



Data Glacier

Your Deep Learning Partner

Model deployment on Flask

Nikola Andrejić

Intern at Data Glacier

Contents

1	Creating and training the model	3
1.1	Basic python libraries	3
1.2	The Dataset	3
1.3	Feature preparation	4
1.4	Creating the model	5
2	Deployment on flask	7

1 CREATING AND TRAINING THE MODEL

1.1 BASIC PYTHON LIBRARIES

We start with importing numpy and pandas as the basic libraries for handling our dataset.

```
In [1]: #Importing the Libraries
import numpy as np
import pandas as pd
```

1.2 THE DATASET

The dataset we use for training our model is the **IMDB Dataset of 50K Movie Reviews**. This dataset contains the review and the label “sentiment” that is either “positive” or “negative”.

```
In [2]: #Importing the dataset
df = pd.read_csv("IMDB Dataset.csv")
```

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   review      50000 non-null  object
 1   sentiment   50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

```
In [4]: df.head()
```

Out[4]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

1.3 FEATURE PREPARATION

We notice that the review contains some html tags.

```
In [5]: #We need to convert html to text
df["review"][1]
```

```
Out[5]: 'A wonderful little production. <br /><br />The filming technique is
very unassuming- very old-time-BBC fashion and gives a comforting, a
nd sometimes discomfoting, sense of realism to the entire piece. <br
 /><br />The actors are extremely well chosen- Michael Sheen not on
ly "has got all the polari" but he has all the voices down pat too!
You can truly see the seamless editing guided by the references to W
illiams\' diary entries, not only is it well worth the watching but
it is a terrificly written and performed piece. A masterful producti
on about one of the great master\'s of comedy and his life. <br /><br
 />The realism really comes home with the little things: the fantas
y of the guard which, rather than use the traditional \'dream\' tech
niques remains solid then disappears. It plays on our knowledge and
our senses, particularly with the scenes concerning Orton and Halliwe
ll and the sets (particularly of their flat with Halliwell\'s mural
s decorating every surface) are terribly well done.'
```

To deal with this, we convert this column to text using the `html2text` function from the `html2text` module.

```
In [6]: from html2text import html2text
```

```
In [7]: #Converting html to text
html2text(df.loc[1,"review"])
```

```
Out[7]: 'A wonderful little production. \n \nThe filming technique is very
unassuming- very old-time-BBC fashion and gives\na comforting, and s
ometimes discomfoting, sense of realism to the entire\npiece. \n
\nThe actors are extremely well chosen- Michael Sheen not only "has
got all the\npolar"but he has all the voices down pat too! You can
truly see the seamless\nediting guided by the references to Williams
\' diary entries, not only is it\nwell worth the watching but it is
a terrificly written and performed piece. A\nmasterful production ab
out one of the great master\'s of comedy and his life. \n \nThe re
alism really comes home with the little things: the fantasy of the g
uard\nwhich, rather than use the traditional \'dream\' techniques re
mains solid then\ndisappears. It plays on our knowledge and our sens
es, particularly with the\nscenes concerning Orton and Halliwell and
the sets (particularly of their flat\nwith Halliwell\'s murals decor
ating every surface) are terribly well done.\n\n'
```

```
In [8]: #Now for the whole dataset
df["review"]=df["review"].apply(lambda rev: html2text(rev))
```

1.4 CREATING THE MODEL

We start with noticing that the dataset is perfectly balanced in terms of labels.

```
In [9]: #This dataset is perfectly balanced
df["sentiment"].value_counts()
```

```
Out[9]: positive    25000
        negative    25000
        Name: sentiment, dtype: int64
```

Next we separate the features (the review) and the label (the sentiment) and then also divide the dataset into train and test subsets.

```
In [10]: #Separate features (which is a review) from a label (sentiment)
X=df["review"]
y=df["sentiment"]
```

```
In [11]: #Perform a train test split with 20% test size
from sklearn.model_selection import train_test_split
```

```
In [12]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,
                                                         random_state=101)
```

We create our model as a pipeline that has two steps. The first step performs the tfidf vectorization with english stop words and the second step applies the Naive Bayes itself. We train the model on the training set and make predictions on the test set.

```
In [13]: #Create a model with tfidf vectorization and Naive Bayes classifier.
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
steps=[("tfidf",TfidfVectorizer(stop_words="english")),
        ("nb",MultinomialNB())]
pipe=Pipeline(steps)
```

```
In [14]: #Train the model
pipe.fit(X_train,y_train)
```

```
Out[14]: Pipeline(steps=[('tfidf', TfidfVectorizer(stop_words='english')),
                          ('nb', MultinomialNB())])
```

```
In [15]: #Predict the labels for the test set
y_predict = pipe.predict(X_test)
```

To evaluate our model we print the classification report and the confusion matrix.

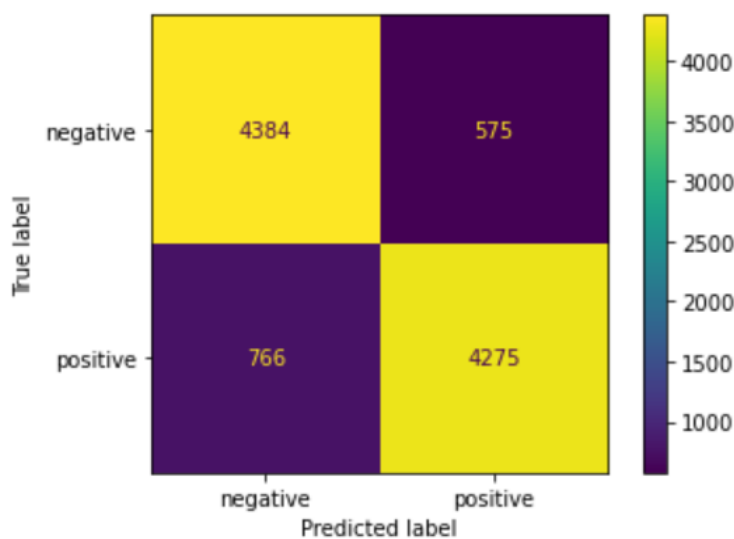
```
In [16]: #Evaluate the model by observing the classification report
#and the confusion matrix
from sklearn.metrics import (plot_confusion_matrix,
                             classification_report)
```

```
In [17]: print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
negative	0.85	0.88	0.87	4959
positive	0.88	0.85	0.86	5041
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

```
In [18]: plot_confusion_matrix(pipe,X_test,y_test)
```

```
Out[18]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x
26763b905e0>
```



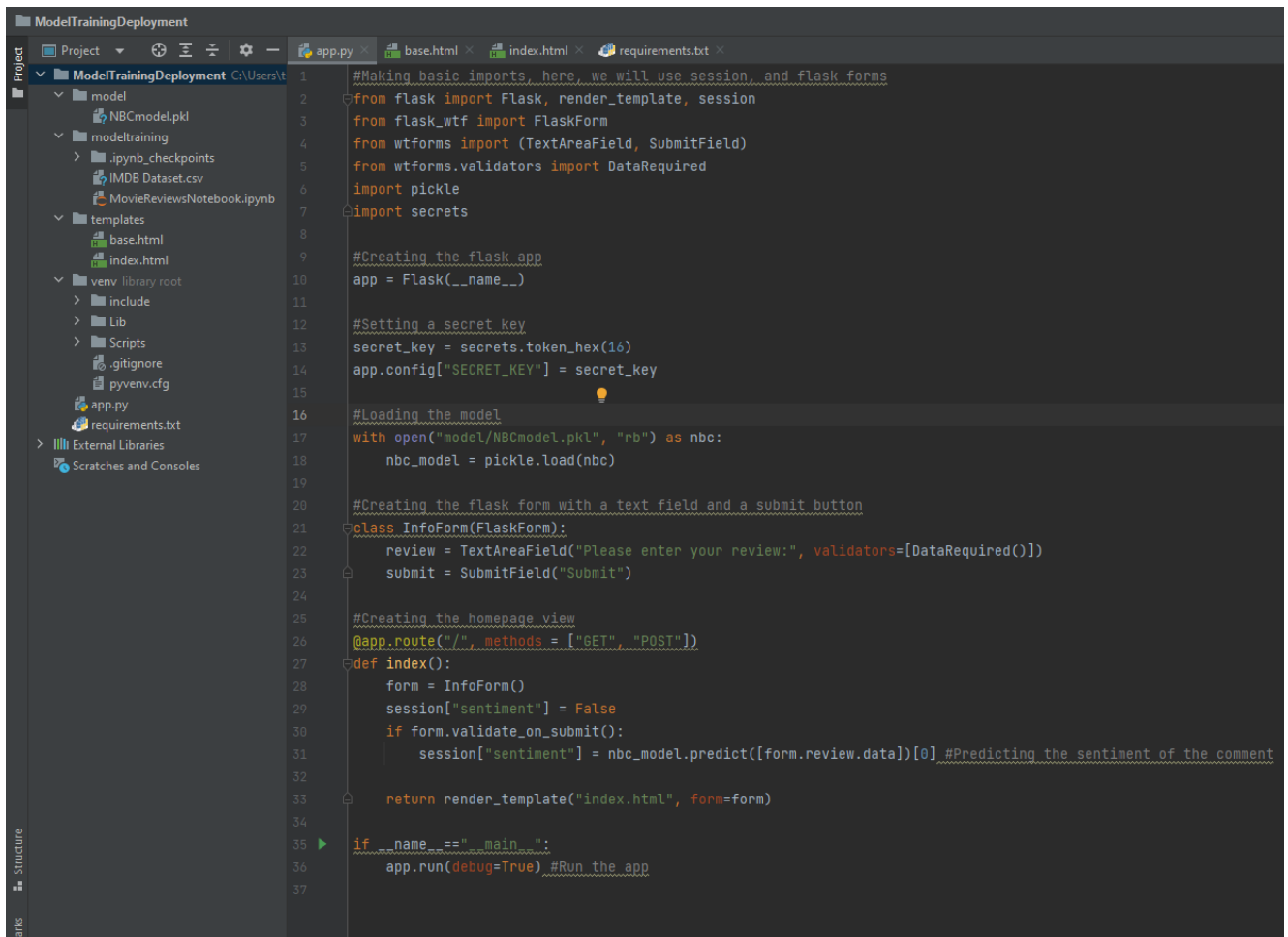
Finally, we save our model with pickle.

```
In [19]: import pickle
```

```
In [20]: #Saving the model with pickle in a .pkl file
with open("../model\\NBCmodel.pkl","wb") as nbc:
    pickle.dump(pipe,nbc)
```

2 DEPLOYMENT ON FLASK

To deploy on flask we create a python project in pycharm where easily set the virtual environment from the requirements.txt file. We will use bootstrap 4 to style our pages and also we will interact with jinja templates via Flask forms and sessions. In the following figures we provide the snapshots of the app.py file, our html files and the requirements.txt file. Notice that we load the model that we previously saved with pickle in lines 17 and 18 of app.py and we apply it for predicting the sentiment in line 31.



```

1  #Making basic imports, here, we will use session, and flask forms
2  from flask import Flask, render_template, session
3  from flask_wtf import FlaskForm
4  from wtforms import (TextAreaField, SubmitField)
5  from wtforms.validators import DataRequired
6  import pickle
7  import secrets
8
9  #Creating the flask app
10 app = Flask(__name__)
11
12 #Setting a secret key
13 secret_key = secrets.token_hex(16)
14 app.config["SECRET_KEY"] = secret_key
15
16 #Loading the model
17 with open("model/NBCmodel.pkl", "rb") as nbc:
18     nbc_model = pickle.load(nbc)
19
20 #Creating the flask form with a text field and a submit button
21 class InfoForm(FlaskForm):
22     review = TextAreaField("Please enter your review:", validators=[DataRequired()])
23     submit = SubmitField("Submit")
24
25 #Creating the homepage view
26 @app.route("/", methods = ["GET", "POST"])
27 def index():
28     form = InfoForm()
29     session["sentiment"] = False
30     if form.validate_on_submit():
31         session["sentiment"] = nbc_model.predict([form.review.data])[0] #Predicting the sentiment of the comment
32
33     return render_template("index.html", form=form)
34
35 if __name__ == "__main__":
36     app.run(debug=True) #Run the app
37

```

```

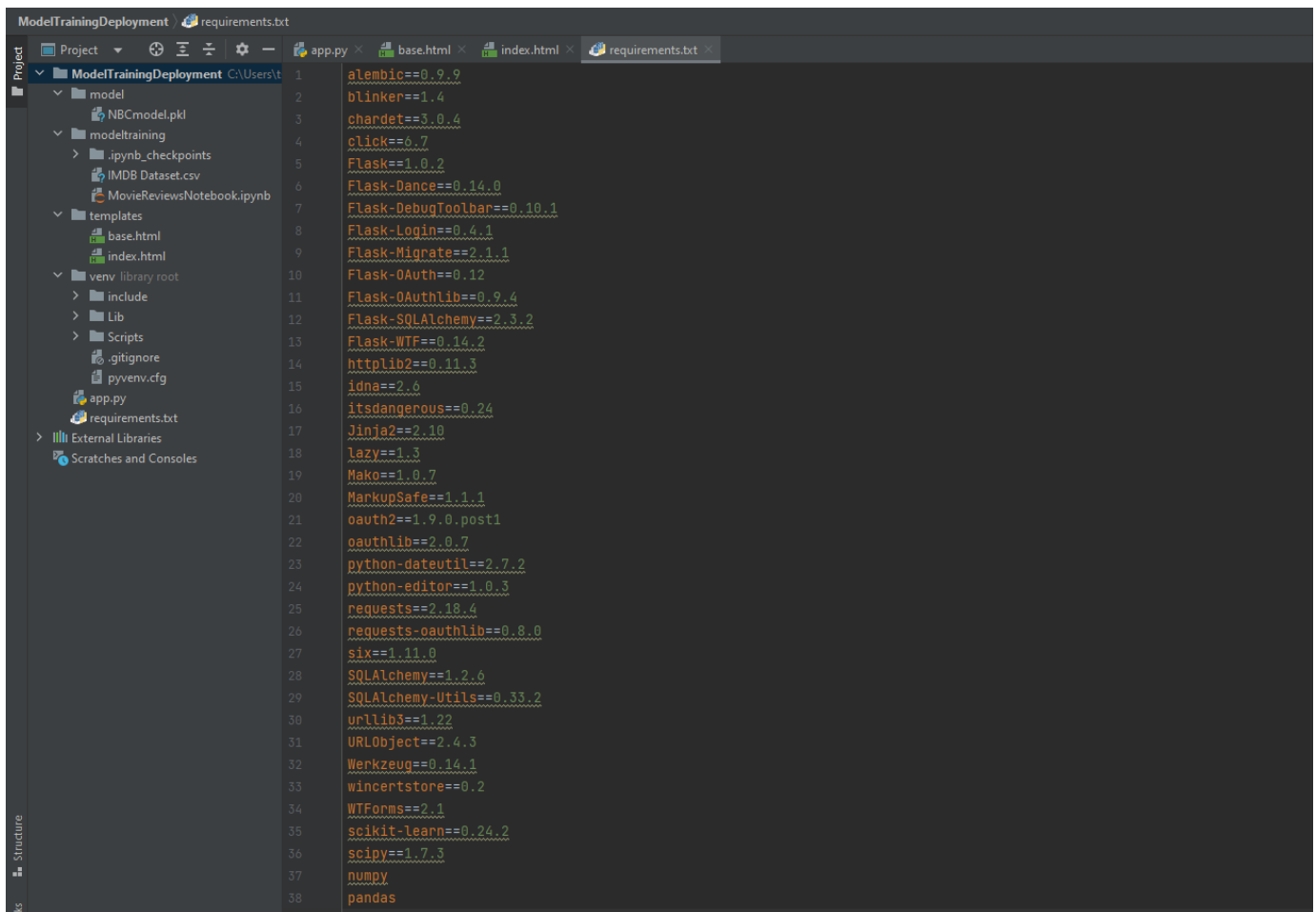
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Movie Reviews Classifier</title>
6   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-sf...>
7   <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzph...>
8   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mI...>
9   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js" integrity="sha384-smHYKdLADwkX0...>
10 </head>
11 <body>
12
13 <nav class="navbar navbar-dark bg-dark">
14   <a class="navbar-brand" href="#">Movie Reviews Classifier</a>
15 </nav>
16 <br>
17 <br>
18
19 {% block content %}
20 {% endblock %}
21
22 </body>
23 </html>

```

```

1 {% extends "base.html"%}
2 {% block content %}
3 <div class="container">
4   <div class="jumbotron">
5     <center>
6       <form method="POST">
7         {{ form.hidden_tag() }}
8         {{form.review.label}}<br>
9         {{form.review(cols="50", rows="10")}}
10        <br>
11        {{form.submit()}}
12        <br>
13      </form>
14    </center>
15  </div>
16
17  {%if session['sentiment'] == "positive" %}
18  <div class="alert alert-success" role="alert">
19    The review is positive &#128077
20  </div>
21  {% elif session['sentiment'] == "negative" %}
22  <div class="alert alert-danger" role="alert">
23    The review is negative &#128078
24  </div>
25  </div>
26  {% endif %}
27
28 {% endblock %}

```

```

1  alembic==0.9.9
2  blinker==1.4
3  chardet==3.0.4
4  click==6.7
5  Flask==1.0.2
6  Flask-Dance==0.14.0
7  Flask-DebugToolbar==0.10.1
8  Flask-Login==0.4.1
9  Flask-Migrate==2.1.1
10 Flask-OAuth==0.12
11 Flask-OAuthLib==0.9.4
12 Flask-SQLAlchemy==2.3.2
13 Flask-WTF==0.14.2
14 httpLib2==0.11.3
15 idna==2.6
16 itsdangerous==0.24
17 Jinja2==2.10
18 lazy==1.3
19 Mako==1.0.7
20 MarkupSafe==1.1.1
21 oauth2==1.9.0.post1
22 oauthlib==2.0.7
23 python-dateutil==2.7.2
24 python-editor==1.0.3
25 requests==2.18.4
26 requests-oauthlib==0.8.0
27 six==1.11.0
28 SQLAlchemy==1.2.6
29 SQLAlchemy-Utils==0.33.2
30 unrlib3==1.22
31 URLObject==2.4.3
32 Werkzeug==0.14.1
33 wincertstore==0.2
34 WTForms==2.1
35 scikit-learn==0.24.2
36 scipy==1.7.3
37 numpy
38 pandas

```

In the figure below we provide the log of running our application.



```

Run: app
C:\Users\tsnik\DGInternship\Nikola\Week4\ModelTrainingDeployment\venv\Scripts\python.exe C:\Users\tsnik\DGInternship\Nikola\Week4\ModelTrainingDeployment/app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 155-404-073
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Finally, let's check out how our application looks in the browser. Before the submission we have a text area where we type our review and a submit button.



Please enter your review:

Submit

After we submit the review we get a notification bar below that tells us the predicted sentiment.

Please enter your review:

Before watching I thought it was boring and not interesting, but I was mistaken. I liked everything: acting of main characters, the unexpected plot and beautiful love.

Submit

The review is positive 🍌

Please enter your review:

It had some interesting moments, but the movie is an overall failure and I don't recommend wasting your time on it

Submit

The review is negative 🍋