1. Load Dataset:

```
!pip install transformers torch scikit-learn pandas
!pip install transformers torch scikit-learn pandas peft
```

Show hidden output

```
import re
import os
import json
import pandas as pd
import matplotlib.pyplot as plt
import torch
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
from transformers import BertTokenizer, BertForSequenceClassification, Trai
from peft import get_peft_model, LoraConfig
```

(1) Covid Fake News Dataset:

```
# List of JSON files to process
json_files = [
    'Cleaned_Covid19_Train.json',
    'Cleaned_Covid19_Dev.json',
]
data_dict = {}
# Process each JSON file
for json_file in json_files:
    # Load the dataset
    with open(json_file, 'r') as file:
        data = json.load(file)

    # Prepare a list to hold the processed data
    jsonl_data = []

    # Extract and process each entry
    for entry in data:
        # Extract the id, tweet, and label
        tweet = entry['tweet']
        label = entry['label']

        # Tokenize the tweet
        tokens = re.findall(r'\b\w+\b', tweet)  # Keep only words and numbe
        reconstructed_tweet = ' '.join(tokens)

        # Prepare the JSONL entry with the required structure
        jsonl_entry = {
            "systemInstruction": {
                "role": "assistant",  # Example role, adjust as needed
                "parts": [
                    {
                        "text": "Classification the content is Fake, Real,
                    }
                ]
            },
            "contents": [
                {
                    "role": "user",
                    "parts": [
                        {
                            "text": f"TRANSCRIPT: \n{reconstructed_tweet}\r
                        }
                    ]
                },
                {
                    "role": "model",
                    "parts": [
                        {
                            "text": label  # The label indicating the model
                        }
                    ]
                }
            ]
        }
        jsonl_data.append(jsonl_entry)

    # Write the processed data to a JSONL file
     output_file = json_file.replace('.json', '.jsonl')  # Change the extens
     with open(output_file, 'w') as outfile:
         for entry in jsonl_data:
             json.dump(entry, outfile)
             outfile.write('\n')  # Write each entry on a new line

     print(f"Processed {json_file} and saved to {output_file}.")

     data_dict[json_file] = jsonl_data
# Access the data using the correct keys - the original filenames
covid_train_data = data_dict['Cleaned_Covid19_Train.json']  # Corrected key
covid_dev_data = data_dict['Cleaned_Covid19_Dev.json']  # Corrected key
# Print the first few entries for verification
print(f"First few entries from claims_test_data:\n{covid_train_data[:5]}")

```

```
Processed Cleaned_Covid19_Train.json and saved to Cleaned_Covid19_Train.jso
Processed Cleaned_Covid19_Dev.json and saved to Cleaned_Covid19_Dev.jsonl.
First few entries from claims_test_data:
[{'systemInstruction': {'role': 'assistant', 'parts': [{'text': 'Classifica
```

(2) Health Fact Dataset:

```python
import json
import re
import os

# List of JSON files to process
json_files = [
    'healthfact_traindata.json',
    'cleaned_healthfact_test.json',
    'cleaned_healthfact_dev.json'
]

data_dict = {}

# Process each JSON file
for json_file in json_files:
    # Prepare a list to hold the processed data
    jsonl_data = []

    # Load the dataset
    with open(json_file, 'r') as file:
        # Read each line as a separate JSON object
        for line in file:
            try:
                entry = json.loads(line)
                # Extract the claim_id, claim, explanation, and label
                claim = entry['claim']
                explanation = entry['explanation']
                label = entry['label']

                # Tokenize the claim
                tokens = re.findall(r'\b\w+\b', claim)  # Keep only words a
                reconstructed_claim = ' '.join(tokens)

                # Prepare the JSONL entry
                jsonl_entry = {
                    "claim": reconstructed_claim,
                    "explanation": explanation,
                    "label": label
                }
                jsonl_data.append(jsonl_entry)
            except json.JSONDecodeError as e:
                print(f"Error decoding JSON: {e}")

    # Use the correct key to store the data in the dictionary - keep the or
    data_dict[json_file] = jsonl_data

# Access the data using the correct keys - the original filenames
healthfact_train_data = data_dict['healthfact_traindata.json']  # Correctec
healthfact_test_data = data_dict['cleaned_healthfact_test.json']  # Correct
healthfact_dev_data = data_dict['cleaned_healthfact_dev.json']  # Correctec
# Print the first few entries for verification
print(f"First few entries from claims_test_data:\n{healthfact_train_data[:5
```

```
First few entries from claims_test_data:
[{'claim': 'The money the Clinton Foundation took from from foreign governm
```

(3) Scifact Dataset:

```python
# List of JSONL files to process
jsonl_files = [
    'dev_3class.jsonl',
    'train_3class.jsonl'
]
data_dict = {}
# Process each JSONL file
for jsonl_file in jsonl_files:
    # Prepare a list to hold the processed data
    processed_data = []

    # Load the dataset
    with open(jsonl_file, 'r') as file:
        for line in file:
            try:
                entry = json.loads(line)

                # Extract the claim_id, claim, explanation, and label
                claim = entry['claim']
                explanation = entry['evidence_text']
                label = entry['label']

                # Tokenize the claim
                tokens = re.findall(r'\b\w+\b', claim)  # Keep only words a
                reconstructed_claim = ' '.join(tokens)

                # Prepare the JSONL entry
                jsonl_entry = {
                    "claim": reconstructed_claim,
                    "explanation": explanation,
                    "label": label
                }

                # Append the modified entry to the processed data list
                processed_data.append(jsonl_entry) # Changed from entry to
            except json.JSONDecodeError as e:
                print(f"Error decoding JSON: {e}")
    data_dict[jsonl_file] = processed_data

scifact_train_data = data_dict['train_3class.jsonl']  # Corrected key
scifact_test_data = data_dict['dev_3class.jsonl']  # Corrected key
print(f"First few entries from claims_test_data:\n{scifact_train_data[:5]}'
```

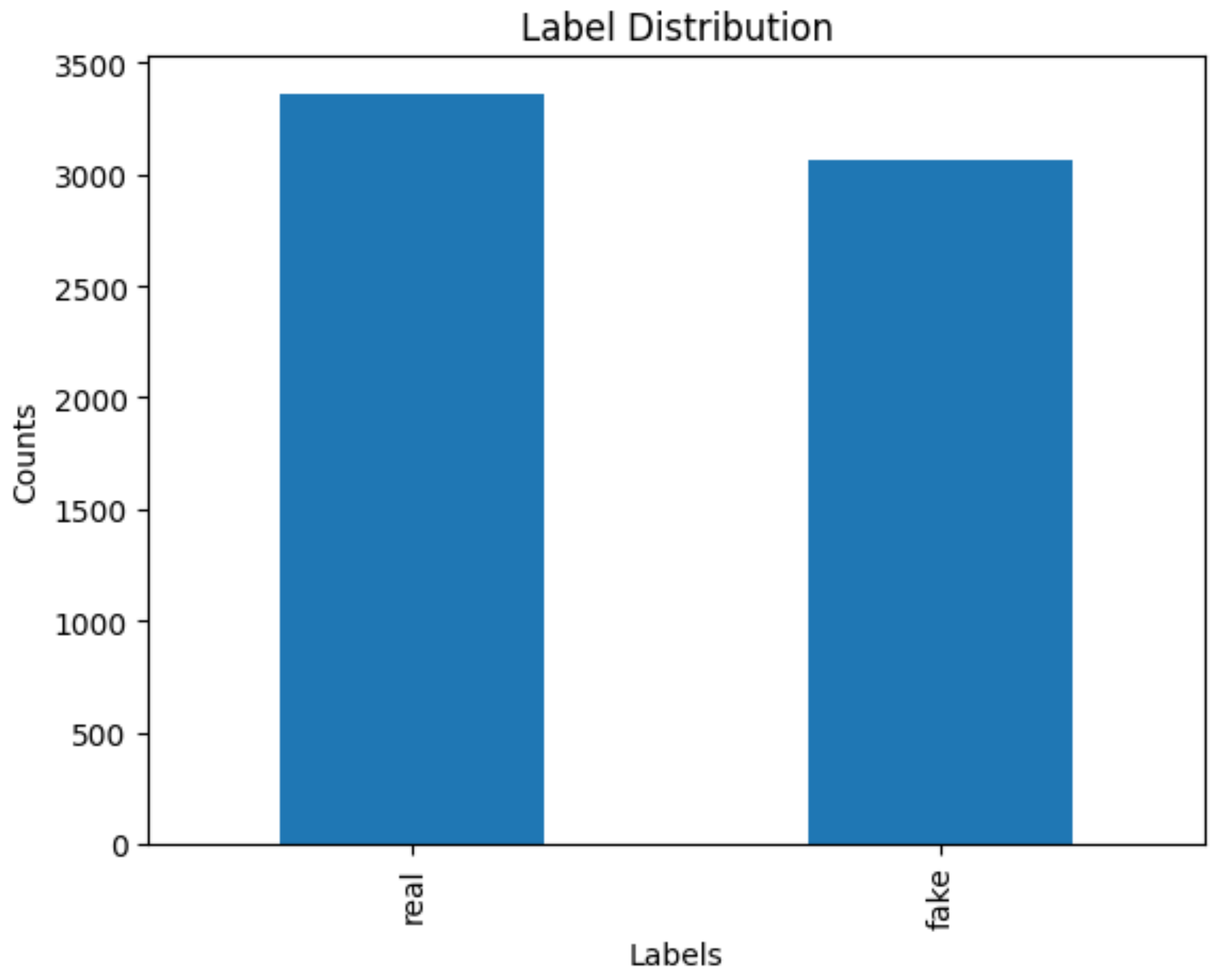
```
First few entries from claims_test_data:
[{'claim': '0 dimensional biomaterials lack inductive properties', 'explana
```

2. Data Exploration

```python
# Function to explore a dataset
def explore_dataset(data, dataset_name):
    print(f"Exploring dataset: {dataset_name}")
    print(f"Number of entries: {len(data)}")

    # Convert to DataFrame for easier analysis
    df = pd.DataFrame(data)

    # Display the first few entries
    print("First few entries:")
    print(df.head())

    # Display basic statistics
    print("\nBasic statistics:")
    print(df.describe(include='all'))

    # Check the distribution of labels (if applicable)
    if 'label' in df.columns:
        label_counts = df['label'].value_counts()
        print("\nLabel distribution:")
        print(label_counts)

        # Plot the label distribution
        label_counts.plot(kind='bar', title='Label Distribution')
        plt.xlabel('Labels')
        plt.ylabel('Counts')
        plt.show()

    print("\n" + "-" * 40 + "\n")

# Explore each dataset
explore_dataset(covid_train_data, "Cleaned Covid19 Train Data")
explore_dataset(healthfact_train_data, "Healthfact Train Data")
explore_dataset(scifact_train_data, "SciFact Train Data")
```

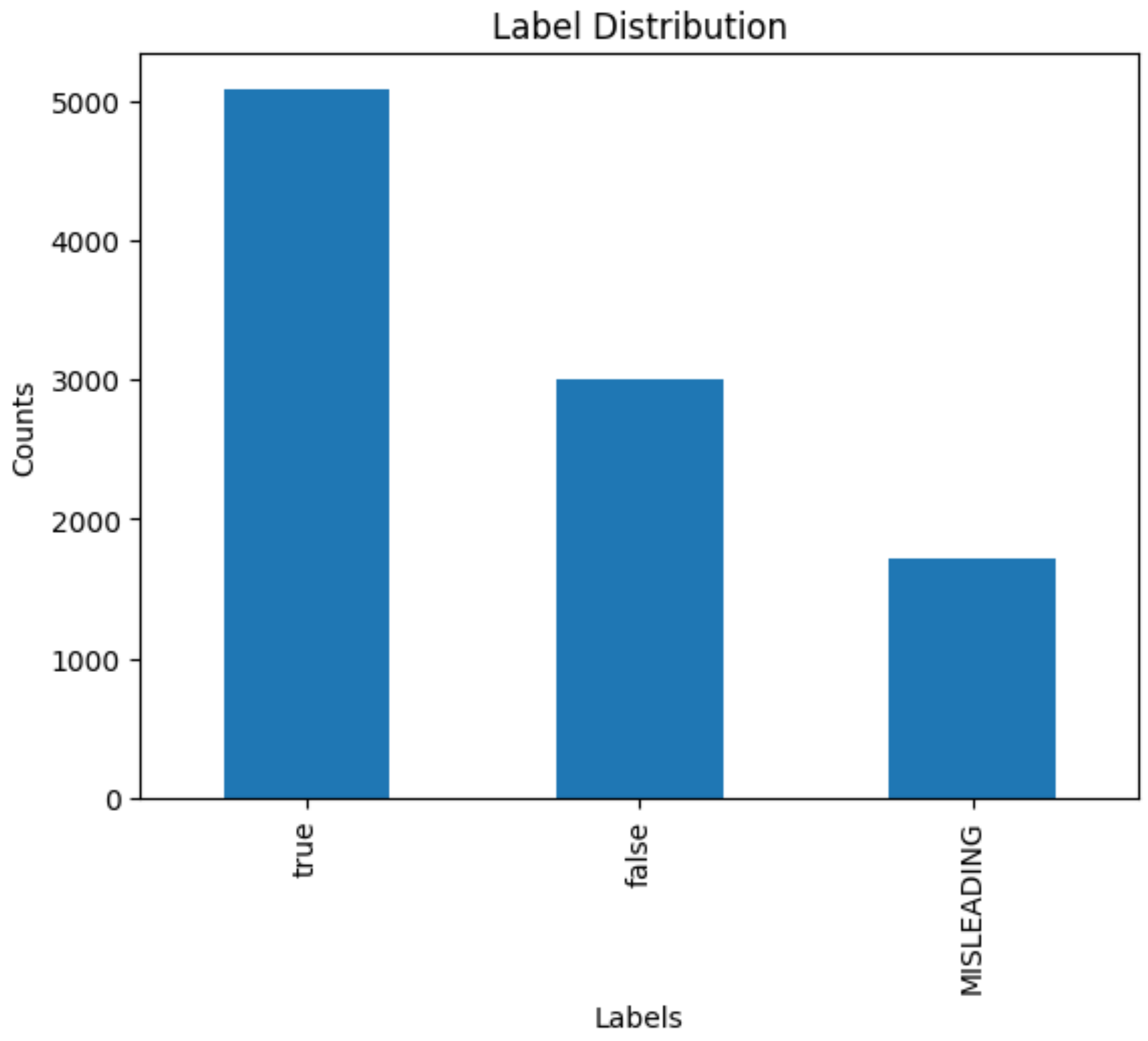
```
Exploring dataset: Cleaned Covid19 Train Data
Number of entries: 6420
First few entries:
                                               claim label
0  The CDC currently reports 99031 deaths In gene...  real
1  States reported 1121 deaths a small rise from ...  real
2  Politically Correct Woman Almost Uses Pandemic...  fake
3  IndiaFightsCorona We have 1524 COVID testing l...  real
4  Populous states can generate large case counts...  real

Basic statistics:
                                                    claim label
count                                                6420  6420
unique                                               6379     2
top     FREE HORSES 52 thoroughbred horses need homes ...  real
freq                                                    3  3360

Label distribution:
```

```
label
real    3360
fake    3060
Name: count, dtype: int64
```



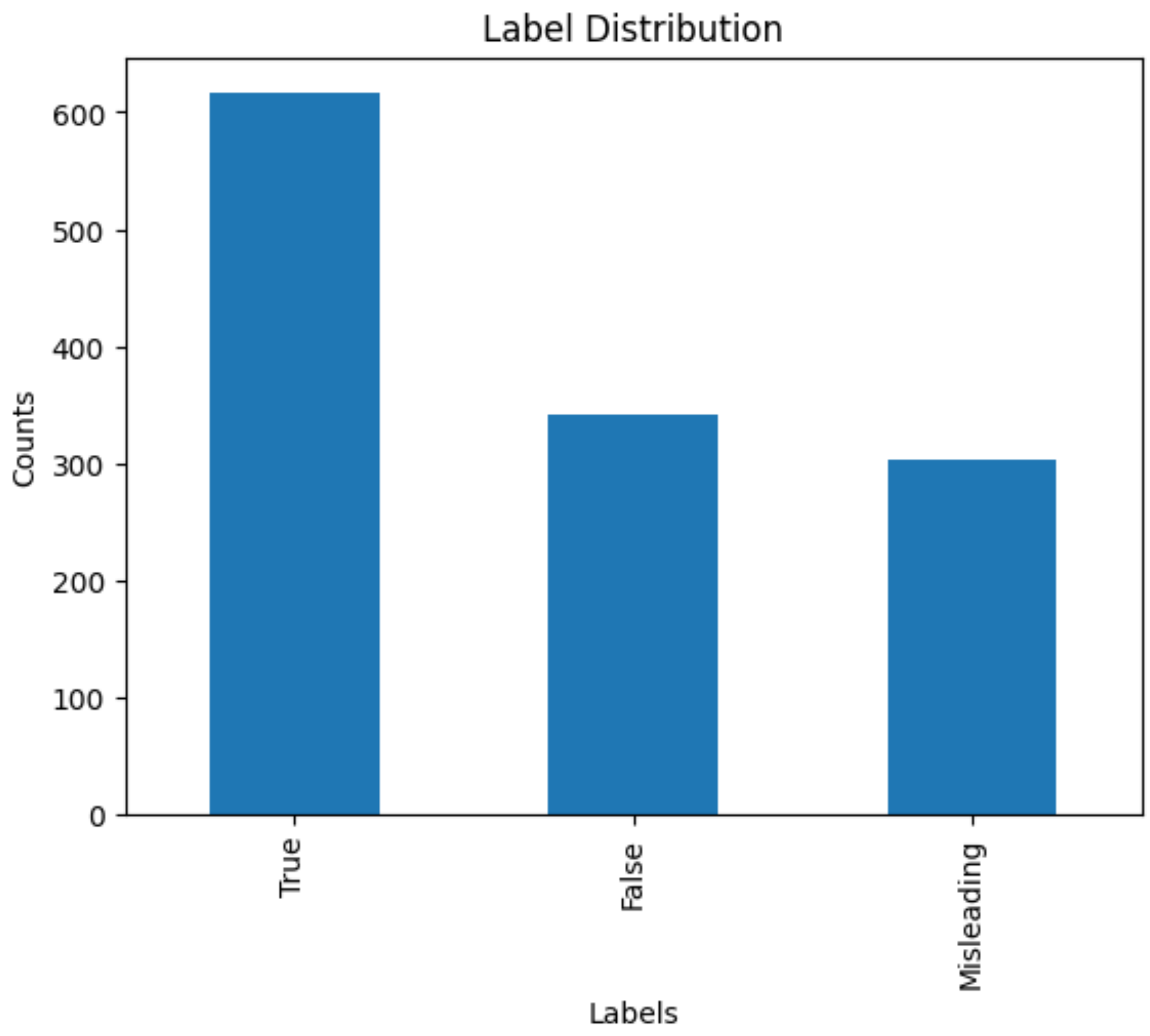
```
----------------------------------------

Exploring dataset: Healthfact Train Data
Number of entries: 9804
First few entries:
                                               claim  \
0  The money the Clinton Foundation took from fro...
1    Annual Mammograms May Have More False Positives
2  SBRT Offers Prostate Cancer Patients High Canc...
3  Study Vaccine for Breast Ovarian Cancer Has Po...
4  Some appendicitis cases may not require emerge...

                                         explanation       label
0  "Gingrich said the Clinton Foundation ""took m...       false
1  This article reports on the results of a study...  MISLEADING
2  This news release describes five-year outcomes...  MISLEADING
3  While the story does many things well, the ove...        true
4  We really don’t understand why only a handful ...        true

Basic statistics:
                                               claim  \
count                                           9804
unique                                          9799
top     Brain scans predict which dyslexics will read
freq                                               2
```

```
                                                explanation label
count                                                  9804  9804
unique                                                 9660     3
top     A U.S. judge on Wednesday appointed prominent ...  true
freq                                                      2  5078

Label distribution:
label
true          5078
false         3001
MISLEADING    1725
Name: count, dtype: int64
```



```
----------------------------------------

Exploring dataset: SciFact Train Data
Number of entries: 1261
First few entries:
                                               claim  \
0  0 dimensional biomaterials lack inductive prop...
1  1 in 5 million in UK have abnormal PrP positivity
2  1 1 of colorectal cancer patients are diagnose...
3  10 of sudden infant death syndrome SIDS deaths...
4  32 of liver transplantation programs required ...
```

```
                                         explanation       label
0                                                     Misleading
1  RESULTS Of the 32,441 appendix samples 16 were...       False
2                                                     Misleading
3                                                     Misleading
4  Policies requiring discontinuation of methadon...        True

Basic statistics:
                                                  claim explanation label
count                                              1261        1261  1261
unique                                              807         691     3
top     Energy balance requires hypothalamic glutamate...             True
freq                                                 11         304   616

Label distribution:
label
True          616
False         341
Misleading    304
Name: count, dtype: int64
```



```
----------------------------------------
```

3. Training Strategy

```
# Convert datasets to DataFrames for easier manipulation
healthfact_df = pd.DataFrame(healthfact_train_data)
scifact_df = pd.DataFrame(scifact_train_data)

# Combine HealthFact and SciFact datasets for pre-training
combined_pretrain_df = pd.concat([healthfact_df, scifact_df], ignore_index=

# Save the combined dataset for pre-training
combined_pretrain_df.to_json('combined_pretrain_data.jsonl', orient='recorc

# Convert COVID-19 dataset to DataFrame
covid_df = pd.DataFrame(covid_train_data)

# Save the COVID-19 dataset for fine-tuning
covid_df.to_json('covid_finetune_data.jsonl', orient='records', lines=True)

print("Datasets combined and saved for train dataset:")
print("1. Combined Pre-train Data: combined_pretrain_data.jsonl")
print("2. COVID-19 Fine-tune Data: covid_finetune_data.jsonl")
```

```
Datasets combined and saved for train dataset:
1. Combined Pre-train Data: combined_pretrain_data.jsonl
2. COVID-19 Fine-tune Data: covid_finetune_data.jsonl
```

```
# Convert datasets to DataFrames for easier manipulation
healthfact_df_test = pd.DataFrame(healthfact_test_data)
scifact_df_test = pd.DataFrame(scifact_test_data)

# Combine HealthFact and SciFact datasets for pre-training
combined_pretrain_df_test = pd.concat([healthfact_df_test, scifact_df_test]

# Save the combined dataset for pre-training
combined_pretrain_df_test.to_json('combined_pretrain_test_data.jsonl', orie

# Convert COVID-19 dataset to DataFrame
covid_df_test = pd.DataFrame(covid_dev_data)

# Save the COVID-19 dataset for fine-tuning
covid_df_test.to_json('covid_finetune_test_data.jsonl', orient='records', l

print("Datasets combined and saved for Test dataset:")
print("1. Combined Pre-train Data: combined_pretrain_test_data.jsonl")
print("2. COVID-19 Fine-tune Data: covid_finetune_test_data.jsonl")
```

```
Datasets combined and saved for Test dataset:
1. Combined Pre-train Data: combined_pretrain_test_data.jsonl
2. COVID-19 Fine-tune Data: covid_finetune_test_data.jsonl
```

4. Training and Evaluation Before fine tuning using Gemini 2.0 Flash

A. Training and Evaluation Combination Between HealthFact and Scifact Dataset

```
import pandas as pd
import torch
from sklearn.metrics import accuracy_score, precision_score, recall_score,
from transformers import BertTokenizer, BertForSequenceClassification, Trai
from sklearn.preprocessing import LabelEncoder
# Load the combined pre-training dataset (HealthFact + SciFact)
train_combined_data = pd.read_json('combined_pretrain_data.jsonl', lines=Tr
val_combined_data = pd.read_json('combined_pretrain_test_data.jsonl', lines

# Load the COVID-19 fine-tuning dataset
train_covid_data = pd.read_json('covid_finetune_data.jsonl', lines=True)
val_covid_data = pd.read_json('covid_finetune_test_data.jsonl', lines=True)

# Assuming the datasets have 'claim' and 'label' columns
train_claims = train_combined_data['claim'].tolist()
train_labels = train_combined_data['label'].tolist()
val_claims = val_combined_data['claim'].tolist()
```

```
val_labels = val_combined_data['label'].tolist()

# Convert string labels to integers
label_encoder = LabelEncoder()
train_labels = label_encoder.fit_transform(train_labels)
val_labels = label_encoder.transform(val_labels)

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

# Tokenize the input data for pre-training
train_encodings = tokenizer(train_claims, truncation=True, padding=True, ma
val_encodings = tokenizer(val_claims, truncation=True, padding=True, max_le

# Create a dataset class
class ClaimsDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

# Create datasets for pre-training
train_dataset = ClaimsDataset(train_encodings, train_labels)
val_dataset = ClaimsDataset(val_encodings, val_labels)

# Load the BERT model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',

# Define training arguments for pre-training with validation loss logging
training_args = TrainingArguments(
    output_dir='./results/pretrain',
    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/pretrain',
    logging_steps=10,
    eval_strategy="epoch",  # Updated to eval_strategy
)

# Create a Trainer instance for pre-training
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=lambda p: {
        'accuracy': accuracy_score(p.label_ids, p.predictions.argmax(-1)),
        'precision': precision_score(p.label_ids, p.predictions.argmax(-1),
        'recall': recall_score(p.label_ids, p.predictions.argmax(-1), avera
        'f1': f1_score(p.label_ids, p.predictions.argmax(-1), average='weig
        'roc_auc': roc_auc_score(p.label_ids, torch.softmax(torch.tensor(p.
    },
)

# Pre-train the model
trainer.train()

```

```
Some weights of BertForSequenceClassification were not initialized from the
You should probably TRAIN this model on a down-stream task to be able to us
```

[4152/4152 13:22, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1	Roc Auc
1	0.945300	0.989038	0.547237	0.474881	0.547237	0.479945	0.886185
2	0.570800	0.998892	0.594177	0.571580	0.594177	0.557430	0.893968
3	0.443300	1.220652	0.590612	0.597319	0.590612	0.590918	0.896878

```
TrainOutput(global_step=4152, training_loss=0.7195591835990577, metrics=
{'train runtime': 802.4998, 'train samples per second': 41.364,
```

B. Training and Evaluation Covid Fake News Dataset

```
# Prepare the training and validation data
train_covid_claims = train_covid_data['claim'].tolist()
train_covid_labels = train_covid_data['label'].tolist()
val_covid_claims = val_covid_data['claim'].tolist()
val_covid_labels = val_covid_data['label'].tolist()

# Convert string labels to integers
label_encoder = LabelEncoder()
train_covid_labels = label_encoder.fit_transform(train_covid_labels)
val_covid_labels = label_encoder.transform(val_covid_labels)

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

# Tokenize the input data for pre-training
train_encodings = tokenizer(train_covid_claims, truncation=True, padding=Tr
val_encodings = tokenizer(val_covid_claims, truncation=True, padding=True,

# Create datasets for pre-training
train_dataset = ClaimsDataset(train_encodings, train_labels)
val_dataset = ClaimsDataset(val_encodings, val_labels)

# Load the BERT model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',

# Pre-train the model
trainer.train()
```

```
Some weights of BertForSequenceClassification were not initialized from the
You should probably TRAIN this model on a down-stream task to be able to us
```

[4152/4152 13:28, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1	Roc Auc
1	0.425300	1.497399	0.566845	0.572513	0.566845	0.559901	0.874701
2	0.297900	1.842367	0.590018	0.596393	0.590018	0.591040	0.888562
3	0.208000	2.297663	0.588829	0.605546	0.588829	0.594764	0.889416

```
TrainOutput(global_step=4152, training_loss=0.3173761432093204, metrics=
{'train runtime': 809.0842, 'train samples per second': 41.028,
```

5. Model Initialization

```
import google.generativeai as genai

# Initialize the Gemini 2.0 Flash Model:
API_KEY = "AIzaSyCjJIJ3ntglUAqZHn6qZaOE5uIGg4txEC4"
genai.configure(api_key=API_KEY)

# Load the Gemini model
model = genai.GenerativeModel("gemini-2.0-flash")
```

```
# Define a fine-tuning function using Gemini API
def generate_response(prompt):
    response = model.generate_content(prompt)
    return response.text

# Example few-shot training prompt
prompt = """
Claim: "6 10 Sky s EdConwaySky explains the latest COVID19 data and governme
"""

response = generate_response(prompt)
print(response)
```

```
This claim appears to be a tweet or social media post promoting a segment o

**Here's a breakdown of the elements:**

* **"6 10"**: This likely refers to the time the tweet was posted, possibly
* **"Sky s"**:  This is likely a shortened form of "Sky News's".
* **"EdConwaySky"**: This is probably the Twitter handle for Ed Conway, who
* **"explains the latest COVID19 data and government announcement"**: This
* **"Get more on the coronavirus data here"**: This is a call to action, en
* **"https t co jvGZlSbFjH https t co PygSKXesBg"**: These are shortened UR

**Potential Issues and Things to Consider:**

* **Data Accuracy:**  While the claim itself isn't making a specific factua
* **Bias:**  It's important to be aware of potential biases. Sky News, like
* **Outdated Information:**  COVID-19 data and government announcements cha
* **Link Safety:** Always be cautious when clicking on shortened links, esp

**In Conclusion:**

The claim itself is a straightforward promotion of a news segment.  However
```

5. Model Initialization

```
!pip install --upgrade google-genai
!gcloud auth application-default login
```

```
Requirement already satisfied: google-genai in /usr/local/lib/python3.11/di
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /usr/local/lib/python
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /usr/local/lib
Requirement already satisfied: httpx<1.0.0,>=0.28.1 in /usr/local/lib/pytho
Requirement already satisfied: pydantic<3.0.0,>=2.0.0 in /usr/local/lib/pyt
Requirement already satisfied: requests<3.0.0,>=2.28.1 in /usr/local/lib/py
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /usr/local/lib
Requirement already satisfied: typing-extensions<5.0.0,>=4.11.0 in /usr/loc
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/pyt
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/pyth
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/d
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/d
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.33.0 in /usr/local/lib/pyth
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/p
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/pytho
Go to the following link in your browser, and complete the sign-in prompts:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=

Once finished, enter the verification code provided in your browser: 4/0Ab_

Credentials saved to file: [/content/.config/application_default_credential

These credentials will be used by any library that requests Application Def
WARNING:
Cannot find a quota project to add to ADC. You might receive a "quota excee
```

```
!pip install --upgrade google-cloud-aiplatform
```

```
Requirement already satisfied: google-cloud-aiplatform in /usr/local/lib/py
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /usr/local/lib
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in /usr/local/lib/
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.11
Requirement already satisfied: google-cloud-storage<3.0.0,>=1.32.0 in /usr/
Requirement already satisfied: google-cloud-bigquery!=3.20.0,<4.0.0,>=1.15.
Requirement already satisfied: google-cloud-resource-manager<3.0.0,>=1.3.3 
Requirement already satisfied: shapely<3.0.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: pydantic<3 in /usr/local/lib/python3.11/dist
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.
Requirement already satisfied: docstring-parser<1 in /usr/local/lib/python3
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in /
Requirement already satisfied: requests<3.0.0,>=2.18.0 in /usr/local/lib/py
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in /usr/local/lib/pyt
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in /usr/loca
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/pyt
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/pyth
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/d
Requirement already satisfied: google-cloud-core<3.0.0,>=2.4.1 in /usr/loca
Requirement already satisfied: google-resumable-media<3.0.0,>=2.0.0 in /usr
Requirement already satisfied: python-dateutil<3.0.0,>=2.8.2 in /usr/local/
Requirement already satisfied: grpc-google-iam-v1<1.0.0,>=0.14.0 in /usr/lo
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /usr/local/lib
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.33.0 in /usr/local/lib/pyth
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/p
Requirement already satisfied: numpy<3,>=1.14 in /usr/local/lib/python3.11/
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
```

```python
from google.colab import auth as google_auth
google_auth.authenticate_user()

import vertexai
from vertexai.generative_models import GenerativeModel
from vertexai.preview.tuning import sft

vertexai.init(project="sit319-25t1-nguyen-ae806d0", location="us-central1")

gemini_pro = GenerativeModel("gemini-2.0-flash-lite-001")

sft_tuning_job = sft.train(
    source_model=gemini_pro,
    train_dataset="gs://daftt/Cleaned_Covid19_Train-7.jsonl",
    tuned_model_display_name="covid_tuning",
    epochs=100,
    learning_rate_multiplier=1,
)
```

```
/usr/local/lib/python3.11/dist-packages/google/auth/_default.py:76: UserWar
  warnings.warn(_CLOUD_SDK_CREDENTIALS_WARNING)
INFO:vertexai.tuning._tuning:Creating SupervisedTuningJob
/usr/local/lib/python3.11/dist-packages/google/auth/_default.py:76: UserWar
  warnings.warn(_CLOUD_SDK_CREDENTIALS_WARNING)
INFO:vertexai.tuning._tuning:SupervisedTuningJob created. Resource name: pr
INFO:vertexai.tuning._tuning:To use this SupervisedTuningJob in another ses
INFO:vertexai.tuning._tuning:tuning_job = sft.SupervisedTuningJob('projects
INFO:vertexai.tuning._tuning:View Tuning Job:
https://console.cloud.google.com/vertex-ai/generative/language/locations/us
```

VIEW TUNING JOB

```python
from google import genai
from google.genai import types
import base64

def generate():
  client = genai.Client(
      vertexai=True,
      project="181085238689",
      location="us-central1",
  )

  msg3_text1 = types.Part.from_text(text="""Clearly the Obama administratic

  model = "projects/181085238689/locations/us-central1/endpoints/541977098
  contents = [
    types.Content(
      role="user",
      parts=[
        types.Part.from_text(text="""Multiple Facebook posts claim that Aus
      ]
    ),
    types.Content(
      role="model",
      parts=[
        types.Part.from_text(text=label)
      ]
    ),
    types.Content(
      role="user",
      parts=[
        msg3_text1
      ]
    ),
  ]
  generate_content_config = types.GenerateContentConfig(
    temperature = 0.2,
    top_p = 0.8,
    max_output_tokens = 1024,
    response_modalities = ["TEXT"],
    safety_settings = [types.SafetySetting(
      category="HARM_CATEGORY_HATE_SPEECH",
      threshold="OFF"
    ),types.SafetySetting(
      category="HARM_CATEGORY_DANGEROUS_CONTENT",
      threshold="OFF"
    ),types.SafetySetting(
      category="HARM_CATEGORY_SEXUALLY_EXPLICIT",
      threshold="OFF"
    ),types.SafetySetting(
      category="HARM_CATEGORY_HARASSMENT",
      threshold="OFF"
    )],
  )

  for chunk in client.models.generate_content_stream(
    model = model,
    contents = contents,
    config = generate_content_config,
    ):
    print(chunk.text, end="")

generate()
```

```
/usr/local/lib/python3.11/dist-packages/google/auth/_default.py:76: UserWar
  warnings.warn(_CLOUD_SDK_CREDENTIALS_WARNING)
fake
```

```
Start coding or generate with AI.
```