Airline Sentiment Analysis Deployment in Docker using Flask

Overview

The airline industry is a very competitive market which has grown rapidly in the past 2 decades. Customer feedback is very crucial to Airline companies as it helps in improving the quality of services and facilities provided to customers. Airline companies resort to traditional customer feedback forms for sentiment analysis which in turn are very tedious and time consuming. The digital connectivity bestows immense power to the customers in terms of vocalizing their thoughts, opinions, and reviews on a brand. The customer views expressed on Twitter, Facebook, and other online forums are forming the base of customer strategy for brands worldwide as they are genuine and much more reliable.

In this capstone project we will be focusing on the sentiment analysis of 6 major US Airlines from Twitter data. Job is to find the problems of each major US airline. For this project we will be using Twitter data (reviews submitted by individuals who traveled through various Airlines) which was scraped from February 2015 and contributors (mostly workers from Crowdflower) were asked to first classify positive, negative, and neutral tweets followed by categorizing negative reasons (such as “late flight” or “rude service”)

Goals

Create a Model which on providing the tweets(reviews) to the training should provide the outcome that whether a particular tweet done by an individual is a positive response or a negative one or neutral. Ultimately with this sentiment analysis or opinion mining Airlines can gain better user insights and strategize their brands to provide better service to the customers.

- **Positive Response**: Reviews that contains good experience of traveler with airlines
- **Negative Response**: Reviews that contains difficulty faced by the traveler
- **Neutral Response**: Reviews which are not specific to be considered in positive or negative

Dataset

We will be using Twitter scrapped data from 2015 which can be found on Kaggle https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment?select=Tweets.csv .It is a structured dataset with about 6 columns and 15000 rows

Specifications

Laptop Specifications (where most of the code and model will be run):
Processor: 2.6 GHz 6-Core Intel Core i7
Memory: 16 GB 2667 MHz DDR4
Graphics: Intel UHD Graphics 630 1536 MB
On top of this if required we will be using paperspace for additional memory and computing.

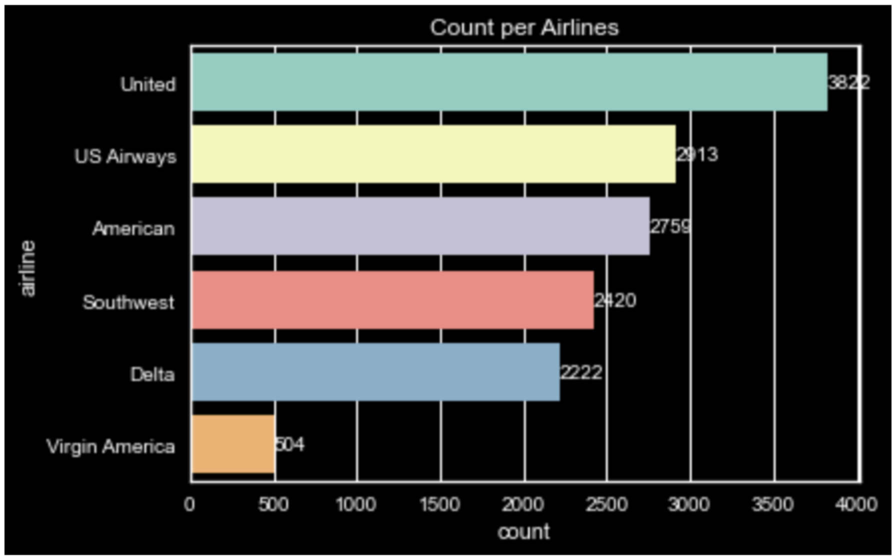
Workflow

Here are steps I have followed in this project from Data Analysis till Deployment.

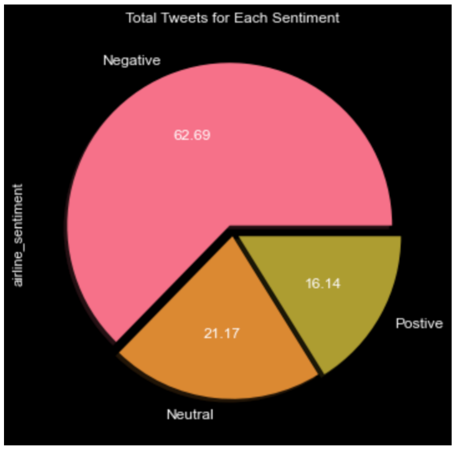
1. EDA
2. Text Processing/Cleaning
   - Remove Stop words
   - Remove punctuation
   - Remove hashtags etc.
3. Encoding
   - Start with TFIDF Vectorizer
   - Word2Vec Embedding
4. Handling Imbalance
   - Measure the performance before and after handling Imbalance
   - SMOTE can be used for this
   - Oversampling and under sampling minority class
5. ML Models
   - Start with Logistic Regression
   - Random Forest
   - Support Vector Machine (SVM)
   - XGBoost, Catboost etc
   - Naive Bayes
   - Deep Learning Models
     - LSTM
6. Use Grid Search/Random Search to tune Hyperparameters
   - Cross Validation using K-Fold
7. Deploy phase - deploy the best model
   - Containerize
   - Docker
   - Build API
   - Build Web GUI/ interface to text out the Model

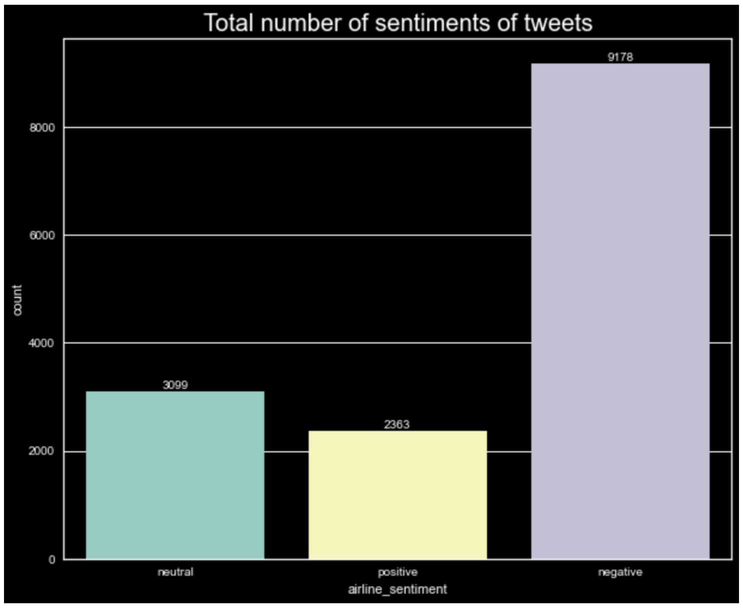
EDA is a very important step while building a Machine Learning application as it gives us a first impression of the dataset. Here are some results of the EDA analysis that was done as part of this project.

Total number of Tweets per Airlines:

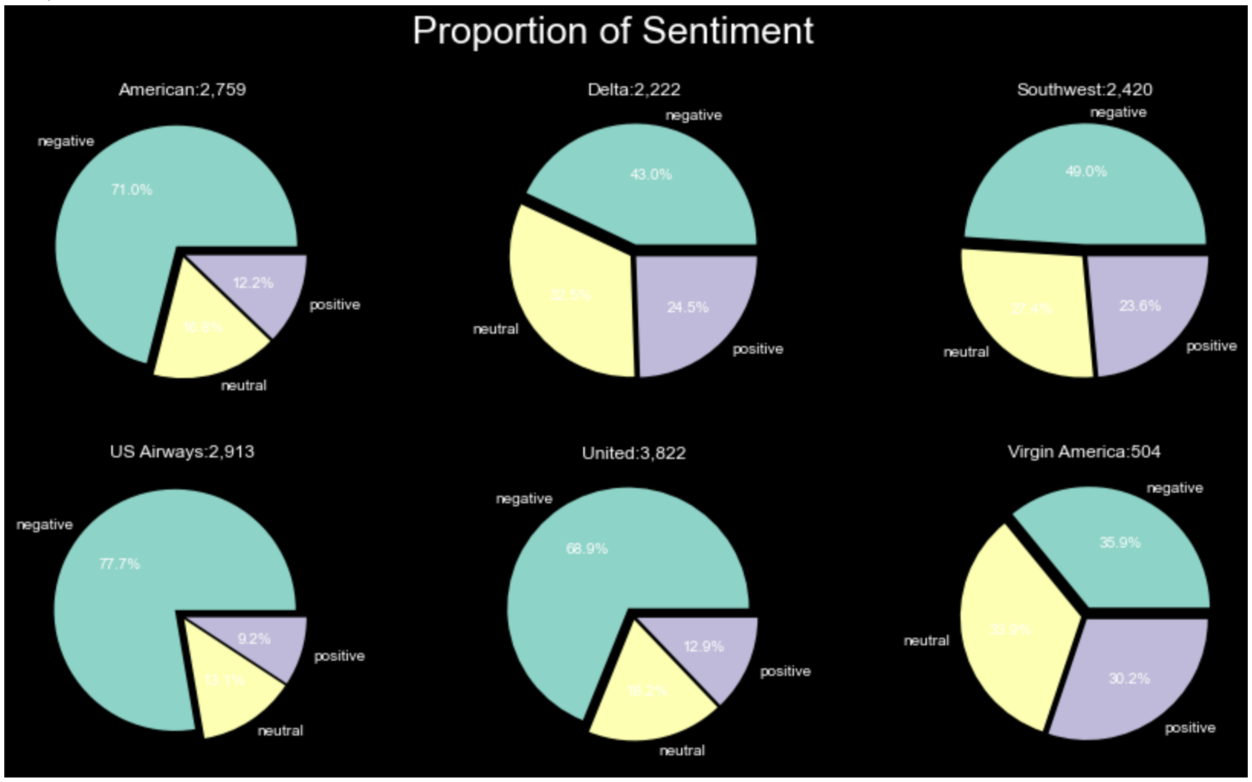


Total Tweets of each Sentiment:





Proportion of Sentiment:



It's clear from the EDA that there is an imbalance of data. We see that Negative sentiments are way more than neutral and positive ones which will bias our model. So, handling imbalance will be a very important step in building this model. We have used **SMOTE** package of Python to handle imbalance.

As part of the Text Cleaning and Processing stop words, punctuations, emoji's etc were removed from the data. As machine learning model only understands numbers, these data went through encoding where we evaluated methods like TFDIF, WordtoVec etc. After that this data was fed into the SMOTE package to balance it out. This balanced data then was used to train various machine leaning models like Logistic regression, K-nearest neighbors, Random Forest, Decision Trees, Support vector Machine, Gradient Boosting Classifier, XG Boost, Ada Boost, Naïve Bayes and LSTM which is a Deep Learning Model. Various Metrics were looked at like Accuracy Score, F1 Score, Precession, Recall, Confusion Matrix, and ROC curves.

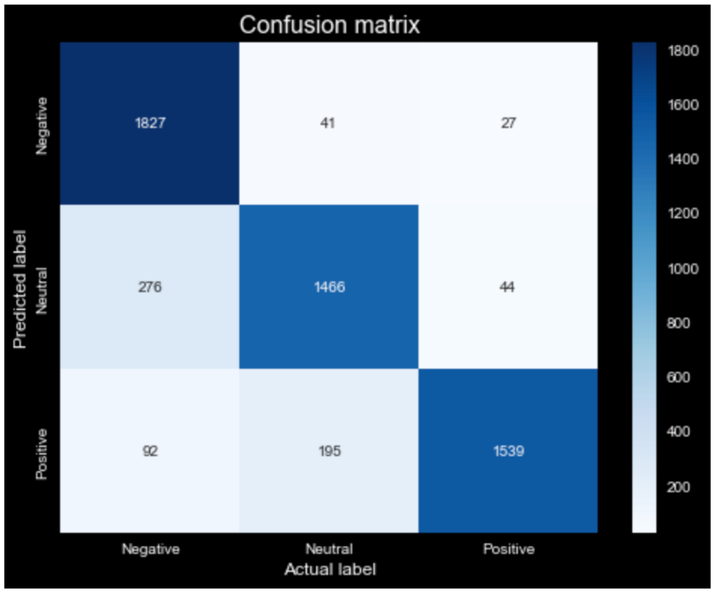
After rigorous iterations of training the model, we found that SVM and Random Forest Classifier gave better results with accuracy close to 88%. In this report we will just look at Support Vector Machine and Random Forest. Remaining results including LSTM can be found in **model.ipynb** notebook.

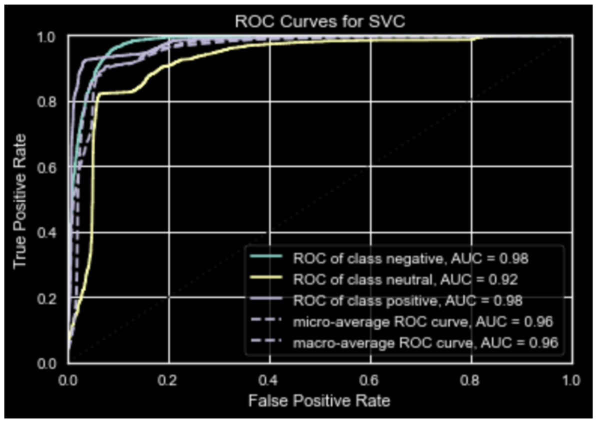
Results

Support Vector Machine

Accuracy Score: 0.8774287270746323

```
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      1895
           1       0.86      0.82      0.84      1786
           2       0.96      0.84      0.90      1826

    accuracy                           0.88      5507
   macro avg       0.88      0.88      0.88      5507
weighted avg       0.88      0.88      0.88      5507

Confusion Matrix :
[[1827   41   27]
 [ 276 1466   44]
 [  92  195 1539]]
```

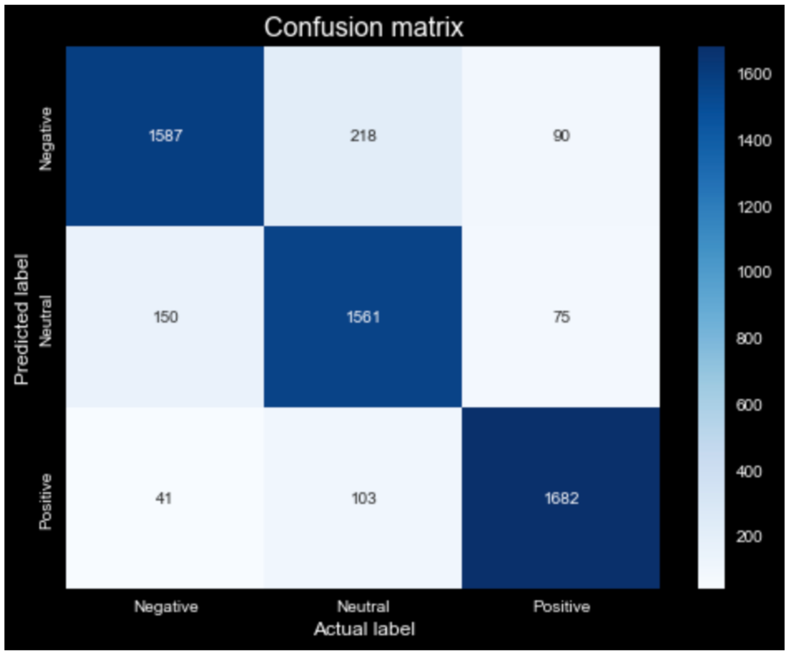


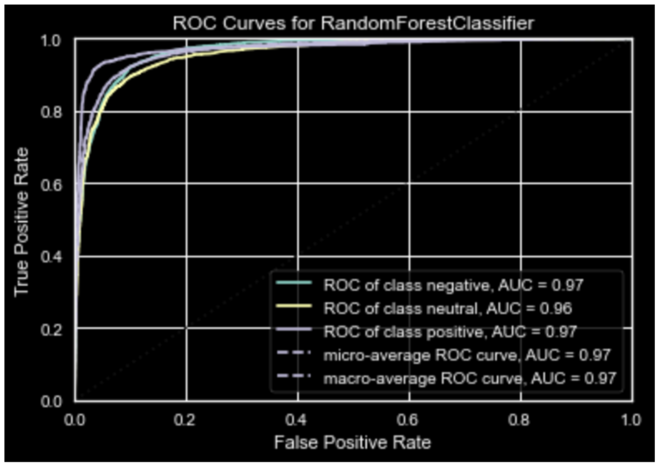


Random Forest Classifier

Accuracy Score: 0.8770655529326312

```
              precision    recall  f1-score   support

           0       0.89      0.84      0.86      1895
           1       0.83      0.87      0.85      1786
           2       0.91      0.92      0.92      1826

    accuracy                           0.88      5507
   macro avg       0.88      0.88      0.88      5507
weighted avg       0.88      0.88      0.88      5507

Confusion Matrix :
[[1587  218   90]
 [ 150 1561   75]
 [  41  103 1682]]
```





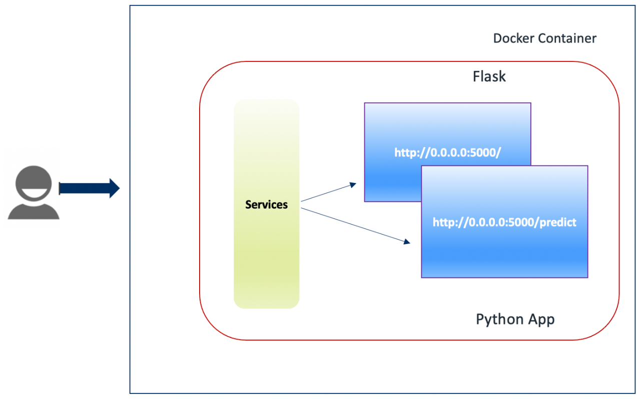
Deployment

As part of the deployment phase, we have exposed few APIs that would return the sentiment of the text entered. We have used python’s FLASK module to create REST APIs. We have then used Docker to containerize our Machine Learning model. This will help us to deploy in production irrespective of the platform.

Deployment phase consists of various steps which are listed below:

- Trained and Saved the Model
  - ML model can be saved using python’s pickle package (serialise)
- Created *app.py* which would build the web service (API) using Flask
  - Run app.py
  - Navigate to http address to see if it’s working
- Created DockerFile with following info:
  - Python version
  - Install packages in requirements.txt
  - Set WORKDIR to /app
  - Expose the port for flask endpoint
  - Specify the host IP address as 0.0.0.0
- Build Docker Image
- Run Docker Container
- Navigate to 0.0.0.0/5000 which is the docker endpoint that will route to Index.html
  - Index.html is a simple html page that has a form which takes in a user entered text and makes POST request to get the sentiment.
- Test out the POST call by providing a text and see if returns a sentiment which can be either positive, negative, or neutral.

We have used Twitter data for Airline and used to predict the sentiments which can be negative, positive, and neutral. We have built a machine learning model using Support Vector Machine which we will be deploying in docker container and expose it as REST API which we will access over HTTP. We have used FLASK python package to expose our machine learning model as a REST API.



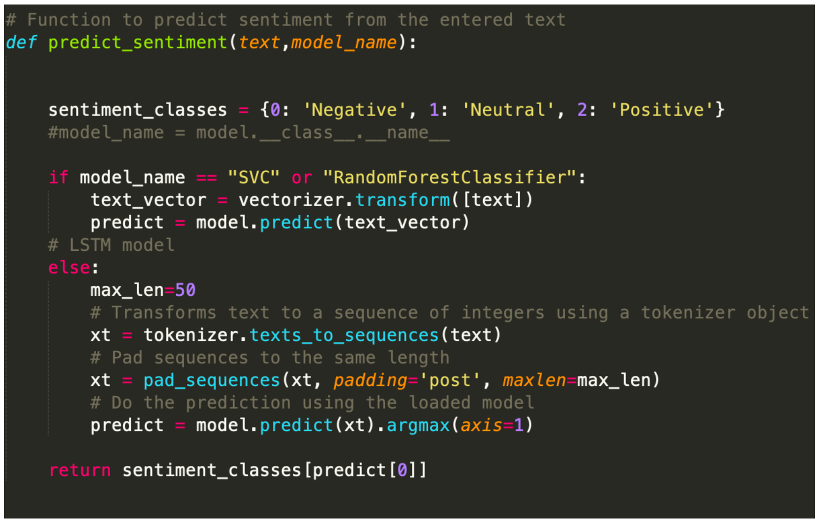
We have saved few models which had similar accuracy like SVM and Random Forest. LSTM model is also saved after it was trained with the test data. All the models are saved in models directory.

App.py

This is the file which creates API for the model using python's FLASK package. It has basically two endpoints:

- http://localhost:5000/ - This endpoint lands onto the Index.html page which is a very simple html page consisting of a form that accepts Text and has a button to give the sentiment prediction. Clicking on that button after entering the text will do a POST call to the ML model with the following endpoint.
- http://localhost:5000/predict - This endpoint calls the predict sentiment function to predict the sentiment. It does some text preprocessing before calling the predict function on the ML Model.

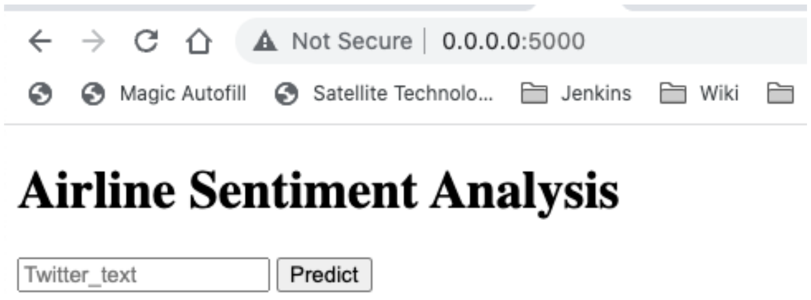
Predict Sentiment function:



```python
# Function to predict sentiment from the entered text
def predict_sentiment(text,model_name):


    sentiment_classes = {0: 'Negative', 1: 'Neutral', 2: 'Positive'}
    #model_name = model.__class__.__name__

    if model_name == "SVC" or "RandomForestClassifier":
        text_vector = vectorizer.transform([text])
        predict = model.predict(text_vector)
    # LSTM model
    else:
        max_len=50
        # Transforms text to a sequence of integers using a tokenizer object
        xt = tokenizer.texts_to_sequences(text)
        # Pad sequences to the same length
        xt = pad_sequences(xt, padding='post', maxlen=max_len)
        # Do the prediction using the loaded model
        predict = model.predict(xt).argmax(axis=1)

    return sentiment_classes[predict[0]]
```

Index.html

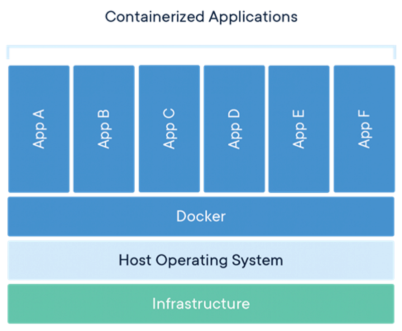
Below screenshot is of the index.html page which shows a simple form with a text field and a button.



Airline Sentiment Analysis

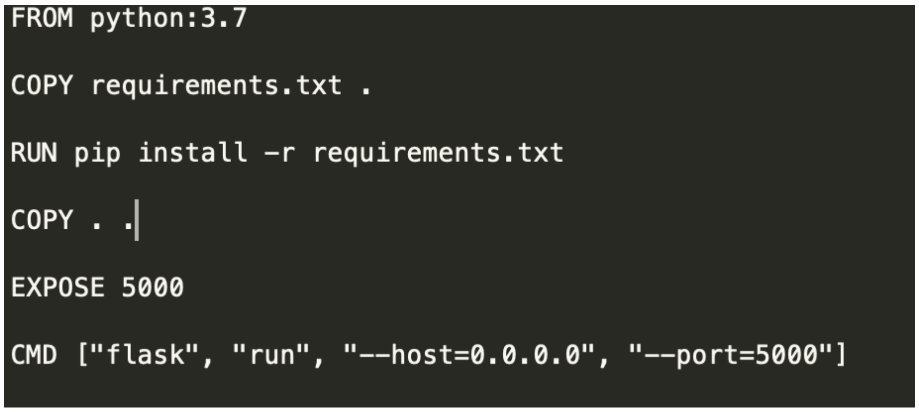
Containerize ML Model

Docker is a tool that is used to ease the deployment of the containers. Docker allow users to create, deploy and run services using containers. You might have worked with virtual machines. Docker is like a virtual machine, but it can share the underline operating system without need of all dedicated operating system.



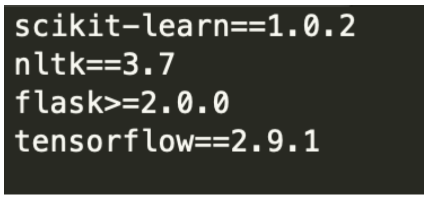
Dockerfile contents:

The Dockerfile is the most important part of this process besides the model itself. The Dockerfile is what creates the image that's used to create the container that hosts the model and API



```
FROM python:3.7

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
```

We have a requiremets.txt file which has all the packages information required for this Machine Leaning Model. Contents of requirements.txt:

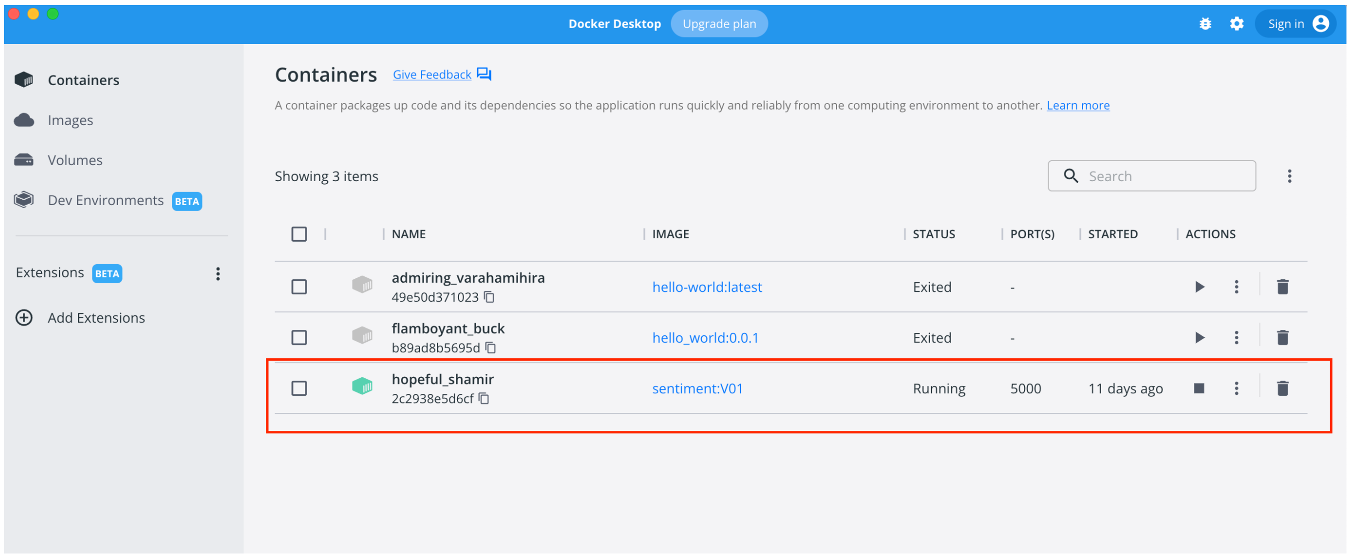


```
scikit-learn==1.0.2
nltk==3.7
flask>=2.0.0
tensorflow==2.9.1
```

Building The Docker image and Docker Container

Following commands were used in the project to build a docker image and then run it.

Build the image: `docker build -t sentiment:V01 .`



Docker Desktop | Upgrade plan | Sign in

Containers | Images | Volumes | Dev Environments BETA | Extensions BETA | Add Extensions

Containers Give Feedback

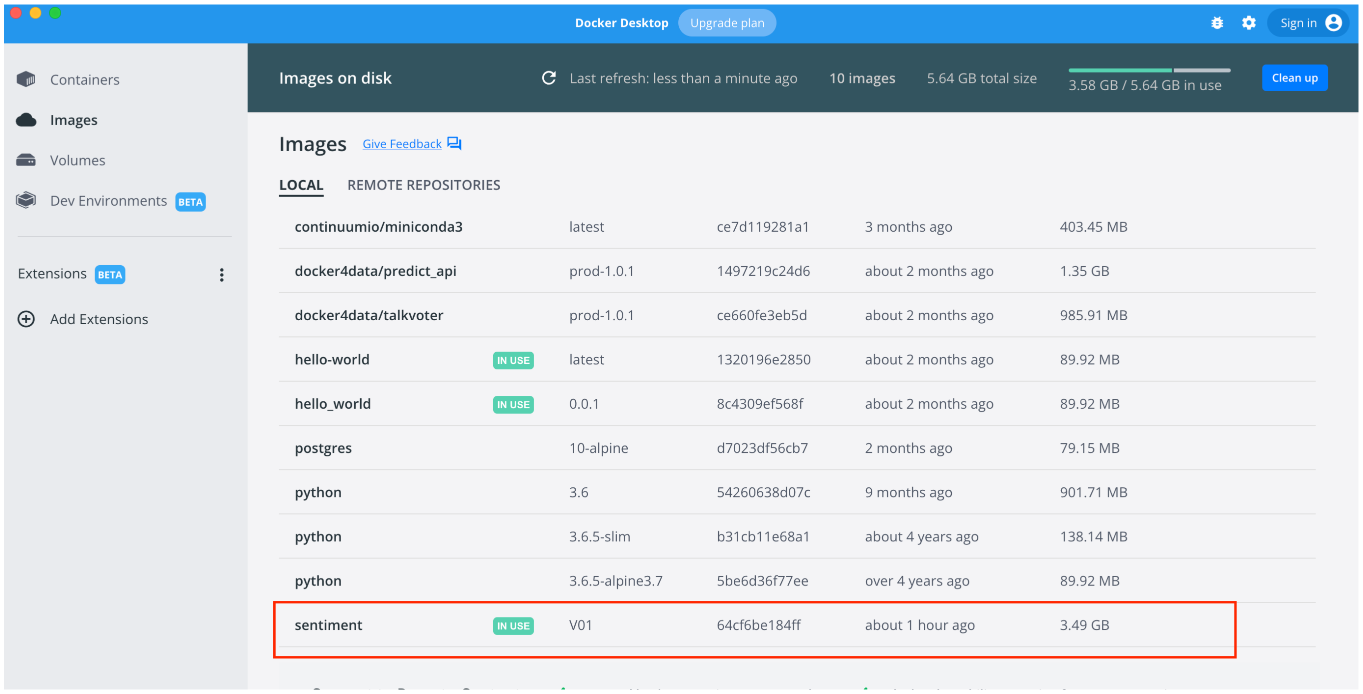
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Showing 3 items

NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
admiring_varahamihira 49e50d371023	hello-world:latest	Exited	-		
flamboyant_buck b89ad8b5695d	hello_world:0.0.1	Exited	-		
hopeful_shamir 2c2938e5d6cf	sentiment:V01	Running	5000	11 days ago	

Run the docker image which was built with previous command :

```
docker run -it --rm -p 5000:5000 sentiment:V01
```



Docker Desktop | Upgrade plan | Sign in

Containers | Images | Volumes | Dev Environments BETA | Extensions BETA | Add Extensions

Images on disk — Last refresh: less than a minute ago — 10 images — 5.64 GB total size — 3.58 GB / 5.64 GB in use — Clean up

Images Give Feedback

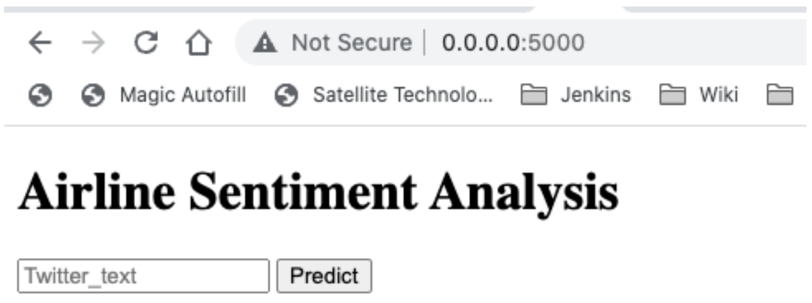
LOCAL | REMOTE REPOSITORIES

continuumio/miniconda3		latest	ce7d119281a1	3 months ago	403.45 MB
docker4data/predict_api		prod-1.0.1	1497219c24d6	about 2 months ago	1.35 GB
docker4data/talkvoter		prod-1.0.1	ce660fe3eb5d	about 2 months ago	985.91 MB
hello-world	IN USE	latest	1320196e2850	about 2 months ago	89.92 MB
hello_world	IN USE	0.0.1	8c4309ef568f	about 2 months ago	89.92 MB
postgres		10-alpine	d7023df56cb7	2 months ago	79.15 MB
python		3.6	54260638d07c	9 months ago	901.71 MB
python		3.6.5-slim	b31cb11e68a1	about 4 years ago	138.14 MB
python		3.6.5-alpine3.7	5be6d36f77ee	over 4 years ago	89.92 MB
sentiment	IN USE	V01	64cf6be184ff	about 1 hour ago	3.49 GB

Output when docker image command is run on console:

```
(myenv) simohanty@LP-SIMOHAN2-OSX api % docker run -it --rm -p 5000:5000
sentiment:V01
2022-09-23 23:43:18.094743: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
shared object file: No such file or directory
2022-09-23 23:43:18.094805: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror
if you do not have a GPU set up on your machine.
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [23/Sep/2022 23:43:22] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Sep/2022 23:43:37] "POST /predict HTTP/1.1" 200 -
172.17.0.1 - - [23/Sep/2022 23:45:23] "POST /predict HTTP/1.1" 200 -
```

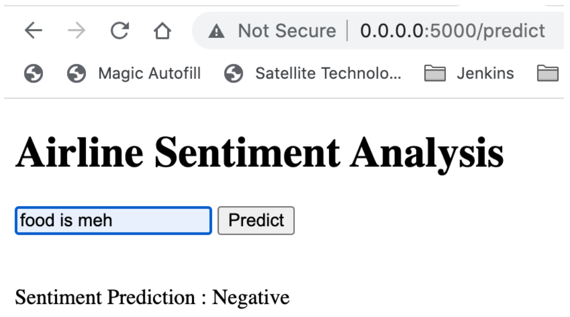
We exposed 0.0.0.0 and port 5000 as part of DockerFile config. So here if we navigate to http://0.0.0.0/5000 then we should be able to reach index.html.



Airline Sentiment Analysis

Results

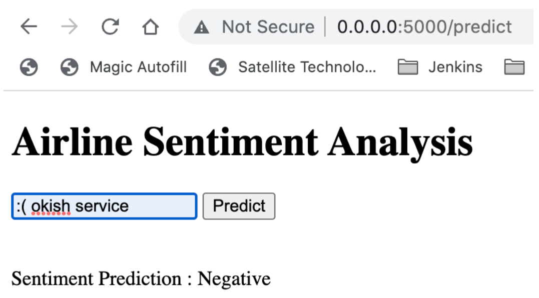
Here are some screenshots of the:



Airline Sentiment Analysis

food is meh

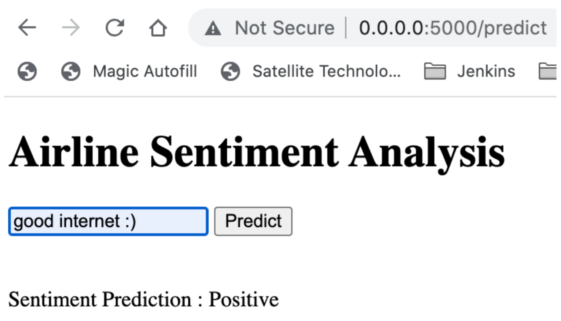
Sentiment Prediction : Negative



Airline Sentiment Analysis

:(okish service

Sentiment Prediction : Negative



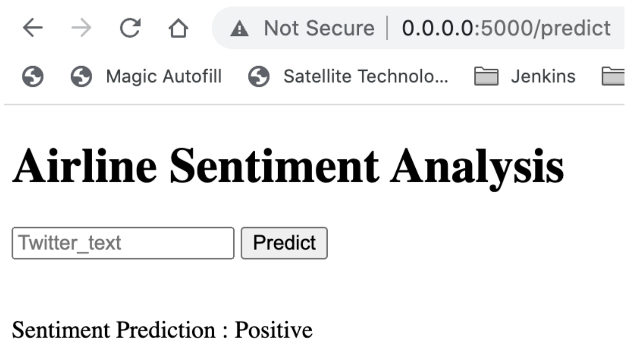
Not Secure | 0.0.0.0:5000/predict

Magic Autofill Satellite Technolo... Jenkins

Airline Sentiment Analysis

good internet :) Predict

Sentiment Prediction : Positive



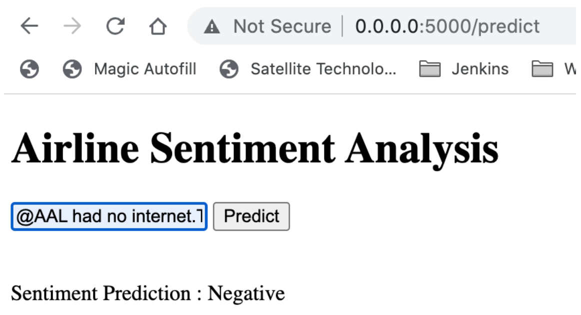
Not Secure | 0.0.0.0:5000/predict

Magic Autofill Satellite Technolo... Jenkins

Airline Sentiment Analysis

Twitter_text Predict

Sentiment Prediction : Positive



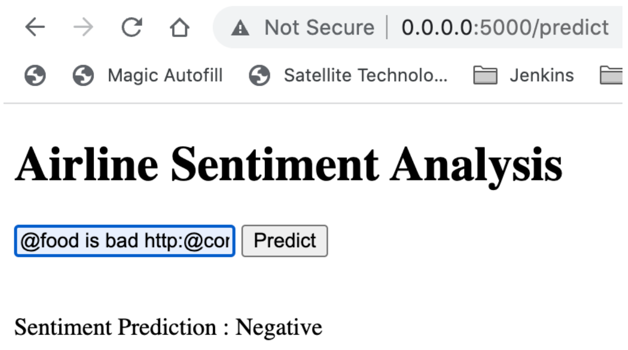
Not Secure | 0.0.0.0:5000/predict

Magic Autofill Satellite Technolo... Jenkins W

Airline Sentiment Analysis

@AAL had no internet.T Predict

Sentiment Prediction : Negative



Not Secure | 0.0.0.0:5000/predict

Magic Autofill Satellite Technolo... Jenkins

Airline Sentiment Analysis

@food is bad http:@cor Predict

Sentiment Prediction : Negative

GitHub Link

https://github.com/simohanty/Capstone/tree/master