AIML_Exit_Exam.ipynb

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

Files

- ..
- drive
  - MyDrive
    - AI ML Notes
    - AI Research Papers
    - Colab Notebooks
    - DataSet
    - ICTAK_Project_Data
    - ICT_Project_IntelligentGOSe...
    - Misc
    - Resume
    - exit_exam
      - Reviews.csv
      - Reviews.csv.zip
    - Prototype.pdf
    - Python Quiz Web App
    - Python Quiz Web App for C...
    - Python Quiz Web App for C...
    - Rohit S_Resume (1).pdf

Disk 68.00 GB available

[1] ✓ 4s

```
import pandas as pd
import string
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from nltk.corpus import stopwords
import nltk
```

[2] ✓ 0s

```
nltk.download('stopwords')
STOPWORDS = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[3]

```
# Text cleaning function
def clean_text(text):
    text = text.lower()
    text = re.sub(f"[{string.punctuation}]", " ", text)
    text = re.sub(r"\d+", "", text)
    text = " ".join([word for word in text.split() if word not in STOPWORDS])
    return text
```

[4] ✓ 14s

```
# Load data
df = pd.read_csv("/content/drive/MyDrive/exit_exam/Reviews.csv")
df = df[['Text', 'Score']].dropna()
```

[5]

```
# Binary label: Positive (Score >= 4), Negative (Score <= 2)
df = df[df['Score'] != 3]
df['Sentiment'] = df['Score'].apply(lambda x: 'Positive' if x >= 4 else 'Negative')
```

[6] ✓ 22s

```
# Clean text
```

Variables Terminal

✓ 2:00 PM Python 3

https://colab.research.google.com/drive/1iz7c6vjL9dt55jRYZ0NshsyWtIlSEHVM#scrollTo=Z6ivt8ycBFVm

AIML_Exit_Exam.ipynb

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

Files

..

drive

MyDrive

AI ML Notes

AI Research Papers

Colab Notebooks

DataSet

ICTAK_Project_Data

ICT_Project_IntelligentGOSe...

Misc

Resume

exit_exam

Reviews.csv

Reviews.csv.zip

Prototype.pdf

Python Quiz Web App

Python Quiz Web App for C...

Python Quiz Web App for C...

Rohit S_Resume (1).pdf

Disk 68.00 GB available

[8] 23s

```
# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

[9] 12s

```
# Logistic Regression Model
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_tfidf, y_train)
y_pred = lr.predict(X_test_tfidf)
```

[10] 1s

```
# Evaluation
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    Negative       0.85      0.68      0.76     16379
    Positive       0.94      0.98      0.96     88784

    accuracy                           0.93    105163
   macro avg       0.90      0.83      0.86    105163
weighted avg       0.93      0.93      0.93    105163
```

Count Vectorizer: simply counts the total frequency or occurance of words in each document, which could potentially lead to insignificant words getting higher importance in features. whereas; TF-IDF (Term Frequency–Inverse Document Frequency): adds weights not just by frequency but also by rarity of the words, there by giving unique words higher importance than regularly occuring ones. There by providing better results.

[ ]

Variables Terminal

2:00 PM Python 3

Very high UV Now

Search

ENG IN 14:10 15-10-2025