience
)

# Fine-tune the model
trainer.train()

# Evaluate the model on the validation set
val_results = trainer.evaluate()
print(f"Validation Results: {val_results}")

# Evaluate the model on the test set
test_predictions = trainer.predict(test_dataset)
predicted_labels = torch.argmax(torch.tensor(test_predictions.predictions), axis=1).numpy()
true_labels = test_predictions.label_ids

accuracy = accuracy_score(true_labels, predicted_labels)
print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Save the fine-tuned model
trainer.save_model("./fine_tuned_albert_bbc")
```

```
Map: 100%                                         1424/1424 [00:00<00:00, 15186.48 examples/s]
Map: 100%                                         356/356 [00:00<00:00, 7916.80 examples/s]
Map: 100%                                         445/445 [00:00<00:00, 7883.46 examples/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as se
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%                                25.0/25.0 [00:00<00:00, 1.79kB/s]
spiece.model: 100%                                     760k/760k [00:00<00:00, 3.96MB/s]
tokenizer.json: 100%                                   1.31M/1.31M [00:00<00:00, 6.69MB/s]
config.json: 100%                                    684/684 [00:00<00:00, 59.4kB/s]
Map: 100%                                         1424/1424 [00:05<00:00, 269.69 examples/s]
Map: 100%                                         356/356 [00:01<00:00, 268.28 examples/s]
Map: 100%                                         445/445 [00:01<00:00, 262.01 examples/s]
model.safetensors: 100%                                  47.4M/47.4M [00:00<00:00, 92.5MB/s]
Some weights of AlbertForSequenceClassification were not initialized from the model checkpoint at albert-base-v2 and are newly initia
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1568: FutureWarning: `evaluation_strategy` is deprecated and wi
  warnings.warn(

Trainable Layers (Unfrozen):
- classifier.weight
- classifier.bias

Frozen Layers:
- albert.embeddings.word_embeddings.weight
- albert.embeddings.position_embeddings.weight
- albert.embeddings.token_type_embeddings.weight
- albert.embeddings.LayerNorm.weight
- albert.embeddings.LayerNorm.bias
- albert.encoder.embedding_hidden_mapping_in.weight
- albert.encoder.embedding_hidden_mapping_in.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.full_layer_layer_norm.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.full_layer_layer_norm.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.query.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.query.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.key.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.key.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.value.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.value.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.dense.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.dense.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.LayerNorm.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.attention.LayerNorm.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.ffn.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.ffn.bias
- albert.encoder.albert_layer_groups.0.albert_layers.0.ffn_output.weight
- albert.encoder.albert_layer_groups.0.albert_layers.0.ffn_output.bias
- albert.pooler.weight
- albert.pooler.bias

Confirmation: Only the classification head is trainable!
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241120_202002-6599ifmb
Syncing run ./results to Weights & Biases (docs)
View project at https://wandb.ai/gopalramaiya94-university-of-houston/huggingface
View run at https://wandb.ai/gopalramaiya94-university-of-houston/huggingface/runs/6599ifmb
[4450/4450 03:09, Epoch 50/50]
```

Epoch	Training Loss	Validation Loss
1	1.540100	1.312427
2	1.264300	1.177279
3	1.122200	1.081533
4	1.016200	1.026681

5	1.031000	0.955037
6	0.951200	0.921532
7	0.912300	0.877709
8	0.862100	0.863705
9	0.817900	0.833133
10	0.791800	0.814544
11	0.789300	0.780272
12	0.738700	0.783698
13	0.738700	0.772521
14	0.725900	0.744116
15	0.700900	0.732275
16	0.704700	0.711969
17	0.680700	0.712144
18	0.666700	0.695764
19	0.689900	0.695035
20	0.655300	0.696136
21	0.686300	0.675784
22	0.641000	0.664260
23	0.663600	0.680874
24	0.634600	0.672876
25	0.624400	0.650458
26	0.605300	0.654670
27	0.622300	0.638607
28	0.630200	0.650738
29	0.607800	0.629308
30	0.626700	0.627555
31	0.566200	0.620789
32	0.614600	0.647604
33	0.596100	0.616612
34	0.577100	0.619082
35	0.579800	0.616939
36	0.569100	0.610567
37	0.595800	0.609997
38	0.571200	0.607330
39	0.562900	0.602881
40	0.536600	0.600636
41	0.565700	0.603056
42	0.574300	0.598347
43	0.589300	0.596923
44	0.578500	0.598486
45	0.581100	0.595183
46	0.582600	0.594929
47	0.564800	0.593465
48	0.572900	0.594372
49	0.556200	0.593552
50	0.569200	0.593443

```
Validation Results: {'eval_loss': 0.5934434533119202, 'eval_runtime': 0.6936, 'eval_samples_per_second': 513.286, 'eval_steps_per_sec
```

```
Test Accuracy: 81.80%
```