```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files dataset (5).zip

- **dataset (5).zip**(application/zip) - 87162674 bytes, last modified: 10/30/2023 - 100% done

```
Saving dataset (5).zip to dataset (5).zip
```

```
import zipfile
with zipfile.ZipFile('dataset (5).zip', 'r') as zip_ref:
    zip_ref.extractall('/content/dataset (5)')
```

```
# Import necessary libraries
import json
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Data Collection
data_dir = '/content/dataset (5)'
json_file = os.path.join(data_dir, '/content/pavbhaji.json')

with open(json_file, 'r') as json_data:
    data = json.load(json_data)
```

```
# Step 2: Data Preprocessing
texts = []
labels = []

for entry in data:
    # Check if 'edges' list exists and has at least one element
    if 'edge_media_to_caption' in entry and 'edges' in entry['edge_media_to_caption'] and len(entry['edge_media_to_caption']['edges']) > 0:
        text = entry['edge_media_to_caption']['edges'][0]['node']['text']  # Extract 'text' data

        # Check if 'pavbhaji' is in the 'tags' to determine the label
        tags = entry['tags']
        label = 1 if 'pavbhaji' in tags else 0

        texts.append(text)
        labels.append(label)
    else:
        # Handle cases where 'edges' list is empty or doesn't exist
        texts.append("")  # Add an empty string for text
        labels.append(0)  # Assume a label of 0 for these cases
```

```
# Step 3: Feature Extraction (TF-IDF Vectorization)
tfidf_vectorizer = TfidfVectorizer(max_features=1000)  # You can adjust the number of features
X = tfidf_vectorizer.fit_transform(texts)

# Step 4: Label Encoding
y = labels

# Step 5: Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Model Selection and Training (Logistic Regression)
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
# Step 7: Model Evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
print(report)

# Step 8: Make Predictions
new_text = "This is a delicious Pav Bhaji!"
new_text_vector = tfidf_vectorizer.transform([new_text])
prediction = model.predict(new_text_vector)

if prediction[0] == 1:
    print("The text indicates Pav Bhaji.")
else:
    print("The text does not indicate Pav Bhaji.")
```

```
Accuracy: 0.91
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        26
           1       0.91      1.00      0.95       274

    accuracy                           0.91       300
   macro avg       0.46      0.50      0.48       300
weighted avg       0.83      0.91      0.87       300

The text indicates Pav Bhaji.
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
  _warn_prf(average, modifier, msg_start, len(result))
```