



**Institute for Advanced Studies  
in Basic Sciences  
GavaZang, Zanjan, Iran**

# **Sentiment Analysis Model Loader**

---

**Prof. Majid Ramezani**

**Report for a Machine Learning Project <1>**

**Faeze Ahmadi**

**14044141**

**November 2025**

## تحلیل احساسات متن (Sentiment Analysis)

کد کامل این پروژه و فایل‌های مربوطه در ریپازیتوري گیت‌هاب قابل دسترسی هست:

<https://github.com/Faeze-Ahmadi/sentiment-analysis-project>

این پروژه یک سیستم ساده تحلیل احساسات (Sentiment Analysis) برای تشخیص مثبت یا منفی بودن متنی هست که از کاربر می‌گیریم. ایده اصلی این پروژه اینه که به کامپیوتر یاد بدیم محتوای یک متن را بخونه و تشخیص بده که متن حس مثبت داره یا منفی؛ مثلاً این متن می‌تونه نظری در مورد فیلم باشه. البته که دیتاست ما هم کامنت‌های مربوط به فیلم و استوری هست. یعنی ما با داشتن حجم زیادی از نظرهای مثبت و منفی و نشون دادن اونها به مدل‌مون آموزش می‌دیم که چی مثبته و چی منفیه، تا وقتی کاربر بهش یه نظر داد، مدل ما خودش پیش‌بینی کنه که اون نظر مثبته یا منفی.

این پروژه دو بخش اصلی دارد:

1. تبدیل متن خام به اعداد قابل فهم برای ماشین (به کمک TF-IDF): چون کامپیوتر نمی‌تونه متن را بخونه و بخارطه همین ما باید متن‌ها رو به عدد تبدیل کنیم و اینطوری مدل‌مون رو آموزش بدیم. Term Frequency – Inverse Document (TF-IDF مخفف عبارت) (فراآنی اصطلاح Frequency) هست که به عنوان یک روش معروف برای تبدیل متن به عدد شناخته می‌شه. Term Frequency (فراآنی اصطلاح) (معکوس فراآنی متن) یعنی این کلمه بین همه متن‌ها چندبار داخل متن تکرار شده و Inverse Document Frequency یعنی این کلمه بین همه متن‌ها چقدر خاص یا کمیاب هست.

2. آموزش یک مدل Machine Learning برای تشخیص احساسات متن‌ها: بحث آموزش دادن هم بر اساس برچسب زدن به متن‌هاست. ما از مدل Logistic Regression برای این کار استفاده می‌کنیم. این مدل، برای classification هست. در واقع ما نیاز داریم که اینجا 0 و 1 ای فکر کنیم، یعنی اگر مدل مثبته پس 1 هست و اگر مدل منفیه پس 0 هست. اما بحث اینجاست که توی متن ما به تحلیل نیاز داریم، پس مدل رگرسیون لجستیک می‌دانیم که متن به دسته 1 تعلق داره یا 0؟.. اینجاست که ما متن یه عدد رو نسبت می‌دیم، مثلاً می‌گیم متن به اندازه 0.87 مثبته؛ پس چون به متن  $< 0.5$  هست، مثبته. حالا در ادامه بیشتر در مورد اینا صحبت می‌کنیم.

بعد از آموزش مدل، خود مدل و ابزار تبدیل متن رو ذخیره می‌کنیم تا بتونیم بعداً بدون نیاز به آموزش دوباره، متن‌های جدید رو پیش‌بینی کنیم. یعنی ما یکبار دیتاست رو به مدل می‌دیم، اون رو آموزش می‌دیم و بعداً دیگه فقط از مدل استفاده می‌کنیم. چون حجم دیتاها زیاده و آموزش هریاره، هم به زمان و هم به فضا نیاز داره و اصلاً هم کار حرفه‌ای نیست.

## ساختار پروژه

### /data فولدر

این بخش شامل داده‌های اصلی پروژه هست. دو فولدر جداگونه برای نظرات مثبت و منفی وجود دارند. جمما 25 هزار متن کوتاه درباره فیلم‌ها در این قسمت قرار داده شده (12,500 متن مثبت و 12,500 متن منفی) و مدل با استفاده از همین داده‌ها آموزش می‌بینه.

← شامل متن‌های منفی /neg

← شامل متن‌های مثبت /pos

### model.pkl فایل

توی این فایل، مدل آموزش‌داده شده ذخیره شده.

این مدل همون چیزیه که بعد از آموزش می‌تونه تشخیص بد که متن مثبته یا منفی.

### vectorizer.pkl فایل

توی این فایل، TF-IDF vectorizer ذخیره شده.

این ابزار وظیفه داره متن خام رو به یه بردار عددی تبدیل کنه تا مدل اون رو بفهمه. بدون این فایل، مدل نمی‌تونه متن جدید رو تحلیل کنه.

### /src فولدر

توی این فولدر تمام فایل‌های مربوط به منطق برنامه وجود دارند. هر فایل یه وظیفه مشخص داره.

• فایل load\_data.py

وظیفه این فایل خوندن داده‌ها از فولدر `data` و برچسب‌گذاری آنهاست.

به زیان ساده، این فایل متن‌ها رو می‌خونه و مشخص می‌کنه که کدومشون مثبت و کدومشون منفی هستن.

• فایل vectorize.py

توی این فایل متون خام به مقادیر عددی تبدیل می‌شن.

این مرحله شامل ساخت TF-IDF vectorizer و تقسیم داده‌ها به train و test هست یعنی این فایل متن رو به چیزی که مدل بتونه پردازش کنه تبدیل می‌کنه.

- **train\_model.py**

این فایل مدل ماشین لرنینگ رو آموزش می‌دهد. مدلی که استفاده شده Logistic Regression هست. اینجا مدل با استفاده از داده‌ها یاد می‌گیره که چطور احساس یک متن رو تشخیص بده.

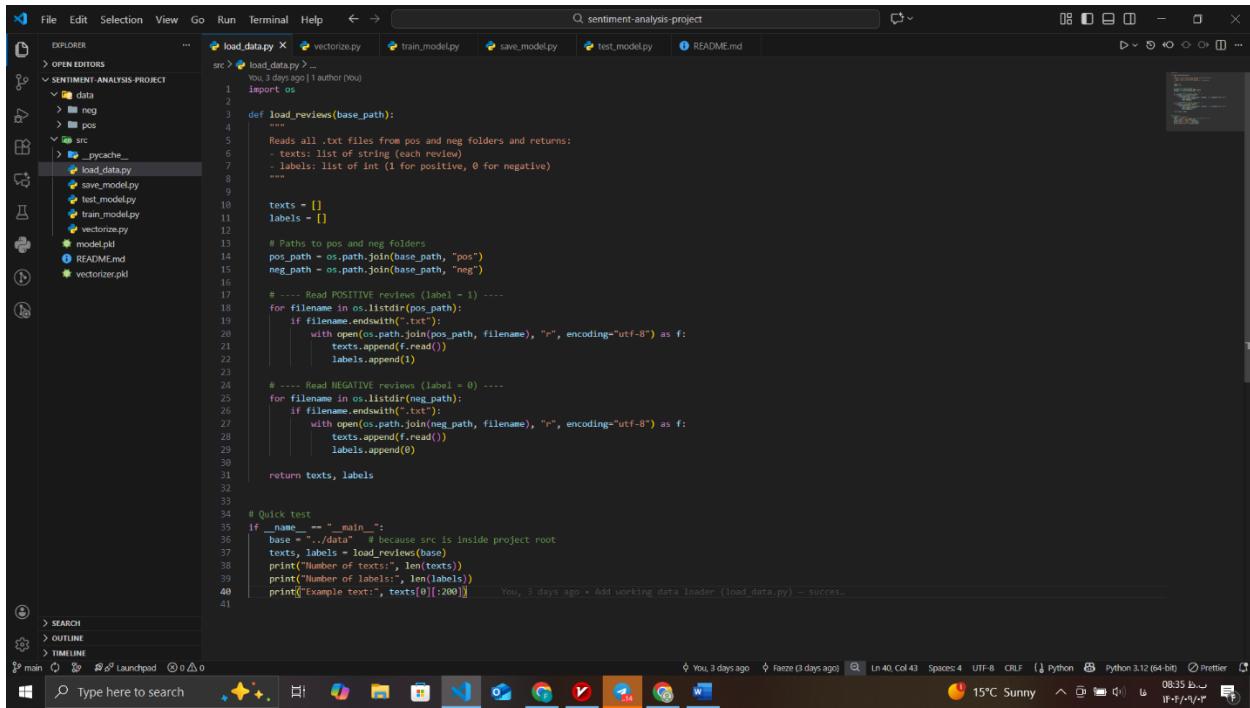
- **save\_model.py**

توی این فایل مدل آموزش داده شده vectorizer و `model.pkl` توی دو فایل جدأگونه ذخیره می‌شن (`vectorizer.pkl` و `model.pkl`). این کار باعث می‌شه که دفعه بعد نیاز نباشه مدل دوباره آموزش داده بشه.

- **test\_model.py**

این فایل یه برنامه تعاملی ساده‌ست؛ برای اینکه ما مدل‌مون رو تست کنیم که درست کار می‌کنه یا نه. یعنی این فایل نتیجه نهایی پروژه رو به ما نشان می‌دهد. به این صورت که کاربر یه جمله وارد می‌کنه و مدل پیش‌بینی می‌کنه که جمله مثبت هست یا منفی.

## مرحله اول: فایل load\_data.py



```
You, 3 days ago | author (You)
File Edit Selection View Go Run Terminal Help ← →
sentiment-analysis-project
OPEN EDITORS
EXPLORER
SENTIMENT-ANALYSIS-PROJECT
src > load_data.py
src > vectorize.py
src > train_model.py
src > save_model.py
src > test_model.py
src > README.md
src > vectorize.pxd
src > model.pxd
src > pycache_
load_data.py
vectorize.py
train_model.py
save_model.py
test_model.py
README.md
vectorize.pxd
load_data.py
You, 3 days ago | author (You)
1 import os
2
3 def load_reviews(base_path):
4     """
5         Reads all .txt files from pos and neg folders and returns:
6         - texts: list of string (each review)
7         - labels: list of int (1 for positive, 0 for negative)
8     """
9
10    texts = []
11    labels = []
12
13    # Paths to pos and neg folders
14    pos_path = os.path.join(base_path, "pos")
15    neg_path = os.path.join(base_path, "neg")
16
17    # ----- Read POSITIVE reviews (label = 1) -----
18    for filename in os.listdir(pos_path):
19        if filename.endswith(".txt"):
20            with open(os.path.join(pos_path, filename), "r", encoding="utf-8") as f:
21                texts.append(f.read())
22                labels.append(1)
23
24    # ----- Read NEGATIVE reviews (label = 0) -----
25    for filename in os.listdir(neg_path):
26        if filename.endswith(".txt"):
27            with open(os.path.join(neg_path, filename), "r", encoding="utf-8") as f:
28                texts.append(f.read())
29                labels.append(0)
30
31    return texts, labels
32
33
34
# Quick test
35 if __name__ == "__main__":
36     base = ".\\data" # because src is inside project root
37     texts, labels = load_reviews(base)
38     print("Number of texts:", len(texts))
39     print("Number of labels:", len(labels))
40     print("Example text:", texts[0][:200])
41
You, 3 days ago | author (You) 3 days ago | author (You) Ln 40 Col 43 Spaces: 4 UTF-8 CRLF Python 3.12 (64-bit) Prettier 15°C Sunny 08:35 بـ ۱۰/۱۰/۲۰۲۳
```

توی این کد، ابتدا کتابخونه استاندارد پایتون به نام `os` وارد شده که برای کار با مسیرها و فایل‌ها کاربرد دارد. سپس تابع `load_reviews` تعریف شده که وظیفه‌اش خوندن همه فایل‌های متنی با پسوند `.txt` از دو فolder `'pos'` و `'neg'` هست؛ این دو فolder شامل نظرات مثبت و منفی هستند. ورودی این تابع مسیر پایه‌ای هست که این دو فolder داخل اون قرار دارن. درون تابع، دو لیست خالی به نام‌های `'texts'` و `'labels'` ساخته شدن تا متن‌ها و برچسب‌هاشون (مثبت یا منفی بودن) توی اون‌ها ذخیره شدن. برای اینکه مسیر دقیق به فolderهای `'pos'` و `'neg'` ساخته بشه، از تابع `os.path.join` استفاده شده تا کد مستقل از سیستم عامل و قابل اجرا روی ویندوز یا لینوکس باشه. بعد، با کمک تابع `os.listdir` همه فایل‌های داخل پوشه مثبت خونده می‌شن و هر فایل با پسوند `'.txt'` باز می‌شه؛ متن داخل فایل با انکوڈینگ UTF-8 خونده شده و به لیست `'texts'` اضافه می‌شه و عدد 1 به عنوان برچسب مثبت به لیست `'labels'` افزوده می‌شه. همین روند برای پوشه نظرات منفی هم تکرار شده با این تفاوت که برچسب صفر (نمایانگر نظر منفی) به لیست برچسب‌ها اضافه می‌شه. در نهایت، دو لیست `'texts'` و `'labels'` که به ترتیب شامل تمام متن‌ها و برچسب‌های اون‌ها هستن، برگردونده می‌شن. در انتهای کد یک بخش تست سریع نوشته شده که در صورت اجرای مستقیم فایل، مسیر داده‌ها به صورت نسبی تنظیم می‌شه، سپس تابع `load_reviews` فراخوانی شده و تعداد کل متنون و برچسب‌ها چاپ می‌شه، همچنین نمونه‌ای از متن اولین نظر هم برای اطمینان از صحّت خوندن داده‌ها نمایش داده می‌شه. به طور کلی این کد وظیفه آماده‌سازی داده‌های اولیه پروژه رو بر عهده داره تا بعدا بتوانیم اون‌ها رو برای آموزش مدل استفاده کنیم.

## خروجی فایل load\_data.py

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure under "SENTIMENT-ANALYSIS-PROJECT".
- EDITOR:** Displays the code for `load_data.py`. The code reads reviews from files in the "data" and "neg" directories.
- TERMINAL:** Shows the command-line output of running the script:

```
PS C:\Users\Faeze\Desktop\sentiment-analysis-project> cd src
PS C:\Users\Faeze\Desktop\sentiment-analysis-project\src> python load_data.py
Number of texts: 25000
Number of labels: 25000
Example text: Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35 years in the teaching profession lead me to believe that Bromwell High's
```
- STATUS BAR:** Shows the current file is Python 3.12 (64-bit) and the terminal uses Prettier.

خروجی به ما نشون می‌ده که تعداد متن‌ها و برچسب‌ها چقدر و این نتیجه لود شدن کامل دیتاست ماست. یه نمونه تکست هم به عنوان مثال نشون می‌ده که بارگذاری کامل رو تایید کنه.

## مرحله دوم: vectorize.py

```
You, 2 days ago | Author (869)
File Edit Selection View Go Run Terminal Help ← →
sentiment-analysis-project
OPEN EDITORS
EXPLORER
SENTIMENT-ANALYSIS-PROJECT
src > vectorize.py
You, 2 days ago | Author (869)
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.model_selection import train_test_split
3
4
5 def vectorize_texts(texts, labels, test_size=0.2):
6     """
7         Converts raw text data into TF-IDF vectors.
8         Splits into train and test sets.
9         Returns: X_train, X_test, y_train, y_test, vectorizer
10    """
11
12    # 1) Converting Text to Numbers with TF-IDF
13    vectorizer = TfidfVectorizer(
14        max_features=10000, # 10k is the best amount to start with.
15        stop_words='english', # Remove useless words like the, a, an
16        ngram_range=(1, 2) # unigram + bigram = much better performance
17    )
18
19    # Constructing the TF-IDF matrix
20    X = vectorizer.fit_transform(texts)
21
22    # Splitting data into train and test
23    X_train, X_test, y_train, y_test = train_test_split(
24        X, labels, test_size=test_size, random_state=42
25    )
26
27    return X_train, X_test, y_train, y_test, vectorizer
28
29
30    # Quick test
31 if __name__ == "__main__":
32     from load_data import load_reviews
33
34     base = '../data'
35     texts, labels = load_reviews(base)
36
37     X_train, X_test, y_train, y_test, vec = vectorize_texts(texts, labels)
38
39     print("Train shape:", X_train.shape)
40     print("Test shape:", X_test.shape)
41     print("Feature count:", len(vec.get_feature_names_out()))
42
```

این کد وظیفه داره که متن های خام رو به شکل بردارهای عددی TF-IDF تبدیل کنه تا مدل های یادگیری ماشین بتونن اون ها رو پردازش کنن. برای این کار از کلاس 'TfidfVectorizer' استفاده شده که ویژگی های مثل حذف کلمات اضافه مثل a و ... the (stop words) استفاده از unigram و bigram، و محدود کردن تعداد ویژگی ها به 10 هزار تا رو داره تا کیفیت و سرعت کار بینه باشه. بعد از ساخت ماتریس TF-IDF، داده ها به دو بخش آموزش و تست تقسیم میشن تا مدل بتونه روی آموزش یاد بگیره و روی تست ارزیابی بشه.

### Unigram چیست؟

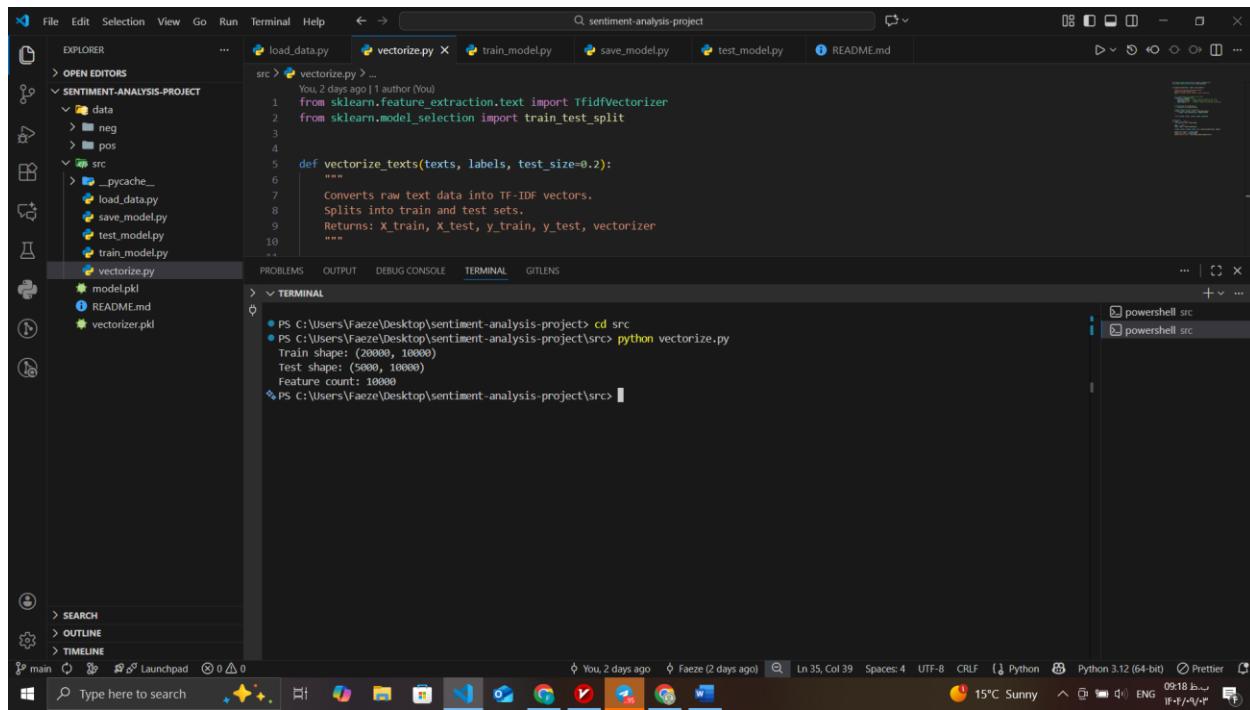
Unigram یعنی وقتی متن رو به کلمات جداگونه تقسیم می کنیم، هر کلمه به تنها یک واحد (Token) محسوب می شه. مثلا جمله "I love movies" رو وقتی با Unigram نگاه کنیم، سه کلمه داریم: "I"، "love" و "movies". پس هر کلمه به صورت جداگونه بررسی می شه و ویژگی ها براساس همین تک کلمه ها ساخته می شن.

### Bigram چیست؟

Bigram یعنی به جای نگاه کردن فقط به تک کلمه ها، به جفت های متواالی کلمات نگاه می کنیم. مثلا همون جمله "I love movies" رو با Bigram به این صورت تقسیم می کنیم: "I love" و "love movies". در واقع، ترکیب دو کلمه پشت سر هم رو به عنوان یه واحد در نظر می گیریم.

حالا چرا از Bigram استفاده می شه؟ چون گاهی معنای یک کلمه به تنها یکی کافی نیست و ترکیب دو کلمه مفهوم دقیق تری به ما میدن. مثلا توی جمله "not good"، اگه فقط به Unigram نگاه کنیم، کلمه "good" بار مثبت دارد. ولی "not good" ترکیبیه که معنی منف می ده. کمک می کنه مدل این ترکیب ها رو بشناسه و نتیجه بهتری به ما بده.

## خروجی فایل vectorize.py



```
You, 2 days ago | 1 author (You)
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.model_selection import train_test_split
3
4
5 def vectorize_texts(texts, labels, test_size=0.2):
6     """
7         Converts raw text data into TF-IDF vectors.
8         Splits into train and test sets.
9         Returns: X_train, X_test, y_train, y_test, vectorizer
10    """
11
12 PS C:\Users\Faeze\Desktop\sentiment-analysis-project> cd src
13 PS C:\Users\Faeze\Desktop\sentiment-analysis-project\src> python vectorize.py
Train shape: (20000, 10000)
Test shape: (5000, 10000)
Feature count: 10000
PS C:\Users\Faeze\Desktop\sentiment-analysis-project\src>
```

خروجی تابع پنج تا متغیر هست: بردارهای آموزش و تست ( $X\_train$ ,  $X\_test$ ,  $y\_train$ ,  $y\_test$ )، برچسب‌های مربوط به هر کدام ( $y\_train$ ,  $y\_test$ ) و خود شیء `vectorizer` که برای تبدیل متن‌های جدید به بردارهای TF-IDF لازم هست. به طور خلاصه، این کد متون را آماده و تقسیم‌بندی می‌کنند تا برای آموزش و ارزیابی مدل آماده باشند.

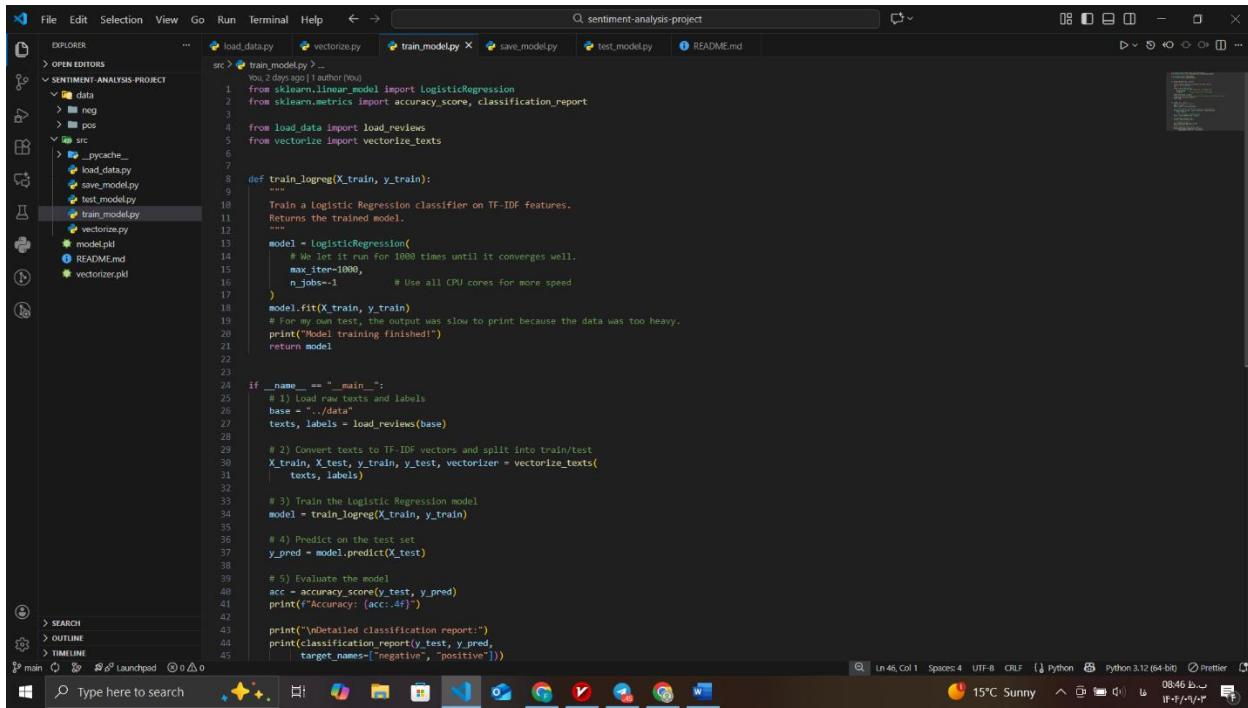
داده‌های آموزش:  $X\_train$ .

داده‌های تست:  $X\_test$ .

برچسب‌های آموزش:  $y\_train$ .

برچسب‌های تست:  $y\_test$ .

## مرحله سوم: فایل train\_model.py



```
File Edit Selection View Go Run Terminal Help < > sentiment-analysis-project
OPEN EDITORS
SENTIMENT-ANALYSIS-PROJECT
src > train_model.py
You, Zdenek [author] (you)
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.metrics import accuracy_score, classification_report
3
4 from load_data import load_reviews
5 from vectorize import vectorize_texts
6
7
8 def train_logreg(X_train, y_train):
9     """
10     Train a Logistic Regression classifier on TF-IDF features.
11     Returns the trained model.
12     """
13     model = LogisticRegression(
14         # We let it run for 1000 times until it converges well.
15         max_iter=1000,
16         n_jobs=-1           # Use all CPU cores for more speed
17     )
18     model.fit(X_train, y_train)
19     # For my own test, the output was slow to print because the data was too heavy.
20     print("Model training finished!")
21     return model
22
23
24 if __name__ == "__main__":
25     # 1) Load raw texts and labels
26     base = "./data"
27     texts, labels = load_reviews(base)
28
29     # 2) Convert texts to TF-IDF vectors and split into train/test
30     X_train, X_test, y_train, y_test, vectorizer = vectorize_texts(
31         texts, labels)
32
33     # 3) Train the logistic regression model
34     model = train_logreg(X_train, y_train)
35
36     # 4) Predict on the test set
37     y_pred = model.predict(X_test)
38
39     # 5) Evaluate the model
40     acc = accuracy_score(y_test, y_pred)
41     print(f"Accuracy: {acc:.4f}")
42
43     print("\nDetailed classification report:")
44     print(classification_report(y_test, y_pred,
45                                  target_names=["negative", "positive"]))
46
47 main 0 0 0 Launchpad 0 0 0 Type here to search 11:46 PM 15°C Sunny 08:46 AM Python 3.12 (64-bit) Prettier
```

این کد برای آموزش یک مدل Logistic Regression استفاده می‌شود که روی ویژگی‌های TF-IDF کار می‌کند. اول با استفاده از تابع `load\_reviews` داده‌های متنی و برجسب‌ها را از پوشه داده‌ها می‌خونیم. بعدش متن‌ها را با تابع `vectorize\_texts` تبدیل به بردارهای عددی TF-IDF می‌کنیم و داده‌ها را به مجموعه آموزش و تست تقسیم می‌کنیم. توی تابع `train\_logreg` یک مدل Logistic Regression ساخته شده که اجازه می‌ده 1000 بار روی داده‌ها تکرار کنند تا به بهترین حالت برسه و از تمام هسته‌های پردازند برای سرعت بیشتر استفاده می‌کنند. مدل روی داده‌های آموزشی آموزش داده می‌شود و پس از اتمام، پیام "Model training finished" نمایش داده می‌شود. سپس مدل روی داده‌های تست پیش‌بینی انجام میده و دقت (accuracy) مدل محاسبه و چاپ می‌شود. در نهایت گزارش جزئی‌تری از عملکرد مدل با استفاده از `classification\_report` نمایش داده می‌شود که نشون میده مدل چقدر در تشخیص درست نظرات مثبت و منفی موفق بوده. این کد به طور کلی مراحل اصلی آموزش، تست و ارزیابی مدل را به صورت ساده و شفاف انجام میده.

## خروجی فایل train\_model.py

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a project structure named "SENTIMENT-ANALYSIS-PROJECT" containing "data", "src", and "pycache\_" folders. Inside "src", there are files: "load\_data.py", "vectorize.py", "train\_model.py", "save\_model.py", and "test\_model.py".
- Code Editor:** The "train\_model.py" file is open, displaying Python code for training a Logistic Regression classifier. The code imports necessary libraries and defines a function "train\_logreg" that trains the model on TF-IDF features.
- Terminal:** The terminal window shows the command "python train\_model.py" being run, resulting in the message "Model training finished!". It also displays a detailed classification report with the following data:

	precision	recall	f1-score	support
negative	0.90	0.88	0.89	2485
positive	0.88	0.90	0.89	2515
accuracy			0.89	5000
macro avg	0.89	0.89	0.89	5000
weighted avg	0.89	0.89	0.89	5000

The terminal also shows the command "PS C:\Users\...\" and the date/time "15°C Sunny".

خروجی فایل `train\_model.py` در اصل مدل آموزش داده شده Logistic Regression هست که آماده استفاده برای پیش‌بینی احساسات متون جدیده است. علاوه بر این، هنگام اجرای فایل، دقت مدل روی داده‌های تست چاپ می‌شود و گزارشی کامل از عملکرد مدل شامل معیارهای مثل دقت، بازیابی و F1-score برای هر کلاس (مثبت و منفی) نمایش داده می‌شود. این خروجی‌ها به مأکمل می‌کنند که بفهمیم مدل چقدر خوب کار می‌کند و چقدر قابل اعتماد است.

(دقت): Precision

می‌گه از بین تمام نمونه‌هایی که مدل گفته «مثبت» یا «یک» هستن، چند درصد واقعاً درست پیش‌بینی شدن. یعنی وقتی مدل می‌گه «مثبت»، چقدر مطمئنیم که واقعاً مثبت باشند.

(بازخوانی یا حساسیت): Recall

می‌گه از بین همه نمونه‌های واقعی مثبت، مدل چقدر توانسته درست تشخیص بده. یعنی مدل چقدر خوب نمونه‌های مثبت واقعی را پیدا کرده.

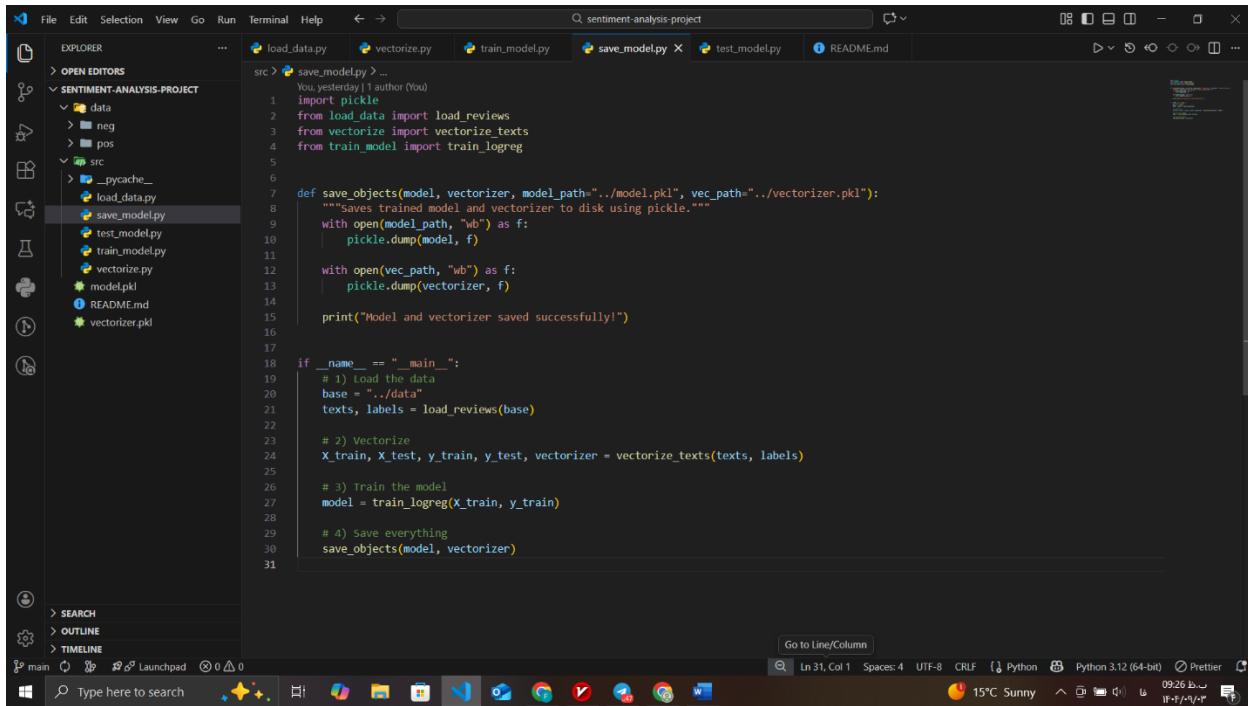
: (امتیاز F1) F1-score

میانگینی بین Precision و Recall هست که تعادل بین این دو رو نشون می‌دهد. وقتی هم دقت بالا باشد و هم بازخوانی بالا، F1 هم بالا میرد.

(حمایت): Support

تعداد واقعی نمونه‌هایی که هر کلاس دارد. یعنی مثلاً چند داده واقعی مثبت داریم و چند داده واقعی منفی. این عدد فقط برای اطلاع هست و معیار عملکرد نیست.

## مرحله چهارم: فایل save\_model.py



The screenshot shows a code editor interface with the following details:

- File Path:** sentiment-analysis-project
- Editor View:** The "save\_model.py" file is open in the main editor area.
- Code Content:** The code defines a function `save\_objects` that saves a trained model and vectorizer to disk using pickle. It includes steps for loading data, vectorizing texts, training a Logistic Regression model, and saving everything.

```
src > save_model.py > ...
You, yesterday | 1 author (You)
1 import pickle
2 from load_data import load_reviews
3 from vectorize import vectorize_texts
4 from train_model import train_logreg
5
6 def save_objects(model, vectorizer, model_path="../model.pkl", vec_path="../vectorizer.pkl"):
7     """Saves trained model and vectorizer to disk using pickle."""
8     with open(model_path, "wb") as f:
9         pickle.dump(model, f)
10
11     with open(vec_path, "wb") as f:
12         pickle.dump(vectorizer, f)
13
14     print("Model and vectorizer saved successfully!")
15
16 if __name__ == "__main__":
17     # 1) Load the data
18     base = "../data"
19     texts, labels = load_reviews(base)
20
21     # 2) Vectorize
22     X_train, X_test, y_train, y_test, vectorizer = vectorize_texts(texts, labels)
23
24     # 3) Train the model
25     model = train_logreg(X_train, y_train)
26
27     # 4) Save everything
28     save_objects(model, vectorizer)
29
30
31
```

- Toolbars and Status Bar:** The status bar at the bottom shows the line number (Ln 31), column (Col 1), spaces (Spaces: 4), encoding (UTF-8), Python version (Python 3.12 (64-bit)), and Prettier status.

این کد یه پروژه ساده برای آموزش مدل تشخیص احساسات متون است که مراحل اصلی رو از بارگذاری داده‌ها تا ذخیره مدل نهایی پوشش می‌ده. ابتدا با استفاده از تابع `load\_reviews`، متن‌های خام و برجسب‌های مربوط به اون‌ها (ثبت یا منفی) از پوشه داده‌ها خونده می‌شن. سپس این متن‌ها به کمک تابع `vectorize\_texts` به بردارهای عددی TF-IDF تبدیل می‌شن و داده‌ها به دو بخش آموزش و تست تقسیم می‌شن تا مدل بتونه روی داده‌های آموزشی یاد بگیره و روی داده‌های تست ارزیابی بشه. در مرحله بعد، مدل Logistic Regression با تابع `train\_logreg` روی داده‌های آموزش آموزش داده می‌شه تا بتونه الگوهای موجود در داده‌ها رو یاد بگیره. در نهایت، برای اینکه نیاز نباشه هر بار برنامه اجرا بشه و مدل دوباره آموزش ببینه، مدل آموزش دیده و همچنین بردارساز-TF-IDF به کمک کتابخانه `pickle` به صورت بازیزی در فایل‌های جداگانه ذخیره می‌شن (`model.pkl` برای مدل و `vectorizer.pkl` برای بردارساز). این کار باعث می‌شه که در دفعات بعدی فقط با بارگذاری این فایل‌ها، بتوان مدل رو بدون آموزش مجدد به کار گرفت.

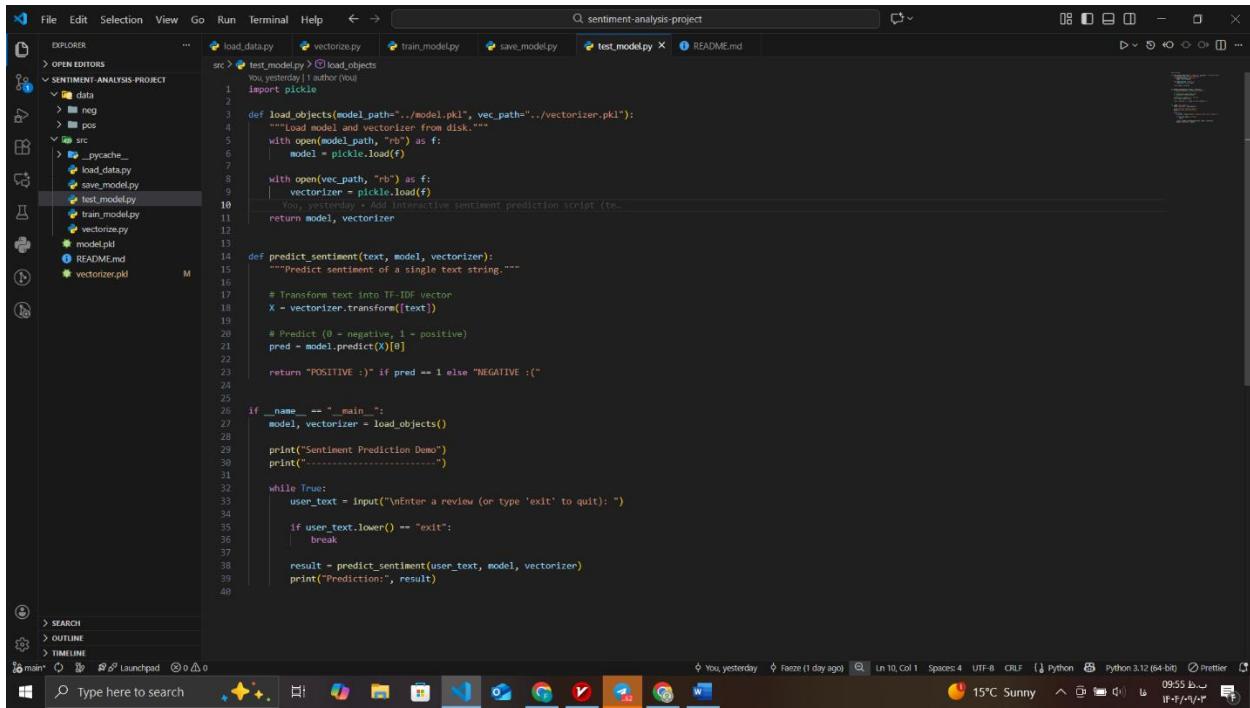
## خروجی فایل save\_model.py

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project named "SENTIMENT-ANALYSIS-PROJECT" containing files: data, neg, pos, src, \_pycache\_, load\_data.py, save\_model.py, test\_model.py, vectorize.py, model.pkl, README.md, and vectorizer.pkl.
- Code Editor:** The active file is "save\_model.py". The code defines a function "save\_objects" that saves a trained model and vectorizer to disk using pickle.
- Terminal:** The terminal shows the command "python save\_model.py" being run, resulting in the message "Model training finished!" and "Model and vectorizer saved successfully!".
- Status Bar:** Shows the current file is "In 31, Col 1", spaces are set to 4, encoding is UTF-8, and the Python version is 3.12 (64-bit). It also displays the date and time as 15°C Sunny 09:35 AM, ENG, and the keyboard layout as US-International.

خروجی نهایی این برنامه دو فایل ذخیره شده روی دیسک هست که شامل مدل آموزش دیده و ابزار تبدیل متن به بردار عددی هست و پایه و اساس مراحل بعدی پیش‌بینی احساسات متن‌های جدید هست. به طور خلاصه، این کد از داده‌های خام شروع می‌کنه، مدل یادگیری ماشین را آموزش می‌ده و اون رو برای استفاده در آینده ذخیره می‌کنه.

## مرحله پنجم: فایل test-model.py



```
You, yesterday | author (text)
1 import pickle
2
3 def load_objects(model_path='../model.pkl', vec_path='../vectorizer.pkl'):
4     """Load model and vectorizer from disk."""
5     with open(model_path, "rb") as f:
6         model = pickle.load(f)
7
8     with open(vec_path, "rb") as f:
9         vectorizer = pickle.load(f)
10
11     return model, vectorizer
12
13
14 def predict_sentiment(text, model, vectorizer):
15     """Predict sentiment of a single text string."""
16
17     # Transform text into TF-IDF vector
18     X = vectorizer.transform([text])
19
20     # Predict (0 = negative, 1 = positive)
21     pred = model.predict(X)[0]
22
23     return "POSITIVE :)" if pred == 1 else "NEGATIVE :("
24
25
26 if __name__ == "__main__":
27     model, vectorizer = load_objects()
28
29     print("Sentiment Prediction Demo")
30     print("-----")
31
32     while True:
33         user_text = input("\nEnter a review (or type 'exit' to quit): ")
34
35         if user_text.lower() == "exit":
36             break
37
38         result = predict_sentiment(user_text, model, vectorizer)
39         print("Prediction:", result)
40
41
42
43
44
45
46
47
48
```

توی این فایل عملاً مرحله «استفاده از مدل» انجام می‌شه؛ یعنی جایی که دیگر قرار نیست مدل رو آموزش بدیم، بلکه می‌خوایم مدل ذخیره شده رو بالا بباریم و روی متن‌های جدید پیش‌بینی انجام بدیم. همه‌چیز از دو فایل `model.pkl` و `vectorizer.pkl` شروع می‌شه؛ این دو فایل همون مدل آموزش‌داده شده و بردارساز TF-IDF هستن که قبل از ذخیره شون کرده بودیم.

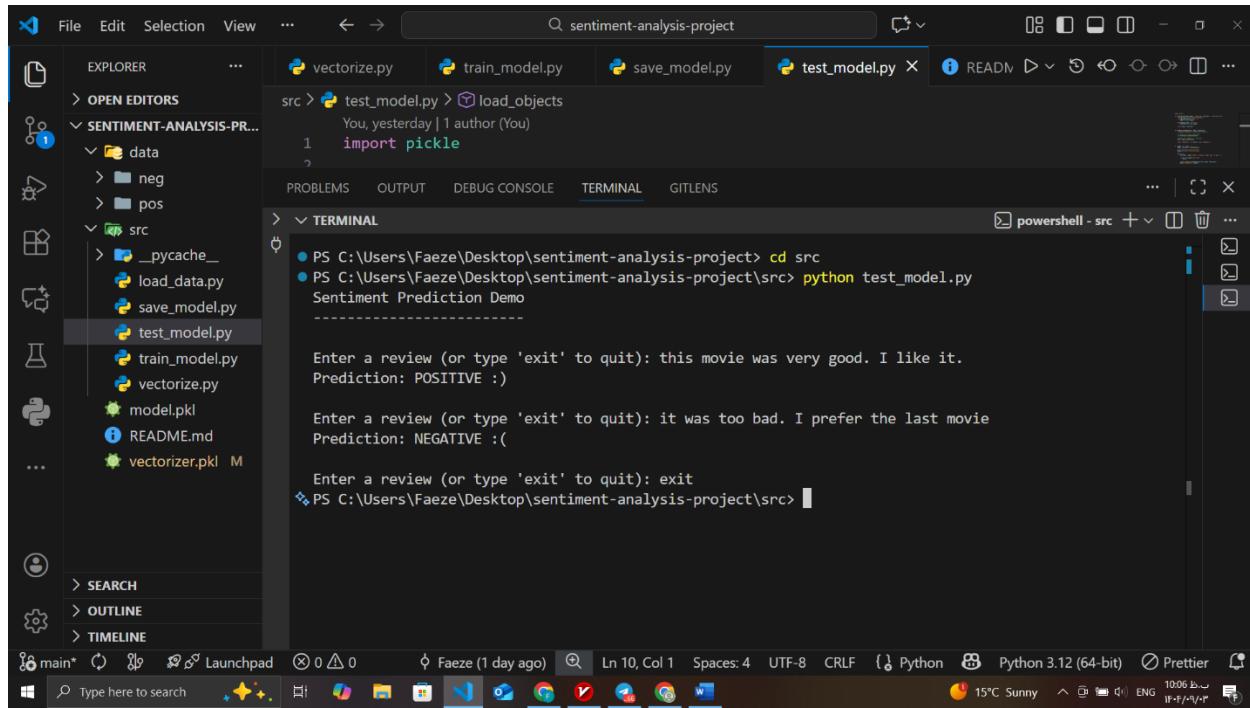
در ابتدای کد، تابع `load\_objects` با استفاده از کتابخانه pickle کارش اینه که این دو فایل رو از دیسک باز کنه و دوباره تبدیل شون کنه به دو شیء قابل استفاده توی پایتون؛ یکی مدل و یکی vectorizer. بدون این مرحله، هر بار باید مدل رو از صفر آموزش می‌دادیم، اما با لود کردن، مستقیم می‌تونیم از اون استفاده کنیم.

بعد از این، تابع `predict\_sentiment` نوشته شده که وظیفه‌اش گرفتن یه متن خام و تبدیل اون به یه بردار TF-IDF (با استفاده از همان vectorizer ذخیره شده) هست. سپس مدل روی این بردار پیش‌بینی انجام می‌ده و مشخص می‌کنه که این متن مثبته یا منفی. خروجی تابع به صورت یه رشته ساده نمایش داده می‌شه:

"NEGATIVE ):" یا "POSITIVE (:"

توی بخش اصلی برنامه (`\_\_if \_\_name\_\_ == "\_\_main\_\_"`)، ابتدا مدل و بردارساز با `load\_objects` لود می‌شن. سپس یک حلقه تعاملی شروع می‌شه که کاربر می‌تونه هر جمله‌ای تایپ کنه و برنامه فوراً احساس متن رو پیش‌بینی و چاپ می‌کنه. اگر کاربر کلمه `exit` رو وارد کنه، برنامه متوقف می‌شه.

## خروجی فایل test\_model.py



```
PS C:\Users\Faeze\Desktop\sentiment-analysis-project> cd src
PS C:\Users\Faeze\Desktop\sentiment-analysis-project\src> python test_model.py
Sentiment Prediction Demo
-----
Enter a review (or type 'exit' to quit): this movie was very good. I like it.
Prediction: POSITIVE :)

Enter a review (or type 'exit' to quit): it was too bad. I prefer the last movie
Prediction: NEGATIVE :(

Enter a review (or type 'exit' to quit): exit
PS C:\Users\Faeze\Desktop\sentiment-analysis-project\src>
```

خروجی اصلی این فایل برای متنی هست که کاربر وارد می‌کنه. یعنی هر بار کاربر یه جمله تایپ میکنه و برنامه بهش میگه مثبته یا منفی:

Prediction: POSITIVE(:

...

یا

...

Prediction: NEGATIVE(:

...

یعنی خروجی نهایی این فایل، یک سیستم تعاملی تشخیص احساسات هست که از مدل آموزش داده شده استفاده می‌کنه و بدون نیاز به آموزش مجدد، روی هر متن دلخواه پیش‌بینی انجام می‌دهد.

توضیحات کلی من راجع به این پروژه تموم شد، برای اطلاع بیشتر از روند پروژه و کدها، می‌توانید فایل `Readme` داخل ریپازیتوری گیت هابم رو بخونین.

من برای این پروژه از ویدئوهای مربوط به ماشین لرنینگ و پردازش زبان طبیعی یوتیوب، سایت [w3schools](#) و مدل‌های زیانی کمک گرفتم. شروع پروژه از روز جمعه 30 آبان ساعت 3 و نیم آغاز شد و لان که دارم گزارشم رو تکمیل می‌کنم، ساعت 10 و نیم شب روز 3 آذر هست. این فقط فرآیند برنامه نویسی بود و من از قبل این زمان هم برای درک دقیق پروژه، چراً انجام اون و مواردی که لازم بود آموزش ببینم هم کلی تحقیق کردم. واقعیتش درک این پروژه و یادگرفتن مسیر انجام اون، خیلی لذت‌بخش بود، چون حالا دیگه می‌دونم مبنای ماشین لرنینگ چیه و اصلاً چجوری می‌شه یه ماشین رو آموزش داد.

بدون شک نتیجه نهایی قابل گسترش هست و من تمام سعیم رو به کار می‌گیرم که به محض یادگیری بیشتر، این پروژه رو هم توسعه بدم.

