# AI & ML Workshop: Text Classifier

Workshop Manual

## Workshop Overview

In this hands-on workshop, you will build a simple machine learning model that classifies short text reviews, specifically *Yelp* reviews, as **positive** or **negative**.
Estimated Time: **1.5 - 2 hours**

You'll use `Python` in `Google Colab to`:

- Prepare and explore a text dataset.
- Clean and organize the data for training.
- Split examples into training and testing sets.
- Convert text into numerical features.
- Train a sentiment classification model.
- Evaluate how well the model performs.
- Experiment with your own review inputs.
- Save your trained model for reuse.

## Getting Started

**Please Read!**

**Important Reminder:** To qualify for your **reward**, you must complete the entire workshop and carefully follow all instructions. Every reflection in the `Google Colab` must include a thoughtful response of at least **50 words**. Finally, make sure to complete the **Final Reflection** in the `Google Form` before submitting your work. Incomplete reflections or skipped steps may result in ineligibility for the reward.

Now, let us get started with this workshop! There are files that you need to download to your computer in order to continue. First, you need to get the `.ipynb` file for the `Google Colab`. One more thing that you will need is the dataset from *Yelp*.

**Instruction**

Navigate to our official website, csed-research-lab.org. On the navigation bar, click `Projects` and you can find the link to the `Google Colab`. Click the link and make a copy of this `Google Colab`.

**Instruction**

Below the link to the `Google Colab`, click the button to download the **CSV** file. You will need it for later in the workshop.

## Step 0: Colab Setup

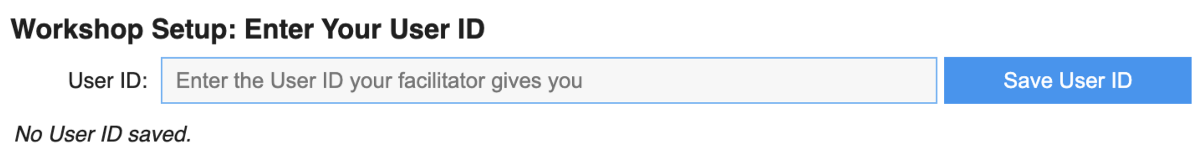
Before diving into building your model, let's get your workspace ready in Google Colab.

**Please Read!**

**Important Reminder:** Always click the **triangle play button** to run the cell *before* typing anything in the textbox. If you type first and then click the play button, your text will disappear, and you'll need to re-enter it. Running the cell first makes sure your reflection box works properly and saves your response when you click **Submit Reflection**.

**Instruction**

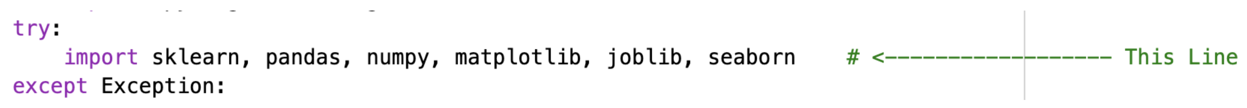
Run the cell labeled **Step 0.1: Setup (run once)** and **Step 0.2: Reflection helpers (run once)**. Enter your User ID and click the blue button **Save User ID**.


Workshop Setup: Enter Your User ID
User ID:
Enter the User ID your facilitator gives you
Save User ID
No User ID saved.

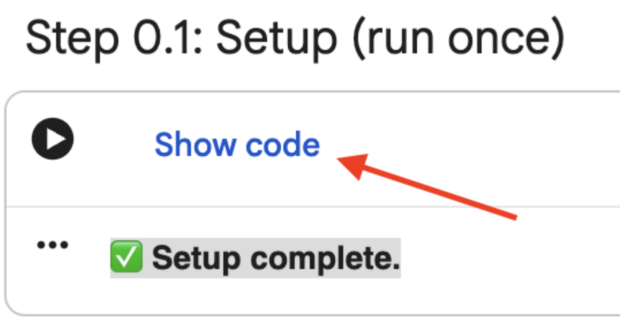
**Instruction**

Run the cell **Step 0.3** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on. *Notes:*

1. *Highlighted lines have arrows like this* `# <------------` .
2. *Click Show code for the cell to expand.*



```
try:
    import sklearn, pandas, numpy, matplotlib, joblib, seaborn    # <------------------ This Line
except Exception:
```



Step 0.1: Setup (run once)

```
✅ Setup complete.
```

**Reflect**

Take a look at **Step 0.1**. Based on the import statements you've seen in the highlighted portion of **Step 0.1**, what do you think the code is doing? Which parts look familiar or new?

**Reveal**

This setup cell does the following:

1. Detects if the notebook is running in Google Colab and enables the custom widget manager.
2. Installs `ipywidgets` if it isn't already available, allowing interactive elements like buttons and sliders.
3. Tries to import key libraries for data processing and machine learning (`scikit-learn`, `pandas`, `numpy`, `matplotlib`, `joblib`, `seaborn`); if any are missing, it installs them automatically.
4. Displays a short confirmation message (`Setup complete`) once everything is ready.

## Step 1: Loading and Cleaning Yelp data

Now that your setup is ready, it's time to meet your data — the foundation of every machine learning project!

**Instruction**

Perfect! Moving on to **Step 1.1** and **Step 1.2**. Run these cells and fill in your reflections. Remember to click the green button **Submit Reflection** before moving on.

**Reflect**

After you download the **CSV** file, open it with **Excel**. What do you see inside? Now, open it with **Notepad**. Now what do you see? What do you think is happening?
If you had to classify a review as positive or negative yourself, what words or phrases would you look for?

**Instruction**

Examine then run the code inside **Step 1.4** and **Step 1.5**, which starts out like the code below. Use the **CSV** file you downloaded at the beginning of the workshop to upload.

```
#@title Step 1.4: Upload or Use Sample Data
import pandas as pd, io
from IPython.display import display
try:
    from google.colab import files
    USING_COLAB = True
...
#@title Step 1.5: Cleaning the dataset
import pandas as pd
valid = {'positive','negative','pos','neg','1','0','true','false'}
df = df.rename(columns={c:c.lower() for c in df.columns})
...
```

**Reveal**

These cells handle loading and cleaning the Yelp review dataset:

1. Attempt to load a CSV file named `yelp_reviews.csv` that contains two columns: `text` and `sentiment`. If no file is uploaded, a small built-in sample dataset is used instead.
2. Convert all column names to lowercase and verify that the required columns exist.
3. Clean the text by removing extra spaces and empty rows to ensure each review is valid.
4. Normalize sentiment labels by converting different forms (`pos`/`1`/`true` → `positive`; `neg`/`0`/`false` → `negative`).
5. Remove any duplicate reviews to prevent bias in training.
6. Finally, print the count of positive and negative examples after cleaning.

## Step 2: Split train/test

In this step, you'll explore a key idea in machine learning — how models learn and how we test what they've learned.

**Instruction**

Examine then run the code in **Step 2.1** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

```
#@title Step 2.1: Train/test split
import numpy as np
from sklearn.model_selection import train_test_split
...
```

**Reflect**

What do you think this code is doing? Why do you think we split the data into two groups?

**Reveal**

This step prepares the data for model training and evaluation:

1. Converts the review text (`text`) and sentiment labels (`sentiment`) into NumPy arrays, a standard format for machine learning tasks.
2. Uses `train_test_split` to divide the dataset into two parts:
   - **Training set (80%)** — used by the model to learn patterns.
   - **Testing set (20%)** — held out to evaluate how well the model performs on unseen data.
3. The `stratify=y` option ensures that both positive and negative labels are evenly represented in each split.
4. Finally, it prints how many samples are in the training and testing groups to confirm that the split worked correctly.

## Step 3: TF-IDF (Term Frequency–Inverse Document Frequency)

In this step, you'll discover how computers can understand raw text and make machine learning with language possible.

**Instruction**

Examine then run the code in **Step 3.1** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

```
#@title Step 3.1: TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
...
```

**Reflect**

What do you think TF-IDF is doing? What do you think 'Term Frequency' and 'Inverse Document Frequency' might mean?

**Reveal**

This step converts text into numerical features using TF–IDF so the model can learn from it:

1. Creates a `TfidfVectorizer` that analyzes single words (unigrams) and pairs of words (bigrams), while removing common English stop words.
2. The vectorizer first **fits** on the training data — it learns the vocabulary and calculates how important each term is relative to the dataset.
3. Then, it **transforms** both the training and testing text into numeric feature matrices, keeping the same learned vocabulary for both.
4. The printed shapes show how many reviews (rows) and word features (columns) were created.

## Step 4: Training

Now it's time for the exciting part — teaching your model how to think!

**Instruction**

Examine then run the code in **Step 4.1** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

**Reflect**

How is using a trained model to make predictions different from writing explicit rules to classify text?

**Instruction**

Run the code in **Step 4.2** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

```
#@title Step 4.2: Fit classifiers
from sklearn.svm import LinearSVC
...
```

**Reflect**

What do you think happens when we call .fit()? What do you think the model is learning?

**Reveal**

This step trains a machine learning model to recognize sentiment patterns in the text:

1. Imports the `LinearSVC` classifier — a linear Support Vector Machine often used for text classification tasks.
2. Creates an instance of the classifier and assigns it to `clf`.
3. Calls `clf.fit(X_train_tfidf, y_train)` to train the model using the TF–IDF features and their corresponding sentiment labels.
4. During training, the model learns to separate positive and negative reviews by finding patterns in the weighted word features.
5. Finally, it prints a confirmation message once training is complete.

## Step 5: Evaluate your model(s)

You've trained your model — now it's time to see how well it actually performs!

**Instruction**

Examine then run the code in **Step 5.1** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

```
#@title Step 5.1: Evaluation
from sklearn.metrics import accuracy_score, classification_report,
    ↪ confusion_matrix
import matplotlib.pyplot as plt, numpy as np, itertools
import seaborn as sns
...
```

**Reflect**

What do you think is happening in Step 5.1? Why is it necessary to evaluate our model?

**Reveal**

This step measures how well the trained model performs on unseen test data:

1. Imports tools from `sklearn.metrics` to calculate key evaluation measures like accuracy, precision, recall, and F1-score.
2. Uses the trained model to make predictions on the test set with `clf.predict(X_test_tfidf)`.
3. Computes and prints the overall **accuracy** and a detailed **classification report**, which breaks down performance for positive and negative classes.
4. Defines a helper function `plot_cm()` to visualize the confusion matrix using `seaborn.heatmap`. This shows where the model predicted correctly and where it made mistakes.
5. Displays the confusion matrix labeled with "positive" and "negative" axes for easy interpretation.

## Step 6. Try it

You did it — your model is built, trained, and ready to take the stage! Now it's your turn to see how smart your AI really is.

**Instruction**

Congrats on reaching this step. This means that you have finished building your own model. Now, let's have some fun testing your own model. Run **Step 6.1: Predict your own review**, write a sample Yelp review, then click the button **Predict**. After that, run **Step 6.2** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

**Reflect**

What was the review you wrote. Was it classified correctly when you hit Predict?

## Step 7: Save the model

You've built, trained, and tested your sentiment model — now it's time to make sure your hard work doesn't disappear!

**Instruction**

Examine then run the code in **Step 7.1** and fill in your reflection. Remember to click the green button **Submit Reflection** before moving on.

```
#@title Step 7.1: Save model
import joblib
```

```
...
```

**Reflect**

Why do you think it is important to save our model?

**Reveal**

This step saves your trained machine learning model so it can be reused later:

1. Imports the `joblib` library, which is commonly used for saving and loading large Python objects such as models.
2. Packages both the trained classifier (`clf`) and the TF–IDF vectorizer into a single dictionary.
3. Saves this dictionary to a file named `yelp_sentiment.joblib`.
4. Prints a confirmation message once the model has been successfully stored.
5. This allows you to reload your trained model later without retraining, making it useful for future testing or deployment.

## Step 8: Final Reflection

Great job — you've made it to the final step of the workshop! All that's left is to complete the **Final Reflection** boxes in the Google Colab.

**Please Read!**

Before you finish, take a few minutes to **double-check your work**. Make sure every reflection box has a complete and thoughtful answer, and that you've clicked **Submit Reflection** . Once you've confirmed everything is done, please **notify the workshop assistant** so your completion can be verified and your reward can be processed. Don't leave without letting them know — this is your final step!