```
#Accelerator-GPU P100
!pip install datasets -U
from datasets import load_dataset
import matplotlib.pyplot as plt
import torch
```

```
Requirement already satisfied: datasets in
/opt/conda/lib/python3.10/site-packages (2.1.0)
Collecting datasets
  Obtaining dependency information for datasets from
https://files.pythonhosted.org/packages/ec/93/454ada0d1b289a0f4a86ac88
dbdeab54921becabac45da3da787d136628f/datasets-2.16.1-py3-none-
any.whl.metadata
  Downloading datasets-2.16.1-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in
/opt/conda/lib/python3.10/site-packages (from datasets) (3.12.2)
Requirement already satisfied: numpy>=1.17 in
/opt/conda/lib/python3.10/site-packages (from datasets) (1.24.3)
Requirement already satisfied: pyarrow>=8.0.0 in
/opt/conda/lib/python3.10/site-packages (from datasets) (11.0.0)
Collecting pyarrow-hotfix (from datasets)
  Obtaining dependency information for pyarrow-hotfix from
https://files.pythonhosted.org/packages/e4/f4/9ec2222f5f5f8ea04f66f184
caafd991a39c8782e31f5b0266f101cb68ca/pyarrow_hotfix-0.6-py3-none-
any.whl.metadata
  Downloading pyarrow_hotfix-0.6-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: dill<0.3.8,>=0.3.0 in
/opt/conda/lib/python3.10/site-packages (from datasets) (0.3.7)
Requirement already satisfied: pandas in
/opt/conda/lib/python3.10/site-packages (from datasets) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in
/opt/conda/lib/python3.10/site-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in
/opt/conda/lib/python3.10/site-packages (from datasets) (4.66.1)
Requirement already satisfied: xxhash in
/opt/conda/lib/python3.10/site-packages (from datasets) (3.4.1)
Requirement already satisfied: multiprocess in
/opt/conda/lib/python3.10/site-packages (from datasets) (0.70.15)
Collecting fsspec[http]<=2023.10.0,>=2023.1.0 (from datasets)
  Obtaining dependency information for
fsspec[http]<=2023.10.0,>=2023.1.0 from
https://files.pythonhosted.org/packages/e8/f6/3eccfb530aac90ad1301c582
da228e4763f19e719ac8200752a4841b0b2d/fsspec-2023.10.0-py3-none-
any.whl.metadata
  Downloading fsspec-2023.10.0-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: aiohttp in
/opt/conda/lib/python3.10/site-packages (from datasets) (3.8.5)
Requirement already satisfied: huggingface-hub>=0.19.4 in
/opt/conda/lib/python3.10/site-packages (from datasets) (0.20.2)
Requirement already satisfied: packaging in
```

```
/opt/conda/lib/python3.10/site-packages (from datasets) (21.3)
Requirement already satisfied: pyyaml>=5.1 in
/opt/conda/lib/python3.10/site-packages (from datasets) (6.0.1)
Requirement already satisfied: attrs>=17.3.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(23.1.0)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(3.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(4.0.3)
Requirement already satisfied: yarl<2.0,>=1.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/opt/conda/lib/python3.10/site-packages (from aiohttp->datasets)
(1.3.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/opt/conda/lib/python3.10/site-packages (from huggingface-hub>=0.19.4-
>datasets) (4.5.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging->datasets)
(3.0.9)
Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/lib/python3.10/site-packages (from requests>=2.19.0-
>datasets) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests>=2.19.0-
>datasets) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests>=2.19.0-
>datasets) (2023.11.17)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas->datasets)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/opt/conda/lib/python3.10/site-packages (from pandas->datasets)
(2023.3)
Requirement already satisfied: tzdata>=2022.1 in
/opt/conda/lib/python3.10/site-packages (from pandas->datasets)
(2023.3)
Requirement already satisfied: six>=1.5 in
```

```
/opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.16.0)
Downloading datasets-2.16.1-py3-none-any.whl (507 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 507.1/507.1 kB 10.1 MB/s eta
0:00:0000:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.4/166.4 kB 14.9 MB/s eta
0:00:00
pting uninstall: fsspec
    Found existing installation: fsspec 2023.12.2
    Uninstalling fsspec-2023.12.2:
      Successfully uninstalled fsspec-2023.12.2
  Attempting uninstall: datasets
    Found existing installation: datasets 2.1.0
    Uninstalling datasets-2.1.0:
      Successfully uninstalled datasets-2.1.0
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
cudf 23.8.0 requires cupy-cuda11x>=12.0.0, which is not installed.
cuml 23.8.0 requires cupy-cuda11x>=12.0.0, which is not installed.
dask-cudf 23.8.0 requires cupy-cuda11x>=12.0.0, which is not
installed.
cudf 23.8.0 requires pandas<1.6.0dev0,>=1.3, but you have pandas 2.0.3
which is incompatible.
cudf 23.8.0 requires protobuf<5,>=4.21, but you have protobuf 3.20.3
which is incompatible.
cuml 23.8.0 requires dask==2023.7.1, but you have dask 2023.12.1 which
is incompatible.
cuml 23.8.0 requires distributed==2023.7.1, but you have distributed
2023.12.1 which is incompatible.
dask-cuda 23.8.0 requires dask==2023.7.1, but you have dask 2023.12.1
which is incompatible.
dask-cuda 23.8.0 requires distributed==2023.7.1, but you have
distributed 2023.12.1 which is incompatible.
dask-cuda 23.8.0 requires pandas<1.6.0dev0,>=1.3, but you have pandas
2.0.3 which is incompatible.
dask-cudf 23.8.0 requires dask==2023.7.1, but you have dask 2023.12.1
which is incompatible.
dask-cudf 23.8.0 requires distributed==2023.7.1, but you have
distributed 2023.12.1 which is incompatible.
dask-cudf 23.8.0 requires pandas<1.6.0dev0,>=1.3, but you have pandas
2.0.3 which is incompatible.
gcsfs 2023.6.0 requires fsspec==2023.6.0, but you have fsspec
2023.10.0 which is incompatible.
raft-dask 23.8.0 requires dask==2023.7.1, but you have dask 2023.12.1
which is incompatible.
raft-dask 23.8.0 requires distributed==2023.7.1, but you have
distributed 2023.12.1 which is incompatible.
s3fs 2023.12.2 requires fsspec==2023.12.2, but you have fsspec
```

```
2023.10.0 which is incompatible.
Successfully installed datasets-2.16.1 fsspec-2023.10.0 pyarrow-
hotfix-0.6
```

```
emotions =load_dataset("emotion", download_mode="force_redownload")
```

```
/opt/conda/lib/python3.10/site-packages/datasets/load.py:1429:
FutureWarning: The repository for emotion contains custom code which
must be executed to correctly load the dataset. You can inspect the
repository content at https://hf.co/datasets/emotion
You can avoid this message in future by passing the argument
`trust_remote_code=True`.
Passing `trust_remote_code=True` will be mandatory to load this
dataset from the next major release of `datasets`.
  warnings.warn(
```


```
{"model_id":"827716db881b40318e25621927105f54","version_major":2,"vers
ion_minor":0}

{"model_id":"3d277464ecb8426ba31eaaf8db5d676b","version_major":2,"vers
ion_minor":0}

{"model_id":"6eb29645fe234d1db1e6b116b53ccf79","version_major":2,"vers
ion_minor":0}
```


```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```


```
{"model_id":"09bf4c39f37f4dd4a5efb47a4c2d5e84","version_major":2,"vers
ion_minor":0}

{"model_id":"d1083bcea56948fda30c3279e1767abe","version_major":2,"vers
ion_minor":0}

{"model_id":"38c7d604ca974264ba2d07534107e766","version_major":2,"vers
ion_minor":0}

{"model_id":"84f120bf77064ec78be6b1dd68bc6bd1","version_major":2,"vers
ion_minor":0}

{"model_id":"df80e0724a15472881ac890d7721f7f1","version_major":2,"vers
ion_minor":0}

{"model_id":"25c74ece12c342a0ae7e0d720f47d1bf","version_major":2,"vers
ion_minor":0}
```


```
emotions["train"].features
```

```
{'text': Value(dtype='string', id=None),
 'label': ClassLabel(names=['sadness', 'joy', 'love', 'anger', 'fear',
'surprise'], id=None)}
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
train_ds=emotions["train"]
device
```

```
device(type='cuda')
```

```
train_ds
```

```
Dataset({
    features: ['text', 'label'],
    num_rows: 16000
})
```

```
train_ds[1]
```

```
{'text': 'i can go from feeling so hopeless to so damned hopeful just
from being around someone who cares and is awake',
 'label': 0}
```

```
import pandas as pd
```

```
emotions.set_format(type="pandas")
df=emotions["train"][:]
```

```
df.head(5)
```

	text	label
0	i didnt feel humiliated	0
1	i can go from feeling so hopeless to so damned...	0
2	im grabbing a minute to post i feel greedy wrong	3
3	i am ever feeling nostalgic about the fireplac...	2
4	i am feeling grouchy	3

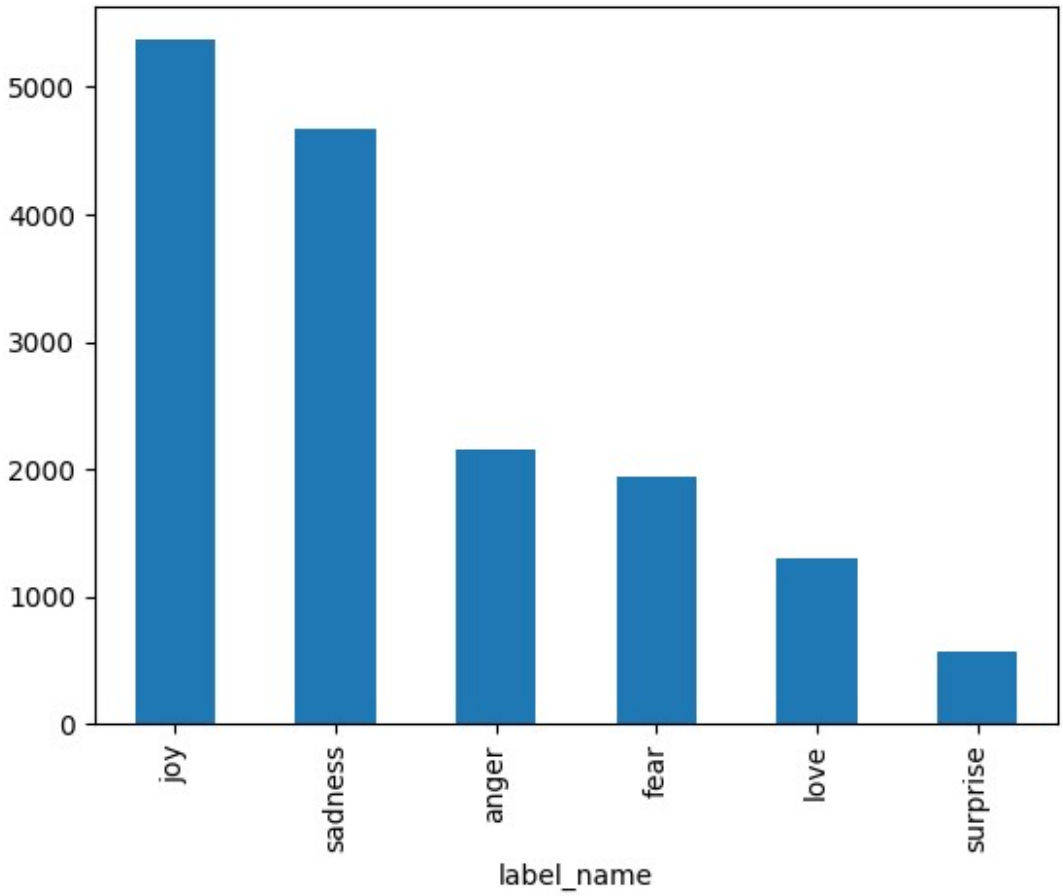
```
def label_int2str(row,split):
    return emotions[split].features["label"].int2str(row)
```

```
df["label_name"]=df["label"].apply(label_int2str,split="train")
df.head(5)
labels=list(df["label_name"].unique())
labels
```

```
['sadness', 'anger', 'love', 'surprise', 'fear', 'joy']
```

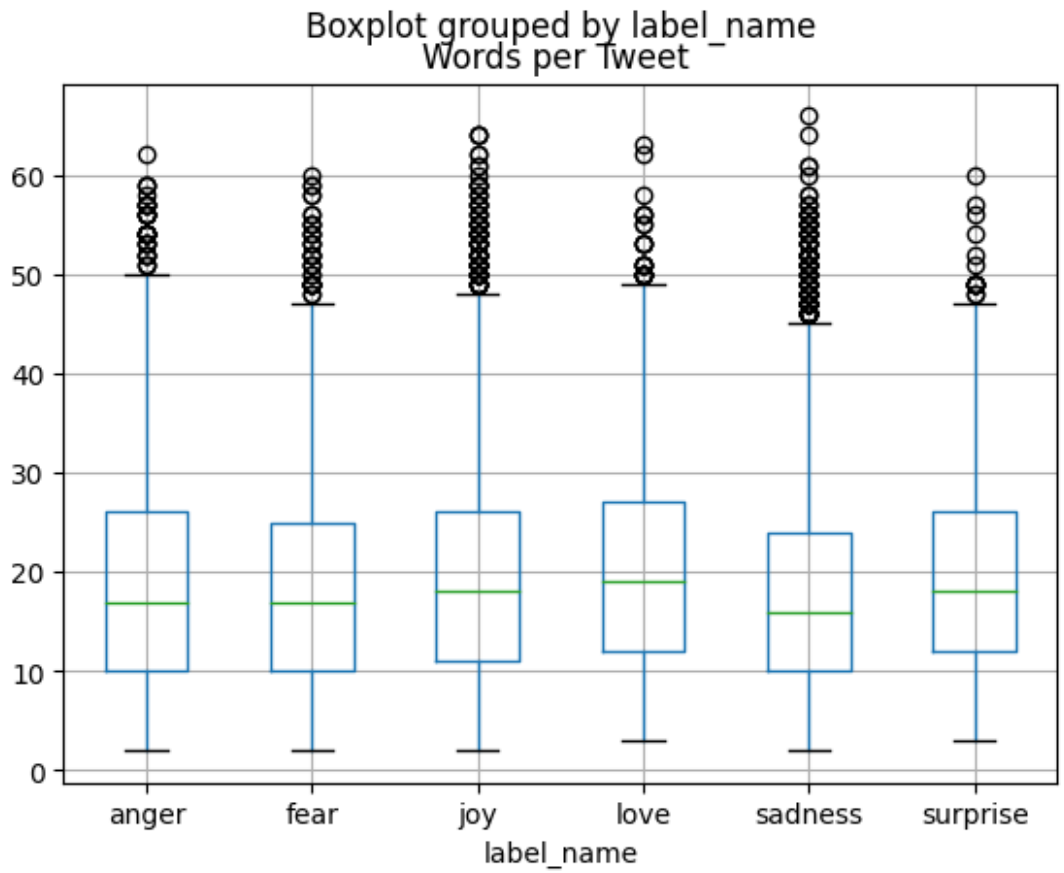
```
df["label_name"].value_counts().plot.bar()
```

```
<Axes: xlabel='label_name'>
```




```
df["Words per Tweet "]=df["text"].str.split().apply(len)
df.boxplot("Words per Tweet ",by="label_name")
```

```
<Axes: title={'center': 'Words per Tweet '}, xlabel='label_name'>
```




```
from transformers import AutoTokenizer
model_name="distilbert-base-uncased"
tokenizer=AutoTokenizer.from_pretrained(model_name)
```


```
{"model_id":"eb9358b4dfa84d749165cb032ea6d7ed","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"a0b4fe31b8704bb2ad22fa490b1458d1","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"49ef2221346346acabaded3f97f8ff70","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"af1bf438284a44518b612781b84b36f8","version_major":2,"vers
ion_minor":0}
```


```
import torch

text="This is a text"

from transformers import AutoModel
text_tensor=tokenizer.encode(text,return_tensors="pt")
model=AutoModel.from_pretrained(model_name).to("cuda")
```

{"model_id":"d2c42519fc9f4d768268859535031538","version_major":2,"version_minor":0}

```
def tokenize(batch):
    return tokenizer(batch["text"],padding=True,truncation=True)

emotions.reset_format()

emotions["train"][1]
```

```
{'text': 'i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake',
 'label': 0}
```

```
emotions_encoded=emotions.map(tokenize,batched=True,batch_size=None)
```

{"model_id":"47e3f1a5e3ae47209fd458ad9a798e6a","version_major":2,"version_minor":0}

{"model_id":"7cb0a9816d554b5fa694418d32e95dcd","version_major":2,"version_minor":0}

{"model_id":"fe9d87a5136045319753fb87933d1a12","version_major":2,"version_minor":0}

```
emotions_encoded["train"][1]
```

```
{'text': 'i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake',
 'label': 0,
 'input_ids': [101,
  1045,
  2064,
  2175,
  2013,
  3110,
  2061,
  20625,
  2000,
  2061,
  9636,
  17772,
  2074,
  2013,
  2108,
  2105,
  2619,
  2040,
  14977,
  1998,
  2003,
```

```
  8300,
  102,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
```

```
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0],
 'attention_mask': [1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
```

```
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
```

```
  0,
  0,
  0,
  0,
  0,
  0]}
```

```
import numpy as np
```

```
import numpy as np
def forward_pass(batch):
    input_ids = torch.tensor(batch["input_ids"]).to("cuda")
    attention_mask = torch.tensor(batch["attention_mask"]).to("cuda")
    with torch.no_grad():
        last_hidden_state = model(input_ids,
attention_mask).last_hidden_state
        last_hidden_state = last_hidden_state.cpu().numpy()
 # Use average of unmasked hidden states for classification
    lhs_shape = last_hidden_state.shape
    boolean_mask = ~np.array(batch["attention_mask"]).astype(bool)
    boolean_mask = np.repeat(boolean_mask, lhs_shape[-1], axis=-1)
    boolean_mask = boolean_mask.reshape(lhs_shape)

    masked_mean = np.ma.array(last_hidden_state,
mask=boolean_mask).mean(axis=1)

    batch["hidden_state"] = masked_mean.data
    return batch


emotions_encoded = emotions_encoded.map(forward_pass, batched=True,
 batch_size=16)
```


```
{"model_id":"387e85021d84461d912c5e42ead8f81b","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"c3e380582ef44ecb8542af6f2f233c25","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"c229dc782839467d85790f8a16b2a7d8","version_major":2,"vers
ion_minor":0}
```


```
len(emotions_encoded["train"][1]["hidden_state"])
```

```
768
```

```
import numpy as np

X_train=np.array(emotions_encoded["train"]["hidden_state"])
```

```
X_valid=np.array(emotions_encoded["validation"]["hidden_state"])
```

```
y_train=np.array(emotions_encoded["train"]["label"])
```

```
y_valid=np.array(emotions_encoded["validation"]["label"])
```

```
X_train.shape
```

```
(16000, 768)
```

```
X_valid.shape
```

```
(2000, 768)
```

```
from umap import UMAP
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
X_scaled=MinMaxScaler().fit_transform(X_train)
```

```
mapper=UMAP(n_components=2,metric="cosine").fit(X_scaled)
```

```
df_emb=pd.DataFrame(mapper.embedding_,columns=['X','Y'])
```

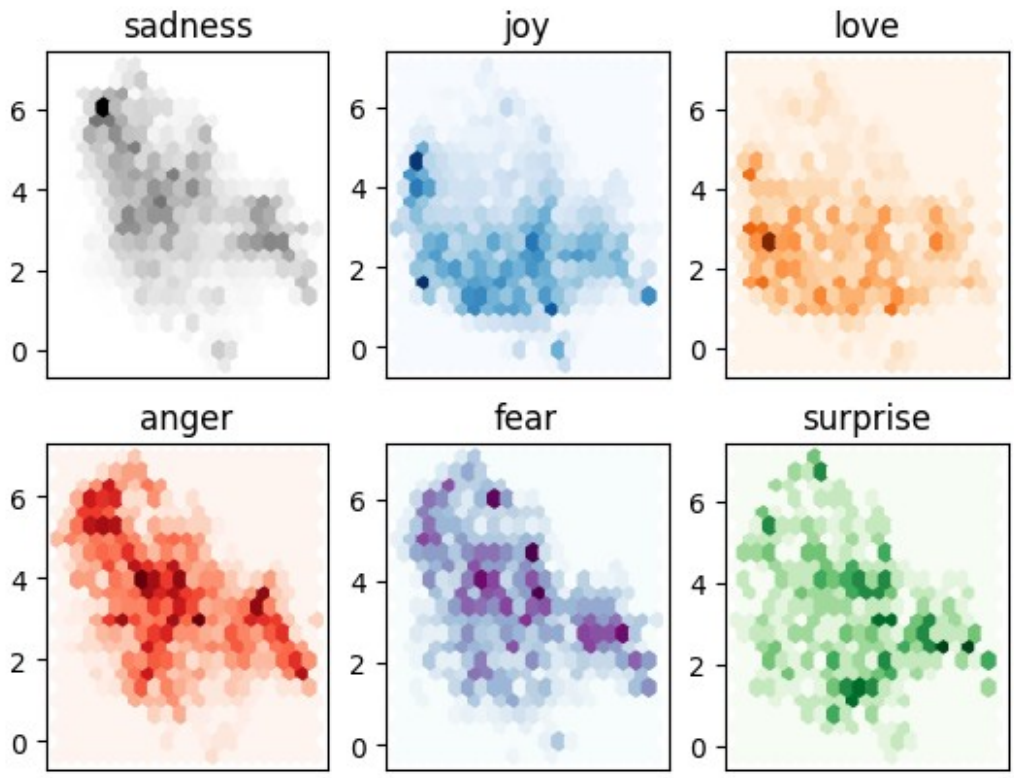
```
df_emb['label'] = y_train
```

```
df_emb.head(5)
```

```
          X         Y  label
0  6.055369  4.180999      0
1  1.216109  4.081319      0
2  5.206962  1.192381      3
3  2.230678  2.506836      2
4  0.565245  6.404640      3
```

PLOTTING LABEL WISE TO SEE SIMILARITIES/DIFFERENCES BETWEEN TENSOR VECTOR OF DIFFERENT LABELS.

```
fig,axes=plt.subplots(2,3)
axes=axes.flatten()
labels=["sadness","joy","love","anger","fear","surprise"]
cmaps=["Greys","Blues","Oranges","Reds","BuPu","Greens"]
for i,(label,cmap) in enumerate(zip(labels,cmaps)):
    df_emb_sub=df_emb.query(f"label=={i}")

axes[i].hexbin(df_emb_sub["X"],df_emb_sub["Y"],cmap=cmap,gridsize=20)
    axes[i].set_title(label)
    axes[i].set_xticks([])
```



PERFORMING BASIC PREDICTIONS USING LOGISTIC REGRESSIONS (SKLEARN)

```
import numpy as np
from sklearn.linear_model import LogisticRegression
np.__version__
```

```
'1.24.3'
```

```
lr=LogisticRegression(n_jobs=-1,penalty=None)
```

```
lr.fit(X_train,y_train)
```

```
huggingface/tokenizers: The current process just got forked, after
parallelism has already been used. Disabling parallelism to avoid
deadlocks...
To disable this warning, you can either:
      - Avoid using `tokenizers` before the fork if possible
      - Explicitly set the environment variable
TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after
```

```
parallelism has already been used. Disabling parallelism to avoid
deadlocks...
To disable this warning, you can either:
      - Avoid using `tokenizers` before the fork if possible
      - Explicitly set the environment variable
TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after
parallelism has already been used. Disabling parallelism to avoid
deadlocks...
To disable this warning, you can either:
      - Avoid using `tokenizers` before the fork if possible
      - Explicitly set the environment variable
TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after
parallelism has already been used. Disabling parallelism to avoid
deadlocks...
To disable this warning, you can either:
      - Avoid using `tokenizers` before the fork if possible
      - Explicitly set the environment variable
TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after
parallelism has already been used. Disabling parallelism to avoid
deadlocks...
To disable this warning, you can either:
      - Avoid using `tokenizers` before the fork if possible
      - Explicitly set the environment variable
TOKENIZERS_PARALLELISM=(true | false)
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic
.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression(n_jobs=-1, penalty=None)
```

```
lr.score(X_valid,y_valid)  #PRINTS ACCURACY ON VALIDATION DATA
```

```
0.65
```

USING DUMMY CLASSIFIER (SKLEARN) TO COMPARE RESULT WITH LOGISTIC REGRESSION ACCURACY

```
from sklearn.dummy import DummyClassifier

dum=DummyClassifier(strategy="uniform")

dum.fit(X_train,y_train)

DummyClassifier(strategy='uniform')

dum.score(X_valid,y_valid) #ACCURACY OF DUMMY CLASSIFIER

0.156
```

DUMMY CLASSIFIER ACCURACY IS 35.2% AND THAT OF LOGISTIC REGRESSION IS 65.25%

```
y_preds=dum.predict(X_valid)

len(y_preds)

2000

from sklearn.metrics import classification_report
print(classification_report(y_valid, y_preds, target_names=labels))
```

```
              precision    recall  f1-score   support

     sadness       0.28      0.18      0.22       550
         joy       0.37      0.18      0.25       704
        love       0.08      0.15      0.11       178
       anger       0.11      0.15      0.13       275
        fear       0.07      0.11      0.09       212
    surprise       0.03      0.11      0.05        81

    accuracy                           0.17      2000
   macro avg       0.16      0.15      0.14      2000
weighted avg       0.24      0.17      0.19      2000
```

```
from transformers import AutoModelForSequenceClassification

num_labels=6
```

```
device=torch.device("cpu")
model=AutoModelForSequenceClassification.from_pretrained(model_name,num_labels=num_labels).to(device)
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are newly initialized: ['classifier.weight', 'pre_classifier.weight', 'pre_classifier.bias', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
emotions_encoded.set_format("torch",columns=["input_ids","attention_mask","label"])
```

```
emotions_encoded["train"][0]
```

```
{'label': tensor(0),
 'input_ids': tensor([  101,  1045,  2134,  2102,  2514, 26608,   102,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0]),
 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])}
```

```
from sklearn.metrics import accuracy_score,f1_score

def compute_metrics(pred):
    labels=pred.label_ids
    preds=pred.predictions.argmax(-1)
    f1=f1_score(labels,preds,average="weighted")
    acc=accuracy_score(labels,preds)
    return {"accuracy":acc,"f1":f1}
```

```
from transformers import Trainer,TrainingArguments

batch_size=64
logging_steps=len(emotions_encoded["train"])//batch_size

training_args=TrainingArguments(output_dir="results",num_train_epochs=
2,learning_rate=2e-
5,load_best_model_at_end=True,metric_for_best_model="f1",weight_decay=
0.01,evaluation_strategy="epoch",disable_tqdm=False,logging_steps=logg
ing_steps,save_strategy="epoch",)

import os
os.environ["CUDA_LAUNCH_BLOCKING"] = "1"
from transformers import Trainer
batch_size = 64
logging_steps = len(emotions_encoded["train"]) // batch_size
training_args = TrainingArguments(output_dir="results",
 num_train_epochs=2,
 learning_rate=2e-5,
 per_device_train_batch_size=batch_size,
 per_device_eval_batch_size=batch_size,
 load_best_model_at_end=True,
 metric_for_best_model="f1",
 weight_decay=0.01,
 evaluation_strategy="epoch",
 disable_tqdm=False,
 logging_steps=logging_steps,
 save_strategy="epoch"
                                  )

trainer = Trainer(model=model, args=training_args,
 compute_metrics=compute_metrics,
 train_dataset=emotions_encoded["train"],
 eval_dataset=emotions_encoded["validation"])
trainer.train();

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

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>


results=trainer.evaluate()

<IPython.core.display.HTML object>

results

{'eval_loss': 0.21765510737895966,
 'eval_accuracy': 0.9205,
 'eval_f1': 0.9204338577917631,
 'eval_runtime': 2.223,
 'eval_samples_per_second': 899.667,
 'eval_steps_per_second': 14.395,
 'epoch': 2.0}

preds_output=trainer.predict(emotions_encoded["validation"])

preds_output.metrics

{'test_loss': 0.21765510737895966,
 'test_accuracy': 0.9205,
 'test_f1': 0.9204338577917631,
 'test_runtime': 2.2561,
 'test_samples_per_second': 886.494,
 'test_steps_per_second': 14.184}

y_preds=np.argmax(preds_output.predictions,axis=1)

print(classification_report(y_valid,y_preds,target_names=labels))

              precision    recall  f1-score   support

     sadness       0.94      0.96      0.95       550
         joy       0.95      0.93      0.94       704
        love       0.83      0.89      0.86       178
       anger       0.93      0.92      0.92       275
        fear       0.85      0.86      0.86       212
    surprise       0.85      0.75      0.80        81

    accuracy                           0.92      2000
   macro avg       0.89      0.89      0.89      2000
weighted avg       0.92      0.92      0.92      2000


tweet="i saw a movie today and it was really good"
```

```
input_tensor=tokenizer.encode(tweet,return_tensors="pt").cuda()

logits=model(input_tensor).logits

logits
```

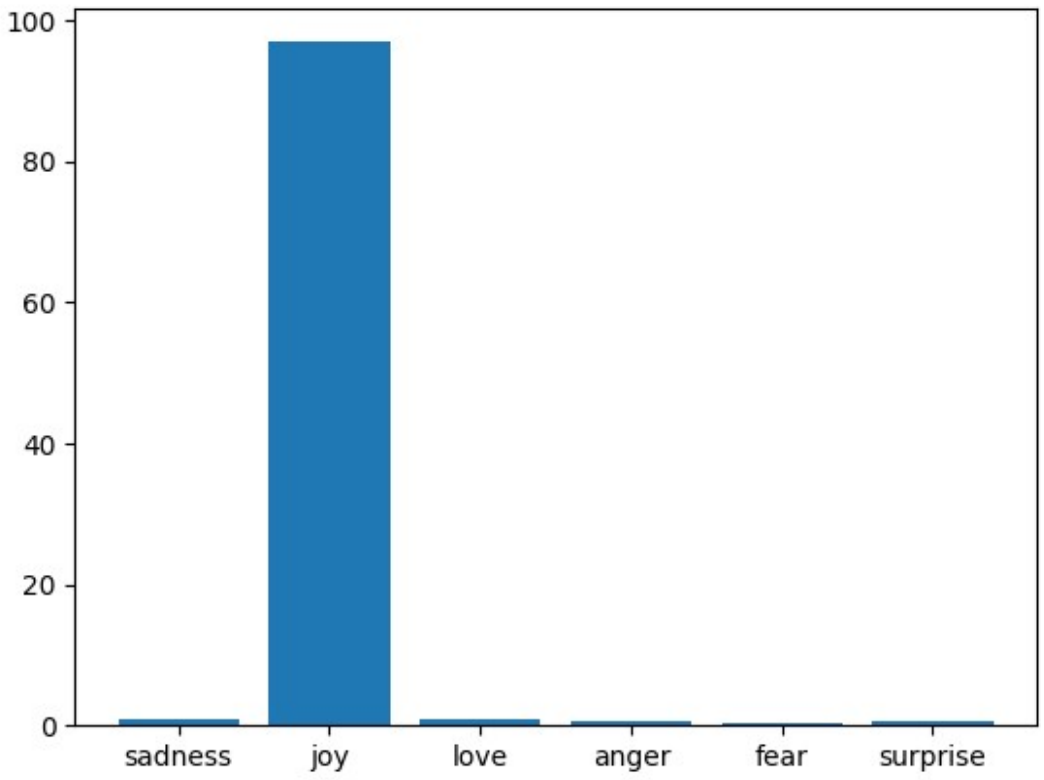
```
tensor([[-0.4387,  4.2708, -0.3444, -0.9805, -1.5487, -1.0088]],
       device='cuda:0', grad_fn=<AddmmBackward0>)
```

```
softmax=torch.nn.Softmax(dim=1)
probs=softmax(logits)[0]

probs=probs.cpu().detach().numpy()

probs
```

```
array([0.0087283 , 0.96879154, 0.00959088, 0.00507713, 0.00287652,
       0.00493561], dtype=float32)
```

```
plt.bar(labels,100*probs)
```

```
<BarContainer object of 6 artists>
```




```
from torch.nn.functional import cross_entropy

def forward_pass_with_label(batch):
    input_ids=torch.tensor(batch["input_ids"]).to("cuda")
```

```
    attention_mask=torch.tensor(batch["attention_mask"]).to("cuda")
    labels=torch.tensor(batch["label"]).to("cuda")
    with torch.no_grad():
        output=model(input_ids,attention_mask)
        pred_label=torch.argmax(output.logits,axis=-1)
        loss=cross_entropy(output.logits,labels,reduction="none")
    batch["predicted_label"]=pred_label.cpu().numpy()
    batch["loss"]=loss.cpu().numpy()
    return batch


emotions_encoded.reset_format()
emotions_encoded["validation"]=emotions_encoded["validation"].map(forw
ard_pass_with_label,batched=True,batch_size=16)
```


```
{"model_id":"d9b7f65f79a449a99091dddc6218868c","version_major":2,"vers
ion_minor":0}
```


```
emotions_encoded.set_format("pandas")
cols=["text","label","predicted_label","loss"]
df_test=emotions_encoded["validation"][:][cols]
df_test["label"]=df_test["label"].apply(label_int2str,split="test")

df["predicted_label"]=(df_test["predicted_label"].apply(label_int2str,
split="test"))

df_test.head(5)
```

	text	label	\
0	im feeling quite sad and sorry for myself but ...	sadness	
1	i feel like i am still looking at a blank canv...	sadness	
2	i feel like a faithful servant	love	
3	i am just feeling cranky and blue	anger	
4	i can have for a treat or if i am feeling festive	joy	

	predicted_label	loss
0	0	0.021657
1	0	0.022079
2	2	0.305429
3	3	0.030766
4	1	0.015522

```
trainer.save_model("models/distilbert-emotion")
tokenizer.save_pretrained("models/distilbert-emotion")
```

```
('models/distilbert-emotion/tokenizer_config.json',
 'models/distilbert-emotion/special_tokens_map.json',
 'models/distilbert-emotion/vocab.txt',
 'models/distilbert-emotion/added_tokens.json',
 'models/distilbert-emotion/tokenizer.json')
```

PREDICTION

```
from transformers import AutoTokenizer,
AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("RapidOrc121/BERt")
model =
AutoModelForSequenceClassification.from_pretrained("RapidOrc121/BERt")

tweet="I am feeling good today"
import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

input_tensor=tokenizer.encode(tweet,return_tensors="pt")
logits=model(input_tensor).logits
softmax=torch.nn.Softmax(dim=1)
probs=softmax(logits)[0]
probs=probs.cpu().detach().numpy()
probs
```

```
array([0.00667796, 0.97960085, 0.00664459, 0.00276055, 0.00209863,
       0.00221754], dtype=float32)
```