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

Choose Files yelp_labelled.txt

- **yelp_labelled.txt**(text/plain) - 61320 bytes, last modified: 4/28/2025 - 100% done

```
Saving yelp_labelled.txt to yelp_labelled.txt
```

```
import pandas as pd

filename = next(iter(uploaded))
data = pd.read_csv(filename, delimiter='\t', header=None, names=['sentence', 'label'])

data.head()
```

	sentence	label
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1

Next steps: Generate code with data View recommended plots New interactive sheet

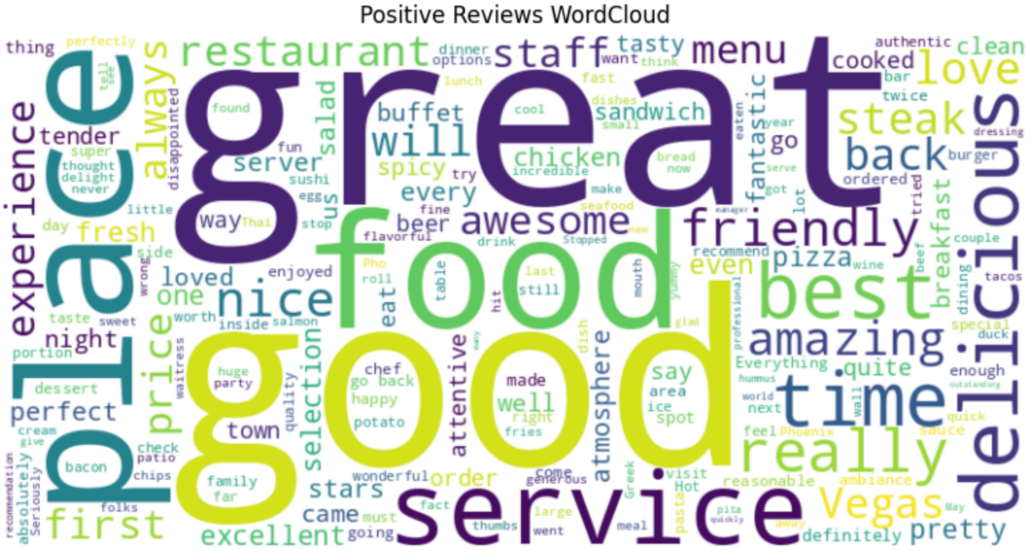
```
from sklearn.model_selection import train_test_split

sentence_train, sentence_test, y_train, y_test = train_test_split(
    data['sentence'], data['label'], test_size=0.2, random_state=42)

print(f"Train set: {len(sentence_train)} samples")
print(f"Test set: {len(sentence_test)} samples")
```

```
Train set: 800 samples
Test set: 200 samples
```

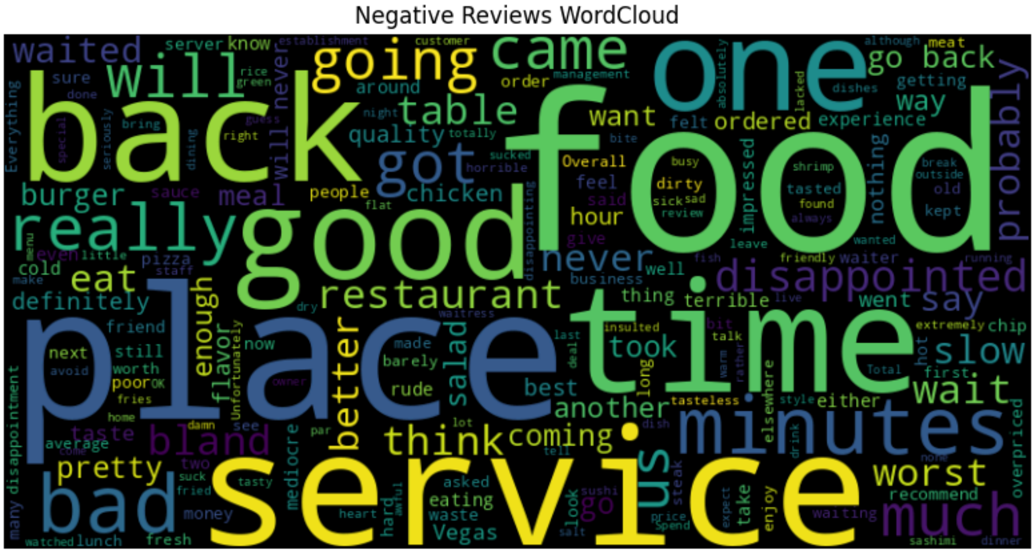
```
print("Training set distribution:\\n", y_train.value_counts())
print("Testing set distribution:\\n", y_test.value_counts())
```

```
Training set distribution:\n label
0    404
1    396
Name: count, dtype: int64
Testing set distribution:\n label
1    104
0     96
Name: count, dtype: int64
```

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

positive_text = ' '.join(sentence_train[y_train == 1])
wordcloud_pos = WordCloud(width=800, height=400, background_color='white').generate(positive_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_pos, interpolation='bilinear')
plt.axis('off')
plt.title('Positive Reviews WordCloud')
plt.show()
```



```
negative_text = ' '.join(sentence_train[y_train == 0])
wordcloud_neg = WordCloud(width=800, height=400, background_color='black').generate(negative_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_neg, interpolation='bilinear')
plt.axis('off')
plt.title('Negative Reviews WordCloud')
plt.show()
```



```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(sentence_train)
X_test = vectorizer.transform(sentence_test)

print(f"Feature matrix shape: {X_train.shape}")
```

```
Feature matrix shape: (800, 1794)
```

```python
bow_data = pd.DataFrame(X_train.toarray(), columns=vectorizer.get_feature_names_out())
bow_data['label'] = y_train.values
bow_data = pd.DataFrame(X_train.toarray(), columns=vectorizer.get_feature_names_out())
bow_data['label'] = y_train.values

bow_data.to_csv('Homework01_BoW.csv', index=False)
print("BoW data saved as Homework01_BoW.csv")
```

```
BoW data saved as Homework01_BoW.csv
```

```python
from sklearn.neighbors import KNeighborsClassifier
import joblib

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
from sklearn.neighbors import KNeighborsClassifier
import joblib

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)

joblib.dump(knn_model, 'knn_model.pkl')
print("KNN Model Saved!")
```

```
KNN Model Saved!
```

```python
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train, y_train)
from sklearn.linear_model import LogisticRegression
import joblib

logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train, y_train)

joblib.dump(logistic_model, 'logistic_model.pkl')
print("Logistic Regression Model Saved!")
```

```
Logistic Regression Model Saved!
```

```python
from sklearn.svm import SVC

svm_model = SVC(probability=True)
svm_model.fit(X_train, y_train)
from sklearn.svm import SVC

svm_model = SVC(probability=True)
svm_model.fit(X_train, y_train)

import joblib
joblib.dump(svm_model, 'svm_model.pkl')
print("SVM Model Saved!")
```

```
SVM Model Saved!
```

```python
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
from sklearn.ensemble import RandomForestClassifier
import joblib

rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

joblib.dump(rf_model, 'rf_model.pkl')
print("Random Forest Model Saved!")
```

```
Random Forest Model Saved!
```

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

def evaluate_model(model, X, y):
    y_pred = model.predict(X)
    y_prob = model.predict_proba(X)[:,1] if hasattr(model, "predict_proba") else model.decision_function(X)
    return {
        'Accuracy': accuracy_score(y, y_pred),
        'Precision': precision_score(y, y_pred),
        'Recall': recall_score(y, y_pred),
        'F1': f1_score(y, y_pred),
        'ROC AUC': roc_auc_score(y, y_prob)
    }

models = {
    'KNN': joblib.load('knn_model.pkl'),
    'Logistic Regression': joblib.load('logistic_model.pkl'),
    'SVM': joblib.load('svm_model.pkl'),
    'Random Forest': joblib.load('rf_model.pkl')
}

results = {name: evaluate_model(model, X_test, y_test) for name, model in models.items()}
results = pd.DataFrame(results).T
results
```

	Accuracy	Precision	Recall	F1	ROC AUC
KNN	0.650	0.644068	0.730769	0.684685	0.658654
Logistic Regression	0.810	0.866667	0.750000	0.804124	0.873397
SVM	0.745	0.778947	0.711538	0.743719	0.830429
Random Forest	0.785	0.850575	0.711538	0.774869	0.868389

Next steps: Generate code with results | View recommended plots | New interactive sheet

```
from sklearn.model_selection import GridSearchCV

param_grid = {'C': [0.01, 0.1, 1, 10, 100]}
grid = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=10)
grid.fit(X_train, y_train)
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

param_grid = {'C': [0.01, 0.1, 1, 10, 100]}
grid = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=10)
grid.fit(X_train, y_train)

best_logistic = grid.best_estimator_
print(f"Best Logistic Regression C: {grid.best_params_}")
```

```
Best Logistic Regression C: {'C': 10}
```

```
import pandas as pd
import joblib
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

# Load your dataset
df = pd.read_csv('yelp_labelled.txt', sep='\t', names=['sentence', 'label'])

# Split the data
sentences_train, sentences_test, y_train, y_test = train_test_split(
    df['sentence'], df['label'], test_size=0.25, random_state=42
)

# Refit your original vectorizer
vectorizer = CountVectorizer(min_df=0.0, lowercase=False, stop_words='english')
vectorizer.fit(sentences_train)

# Save it as vectorizer.pkl
joblib.dump(vectorizer, 'vectorizer.pkl')
print(" Saved vectorizer.pkl")
```

```
 Saved vectorizer.pkl
```

```
results_data = {
    'Methods': list(results.index),
    'Accuracy': results['Accuracy'].values,
    'Precision': results['Precision'].values,
    'Recall': results['Recall'].values,
    'F1-score': results['F1'].values,
    'AUC score': results['ROC AUC'].values
}

evaluation_summary = pd.DataFrame(results_data)
evaluation_summary
```

	Methods	Accuracy	Precision	Recall	F1-score	AUC score
0	KNN	0.650	0.644068	0.730769	0.684685	0.658654
1	Logistic Regression	0.810	0.866667	0.750000	0.804124	0.873397
2	SVM	0.745	0.778947	0.711538	0.743719	0.830429
3	Random Forest	0.785	0.850575	0.711538	0.774869	0.868389

Next steps: Generate code with evaluation_summary | View recommended plots | New interactive sheet

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer
import joblib

df = pd.read_csv("yelp_labelled.txt", sep='\t', names=["sentence", "label"])

X_train_text, X_test_text, y_train, y_test = train_test_split(
    df["sentence"], df["label"], test_size=0.2, random_state=42)

vectorizer = CountVectorizer(stop_words='english')
X_train = vectorizer.fit_transform(X_train_text)

model = LogisticRegression()
model.fit(X_train, y_train)

joblib.dump(vectorizer, "vectorizer.pkl")
joblib.dump(model, "logistic_model.pkl")

print("Saved: vectorizer.pkl and logistic_model.pkl")
```

```
Saved: vectorizer.pkl and logistic_model.pkl
```

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.28.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.0 (from gradio)
  Downloading gradio_client-1.10.0-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.30.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.16)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.3)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.9.3 (from gradio)
```

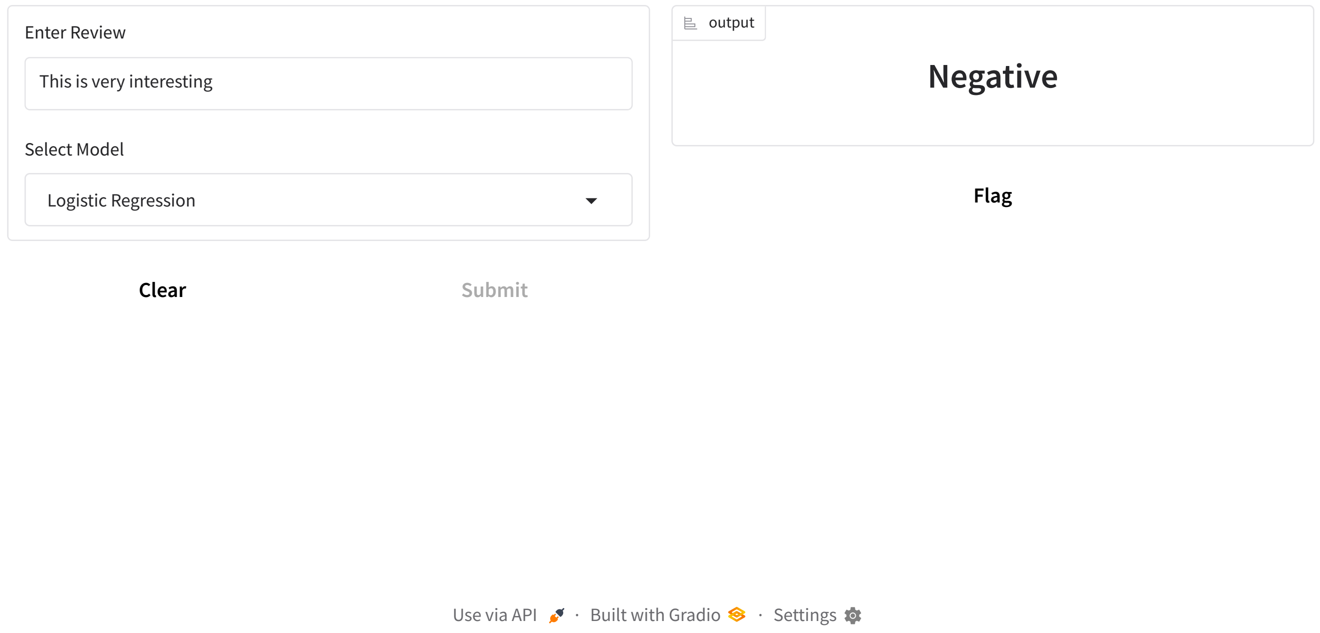
```
      Downloading ruff-0.11.7-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
    Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
      Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
    Collecting semantic-version~=2.0 (from gradio)
      Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
    Collecting starlette<1.0,>=0.40.0 (from gradio)
      Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
    Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
      Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
    Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.2)
    Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
    Collecting uvicorn>=0.14.0 (from gradio)
      Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
    Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (2025.
    Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0
    Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
    Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1
    Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.1.31)
    Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
    Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->grad
    Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3
    Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2
    Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio
    Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gra
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025
    Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (20
    Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->
    Requirement already satisfied: pydantic-core==2.33.1 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->g
    Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0
```

```
loaded_vec = joblib.load("vectorizer.pkl")
loaded_model = joblib.load("logistic_model.pkl")

def test_sentiment(text):
    vec = loaded_vec.transform([text])
    prob = loaded_model.predict_proba(vec)[0]
    return {
        "Prediction": "Positive" if prob[1] > 0.5 else "Negative",
        "Positive Probability": round(prob[1], 3),
        "Negative Probability": round(prob[0], 3)
    }

test_sentiment("This class is amazing")
```

```
{'Prediction': 'Positive',
 'Positive Probability': np.float64(0.751),
 'Negative Probability': np.float64(0.249)}
```

```
import gradio as gr
import joblib
import numpy as np

knn_model = joblib.load("knn_model.pkl")
logistic_model = joblib.load("logistic_model.pkl")
svm_model = joblib.load("svm_model.pkl")
rf_model = joblib.load("rf_model.pkl")

vectorizer = joblib.load("vectorizer.pkl")

models = {
    'KNN': knn_model,
    'Logistic Regression': logistic_model,
    'SVM': svm_model,
    'Random Forest': rf_model
}

def predict_sentiment(text, model_name):
    vec = vectorizer.transform([text])
    model = models[model_name]
    pred = model.predict(vec)
    return 'Positive' if pred[0] == 1 else 'Negative'

gr.Interface(
    fn=predict_sentiment,
    inputs=[
        gr.Textbox(label="Enter Review"),
        gr.Dropdown(list(models.keys()), label="Select Model")
    ],
    outputs=gr.Label()
).launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Au

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://8cdf7fd88f4d48c68a.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the
```



Enter Review

This is very interesting

Select Model

Logistic Regression

output

Negative

The link for the Hugging face : https://huggingface.co/spaces/nchenchu/Homework_01_Deploy_ML_model