```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, f1_score, classification_report
import joblib
```

```
!pip install --upgrade --force-reinstall transformers==4.44.2
!pip install --upgrade --force-reinstall accelerate datasets
```

```python
import pandas as pd
from google.colab import files

# This opens a file picker in Colab → select your local CSV
uploaded = files.upload()

# Get the first uploaded filename
filename = list(uploaded.keys())[0]

# Read into pandas
df = pd.read_csv(filename)

# Quick sanity check
print("Rows:", len(df))
print("Columns:", df.columns.tolist())
print(df.head())
```

Choose files data.csv

data.csv(text/csv) - 79825 bytes, last modified: 23/09/2025 - 100% done

```
Saving data.csv to data.csv
Rows: 2129
Columns: ['reply', 'label']
                                         reply     label
0                       Can we discuss pricing??-  NEUTRAL
1  Im excited to explore this further, plz send c...  POSITIVE
2                   We not looking for new solutions.  negative
3                Could u clarify features included?-   neutral
4           lets,, schedule a meeting to dive deeper  positive
```

```
-none-any.whl.metadata (57 kB)
Collecting fsspec>=2023.5.0 (from huggingface-hub<1.0,>=0.23.2->transformers==4.44.2)
  Using cached fsspec-2025.9.0-py3-none-any.whl.metadata (10 kB)
Collecting typing-extensions>=3.7.4.3 (from huggingface-hub<1.0,>=0.23.2->transformers==4.44.2)
  Using cached typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/cli/base_command.py", line 179, in exc_logging_wrapp
    status = run_func(*args)
             ^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/cli/req_command.py", line 67, in wrapper
    return func(self, options, args)
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/commands/install.py", line 377, in run
```

```python
df = df.rename(columns={"reply": "text", "label": "label"})
df["text"] = df["text"].astype(str).str.strip().str.replace(r"\s+", " ", regex=True)
df["label"] = df["label"].str.lower().str.strip()

# Quick check
print("Rows:", len(df))
print("Columns:", df.columns.tolist())
print("\nLabel distribution:\n", df["label"].value_counts())
df.head()
```

```
Rows: 2129
Columns: ['text', 'label']

Label distribution:
 label
positive    710
negative    710
neutral     709
Name: count, dtype: int64
```

	text	label
0	Can we discuss pricing??	neutral
1	Im excited to explore this further, plz send c...	positive
2	We not looking for new solutions.	negative
3	Could u clarify features included?	neutral
4	lets,, schedule a meeting to dive deeper	positive

Next steps: Generate code with df New interactive sheet

```
  File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/resolvelib/resolvers.py", line 239, in _attempt_to_pin
    criteria = self._get_updated_criteria(candidate)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/resolvelib/resolvers.py", line 230, in _get_updated_cr
    self._add_to_criteria(criteria, requirement, parent=candidate)
  File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/resolvelib/resolvers.py", line 173, in _add_to_criteri
    if not criterion.candidates:
           ^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/resolvelib/structs.py", line 156, in __bool__
    return bool(self._sequence)
           ^^^^^^^^^^^^^^^^^^^^
                            es/pip/_internal/resolution/resolvelib/found_candidates.py", line 174
           ^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/resolution/resolvelib/found_candidates.py", line 162
                                                    in self._incompatible_ids)
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/resolution/resolvelib/found_candidates.py", line 49,
                                                         /resolvelib/factory.py", line 300, in iter
    result = self._finder.find_best_candidate(
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/index/package_finder.py", line 884, in find_best_can
```

```python
train_df, test_df = train_test_split(
    df, test_size=0.2, stratify=df['label'], random_state=42
)

print("Train size:", len(train_df), "Test size:", len(test_df))
```

```
Train size: 1703 Test size: 426
```

```
  File "/usr/local/lib/python3.12/dist-packages/pip/_internal/index/package_finder.py", line 792, in process_proje
    package_links = self.evaluate_links(
                    ^^^^^^^^^^^^^^^^^^^^
```

```python
pipeline = Pipeline([
    ("tfidf", TfidfVectorizer(
        lowercase=True,
        strip_accents="unicode",
        stop_words="english",
        ngram_range=(1, 2),
        max_features=20000
    )),
    ("clf", LogisticRegression(
        max_iter=200,
        solver="liblinear",
        random_state=42
    ))
])

param_grid = {"clf__C": [0.25, 0.5, 1.0, 2.0, 4.0]}
grid = GridSearchCV(
    pipeline, param_grid=param_grid,
    cv=5, n_jobs=-1, scoring="f1_macro", verbose=1
)

grid.fit(train_df['text'], train_df['label'])
```

```
                      ^^^^^^^^^^^^^^^^^^^^^^
    Fitting 5 folds for each of 5 candidates, totalling 25 fits
      File "/usr/local/lib/python3.12/dist-packages/pip/_internal/cli/base_command.py", line 100, in main
    ▸          GridSearchCV
    ▸   best_estimator_: Pipeline          /dist-packages/pip/_internal/cli/base_command.py", line 232, in _main
          ▸ TfidfVectorizer
                                           /dist-packages/pip/_internal/cli/base_command.py", line 215, in exc_logging_wrapp
        ▸ LogisticRegression               ncelled by user")
                                           ng/__init__.py", line 1586, in critical
                                           , **kwargs)
      File "/usr/lib/python3.12/logging/__init__.py", line 1684, in _log
        self.handle(record)
```

```
    y_pred = grid.predict(test_df['text'])
    print("Accuracy:", accuracy_score(test_df['label'], y_pred))
    print("Macro F1:", f1_score(test_df['label'], y_pred, average="macro"))
    print("\nClassification Report:\n", classification_report(test_df['label'], y_pred))
```

```
    Accuracy: 0.9882629107981221
    Macro F1: 0.9882781717888101

    Classification Report:
                   precision    recall  f1-score   support

        negative       0.99      0.98      0.99       142
         neutral       1.00      0.99      0.99       142
        positive       0.97      1.00      0.99       142

        accuracy                           0.99       426
       macro avg       0.99      0.99      0.99       426
    weighted avg       0.99      0.99      0.99       426
```

```
      File "/usr/local/lib/python3.12/dist-packages/pip/_internal/utils/logging.py", line 177, in emit
        self.console.print(renderable, overflow="ignore", crop=False, style=style)
      File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/rich/console.py", line 1674, in print
        renderables = self._collect_renderables(
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^
      File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/rich/console.py", line 1553, in _collect_renderables
        check_text()
      File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/rich/console.py", line 1531, in check_text
        append(sep_text.join(text))
               ^^^^^^^^^^^^^^^^^^^^
      File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/rich/text.py", line 803, in join
        for text in iter_text():
                    ^^^^^^^^^^^
      File "/usr/local/lib/python3.12/dist-packages/pip/_vendor/rich/text.py", line 790, in iter_text
        for last, line in loop_last(lines):
                          ^^^^^^^^^^^^^^^^
```

```
    import joblib
    joblib.dump(grid.best_estimator_, "baseline_model.joblib")
```

```
    ['baseline_model.joblib']
    ---------------------------------------------------------------------------
    KeyboardInterrupt                         Traceback (most recent call last)
```

```
    from google.colab import files
    files.download("baseline_model.joblib")
```

```
                                   11 frames
    /usr/lib/python3.12/pathlib.py in stat(self, follow_symlinks)
        838         os.stat() does.
```

```
    !pip install -q transformers datasets accelerate evaluate
```

```
        842     def lstat(self):
```

```
    !pip install -q --upgrade transformers
```

```
    import pandas as pd
    import numpy as np
    from sklearn.model_selection import train_test_split

    import torch
    from datasets import Dataset
    from transformers import (
        AutoTokenizer,
        AutoModelForSequenceClassification,
        TrainingArguments,
        Trainer,
        DataCollatorWithPadding,
    )
```

```
# Ensure columns are "text" and "label" with lowercase labels
df["label"] = df["label"].str.lower().str.strip()

# Train / validation split
train_df, val_df = train_test_split(
    df, test_size=0.2, stratify=df["label"], random_state=42
)

# Hugging Face dataset objects
train_ds = Dataset.from_pandas(train_df)
val_ds = Dataset.from_pandas(val_df)

# Label mapping
labels = ["negative", "neutral", "positive"]
label2id = {l: i for i, l in enumerate(labels)}
id2label = {i: l for l, i in label2id.items()}

def encode_labels(example):
    example["labels"] = label2id[example["label"]]
    return example

train_ds = train_ds.map(encode_labels)
val_ds = val_ds.map(encode_labels)
```

```
Map: 100%                                1703/1703 [00:00<00:00, 4436.51 examples/s]

Map: 100%                                426/426 [00:00<00:00, 4525.41 examples/s]
```

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

def tokenize(examples):
    return tokenizer(examples["text"], truncation=True, padding=False, max_length=256)

train_ds = train_ds.map(tokenize, batched=True)
val_ds = val_ds.map(tokenize, batched=True)

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

```
/usr/local/lib/python3.12/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `clean_up_toker
  warnings.warn(
Map: 100%                                1703/1703 [00:00<00:00, 7200.24 examples/s]

Map: 100%                                426/426 [00:00<00:00, 4602.28 examples/s]
```

```
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=len(labels),
    id2label=id2label,
    label2id=label2id,
)
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-bas
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
from sklearn.metrics import accuracy_score, f1_score

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=-1)
    acc = accuracy_score(labels, preds)
    f1 = f1_score(labels, preds, average="macro")
    return {"accuracy": acc, "f1": f1}
```

```
from transformers import TrainingArguments
training_args = TrainingArguments(
    output_dir="./distilbert-reply-clf",
    evaluation_strategy="epoch",   # or "steps"
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=4,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    greater_is_better=True,
    logging_dir="./logs",
    logging_steps=20,
)
```

```
/usr/local/lib/python3.12/dist-packages/transformers/training_args.py:1525: FutureWarning: `evaluation_strategy` is
  warnings.warn(
```

```
!pip install --upgrade transformers
```

```
import transformers
print(transformers.__version__)
```

```
from datasets import Dataset
from sklearn.model_selection import train_test_split

# 1. Normalize labels
df["label"] = df["label"].str.lower().str.strip()

# 2. Train/validation split
train_df, val_df = train_test_split(
    df, test_size=0.2, stratify=df["label"], random_state=42
)

# 3. Define mappings
labels = ["negative", "neutral", "positive"]
label2id = {l: i for i, l in enumerate(labels)}
id2label = {i: l for l, i in label2id.items()}

# 4. Convert pandas → Dataset, keeping only needed columns
train_ds = Dataset.from_pandas(train_df[["text", "label"]].reset_index(drop=True))
val_ds = Dataset.from_pandas(val_df[["text", "label"]].reset_index(drop=True))

# 5. Encode labels → integers
def encode_labels(example):
    return {"labels": label2id[example["label"]]}

train_ds = train_ds.map(encode_labels)
val_ds = val_ds.map(encode_labels)

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

def tokenize(batch):
    return tokenizer(batch["text"], truncation=True, padding=True, max_length=256)

train_ds = train_ds.map(tokenize, batched=True)
val_ds = val_ds.map(tokenize, batched=True)

# 7. Remove original string label column
train_ds = train_ds.remove_columns(["label"])
val_ds = val_ds.remove_columns(["label"])

# 8. Verify
print(train_ds[0])
```

```
Map: 100%                                   1703/1703 [00:00<00:00, 9920.73 examples/s]

Map: 100%                                   426/426 [00:00<00:00, 4857.16 examples/s]
/usr/local/lib/python3.12/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `clean_up_toke
  warnings.warn(
Map: 100%                                   1703/1703 [00:00<00:00, 6723.83 examples/s]

Map: 100%                                   426/426 [00:00<00:00, 3659.75 examples/s]
{'text': 'Please share the details, I’m interested.', 'labels': 2, 'input_ids': [101, 3531, 3745, 1996, 4751, 1010,
```

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=len(labels),
    id2label=id2label,
    label2id=label2id,
)
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-bas
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="./distilbert-reply-clf",
    eval_strategy="epoch",        # use "eval_strategy" (new name)
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=4,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    greater_is_better=True,
    logging_dir="./logs",
    logging_steps=20,
    report_to="none",             # ✅ turn off wandb prompt
)
```

```
from sklearn.metrics import accuracy_score, f1_score

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = logits.argmax(axis=-1)
    acc = accuracy_score(labels, preds)
    f1 = f1_score(labels, preds, average="macro")
    return {"accuracy": acc, "f1": f1}
```

```
from transformers import Trainer, DataCollatorWithPadding

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_ds,
    eval_dataset=val_ds,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)
```

```
trainer.train()
```

```
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:666: UserWarning: 'pin_memory' argument is se
  warnings.warn(warn_msg)
```

[428/428 18:11, Epoch 4/4]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.024900	0.025807	0.995305	0.995305
2	0.006000	0.004189	1.000000	1.000000
3	0.003800	0.002724	1.000000	1.000000
4	0.003100	0.002383	1.000000	1.000000

```
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:666: UserWarning: 'pin_memory' argument is se
  warnings.warn(warn_msg)
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:666: UserWarning: 'pin_memory' argument is se
  warnings.warn(warn_msg)
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:666: UserWarning: 'pin_memory' argument is se
  warnings.warn(warn_msg)
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:666: UserWarning: 'pin_memory' argument is se
  warnings.warn(warn_msg)
TrainOutput(global_step=428, training_loss=0.07845190939467366, metrics={'train_runtime': 1104.7997,
'train_samples_per_second': 6.166, 'train_steps_per_second': 0.387, 'total_flos': 24674562830736.0, 'train_loss':
```