OBIDA ALHAMOUD

Batch code: LISUM35

27-july-2024

Submitted to: Data Galcier

1.introduction

In this project, we are going to deploying a ML Model (SVM).

This model can classify the airline reviews if it is positive or natural or negative.

The model trained on airlines dataset from Kaggle.

In this project we will:

- Import the data
- Import necessary packages
- Find the best parameters for our model using grid search
- Deploy the model using Flask framework

2. Import the data

The data is collected manually and uploaded to Kaggle

This dataset contain tweets id’s and the text , airline sentiment

import the data

```
df = pd.read_csv('airline_tweets.csv')
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America

3. Import necessary packages

```
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.2,random_state=99)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

- And Flask Framework:

Pip install Flask

4.Find the best parameters for our model using gridsearch

finding the best parameters for the model

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# pipeline with standard scaling and SVC
pipeline = Pipeline([
    ('scaler', TfidfVectorizer(stop_words='english')),
    ('svc', SVC())
])

#parameter grid
param_grid = {
    'svc__C': [0.1, 1, 10, 100],
    'svc__gamma': [1, 0.1, 0.01, 0.001],
    'svc__kernel': ['rbf', 'linear', 'poly', 'sigmoid'],
    'svc__class_weight': [None, 'balanced']
}

#GridSearchCV object to find the best performans
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-1)

# Fitting the grid search to the data
grid_search.fit(X_train, y_train)

print("Best parameters found: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best parameters found:  {'svc__C': 100, 'svc__class_weight': 'balanced', 'svc__gamma': 1, 'svc__kernel': 'rbf'}
Best accuracy:  0.7845796097392783
```

Model Report

```
from sklearn.metrics import ConfusionMatrixDisplay,classification_report,confusion_matrix
```

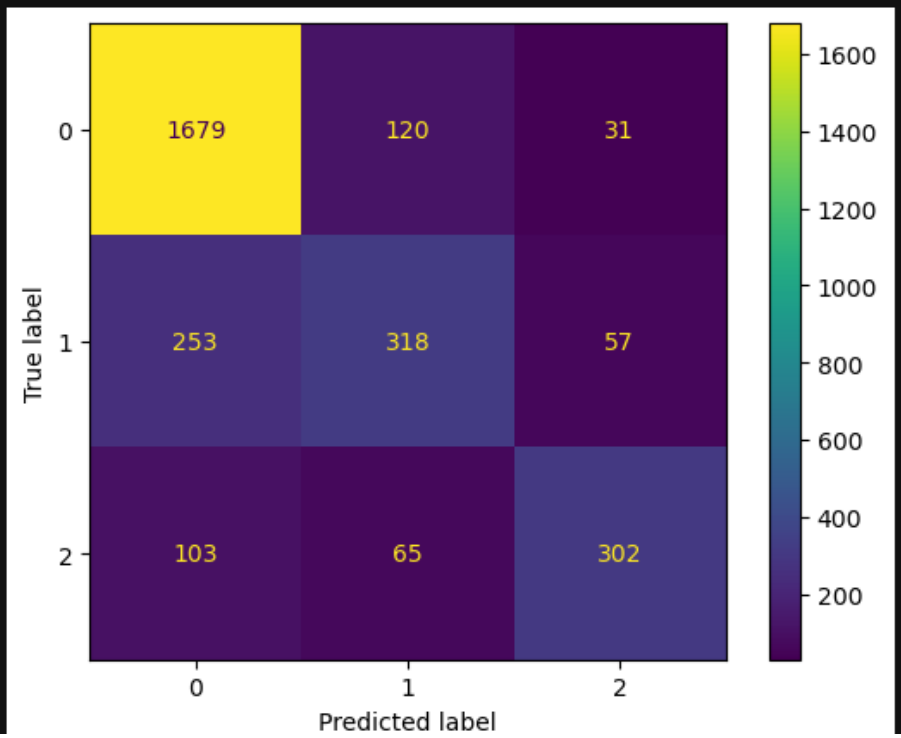
```
preds = best_pipeline.predict(X_test)
print(classification_report(y_test,preds))
cm = confusion_matrix(y_test, preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
              precision    recall  f1-score   support

    negative       0.83      0.92      0.87      1830
     neutral       0.63      0.51      0.56       628
    positive       0.77      0.64      0.70       470

    accuracy                           0.79      2928
   macro avg       0.74      0.69      0.71      2928
weighted avg       0.78      0.79      0.78      2928
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x26848b2fc50>
```



5.Deploy the model using Flask framework

First lets save the model!

Saving the Model & Using it

```
import joblib
```

```
joblib.dump(best_pipeline, 'best_pipeline_model.joblib')
```

```
['best_pipeline_model.joblib']
```

```
loaded_model = joblib.load('Flaskapp/best_pipeline_model.joblib')
```

```
loaded_model.predict(['Good Flight '])
```

```
array(['positive'], dtype=object)
```

5.1 app.py

```python
from flask import Flask, request, render_template
import joblib

app = Flask(__name__)

# Loading the saved model
model = joblib.load('best_pipeline_model.joblib')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        review = request.form['review']
        prediction = model.predict([review])[0]
        return render_template('index.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```

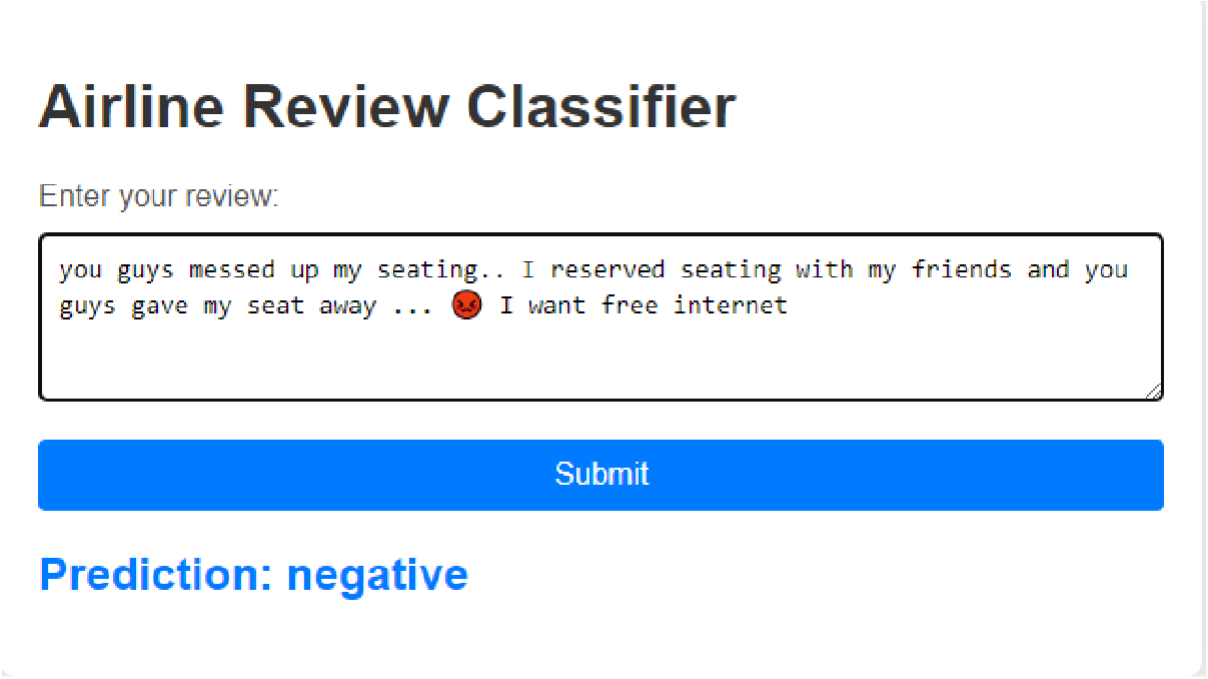
6. Running the app

in the terminal execute this command

python app.py

after that we open a web browser and navigate to http://172.0.0.1:5000/

after that you can simply put any review in the text box and get the prediction.



---

7. Deploying on Heroku

First we need to create 2 files:

- Requirements.txt
- Procfile

In requirements file we add all the packeges are used in the project

```
pillow==10.3.0
protobuf==4.25.3
pyarrow==16.1.0
pydeck==0.9.1
Pygments==2.18.0
python-dateutil==2.9.0.post0
pytz==2024.1
referencing==0.35.1
requests==2.32.2
rich==13.7.1
rpds-py==0.18.1
six==1.16.0
smmap==5.0.1
streamlit==1.35.0
tenacity==8.3.0
toml==0.10.2
toolz==0.12.1
tornado==6.4
typing_extensions==4.11.0
tzdata==2024.1S
urllib3==2.2.1
```

Next is the procfile

```
web: gunicorn app:app
```

Next we setup Heroku

8. Heroku

After we sign up to Heroku we execute these commands:

Heroku login to login to your Heroku account
git init to initialize you git repo
git add . and git commit to commit your changes
git push Heroku master to push the files into Hekoru

Heroku open <app name> to open your Heroku app

To access to my app use this link below :

https://afternoon-lake-87217-93bc8ee0d3ac.herokuapp.com/