Machine Learning Engineer Nanodegree

Capstone Project

Jitendra Reddy Muthyala
June 11th, 2017

I. Definition

Project Overview

Sentiment Analysis: Sentiment analysis is a task used to detect the sentiment in the user entered data (e.g. text data). In the present internet world, user opinions are playing vital role in advancements of any industry. By using sentiment analysis, we can detect the percentage of positivity over percentage of negativity in the user entered reviews. This way, we can have a insight on what people think about our products.

In this project, I created a Flask web application capable of classifying entered reviews into positive or negative. The application uses as classifier trained the Logistic Regression Model and IMDb reviews dataset.

This project was inspired by the KL Santhosh Kumars “Opinion mining and sentiment analysis on online customer review”, Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference.

Problem Statement

The agenda is to classify the polarity of a given text at the web app text box; the tasks involved are the following:

1. Download and preprocess the aclImdb_v1 dataset.
2. Train a classifier that can determine the polarity of a given text.
3. Make the classifier run on Flask.

4. Make the app to classify the polarity of entered movie review.
5. Print the results on the screen.

The final application is expected to predict the polarity of the user entered movie reviews.

Metrics

The metrics used in the project are:

1. Accuracy is a common metric for binary classification test. The accuracy is the proportion of true results among the total number of cases examined.

$$accuracy = \frac{true\ postives + true\ negatives}{dataset\ size}$$

   As my dataset is balanced, I used accuracy as a main evaluation metric to know how accurately my model will predict the output result.

2. Recall is the fraction of relevant instances that have been retrieved over total relevant instances.

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

   I used recall metric to know the percentage of true outputs of my regression model compared to the other two classification models used in the project.

3. Precision is the fraction of relevant instances among the retrieved instances.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

   I calculated this metric to know the exactness of my model in predicting polarity of reviews.

4. F-Score is a measure that combines precision and recall is the harmonic mean of precision and recall.

$$\mathrm{F} = 2 \,.\, \frac{\mathrm{precision} \,.\, \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

The main reason behind using F-score, recall and precision metrics is for comparing the performance of Logistic Regression model with SVM and Naïve Bayes models, that to convey why I used this model for my classification problem. As the basic rule to choose a best classification model is the recall and f-scores of the model should be higher and the precision value should be lower.

II. Analysis

Data Exploration

In this project we will be working on a large dataset of movie reviews for the Internet Movie Database (IMDb) which has collected by Maas et al. The movie dataset consists of 50,000 movie reviews that are labeled as either positive or negative; positive in the sense that a movie was rated with more than five starts on IMDb, and negative means movies rated below five stars on IMDb, that neutral reviews are not included (outliers) in the dataset. These reviews are used to study the user opinions by implementing Logistic Regression and Stochastic Gradient Descent models for classification of data.

A compressed archive of dataset (84.1MB) can be downloaded from http://ai.stanford.edu/~amaas/data/sentiment/ as a gzip compressed tarball archive.

Exploratory Visualization

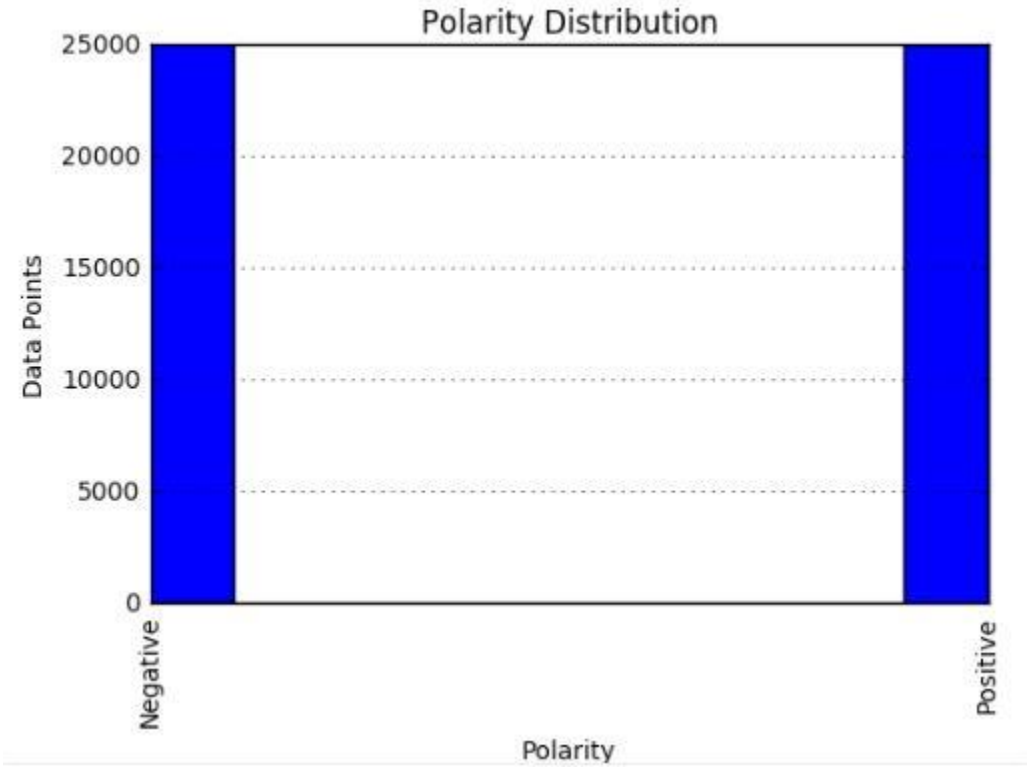
The table below shows the sample data of our dataset. The download data is in the form of text files and divided and directed to respective polarity subdirectories. The present form data is complex to explore, so there is a need to convert whole text files into a single CSV file using DataFrame. The converted data to be shuffled as the previous data is in sorted form. By using permutation methods the DataFrame data is shuffled and stored into a CSV file.

Table 1: Sample head data in data-frame.

	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0
3	hi for all the people who have seen this wonde...	1
4	I recently bought the DVD, forgetting just how...	0

The generated CSV file consists of two labels 'review' and 'sentiment', where the 'review' column consists of reviews of different users and 'sentiment' column consists of polarity of the user opinion.

The plot below shows the polarity distribution of reviews, that there are total 50,000 data points, in which 25,000 data points are positive reviews and the remaining 25,000 reviews are of negative reviews.



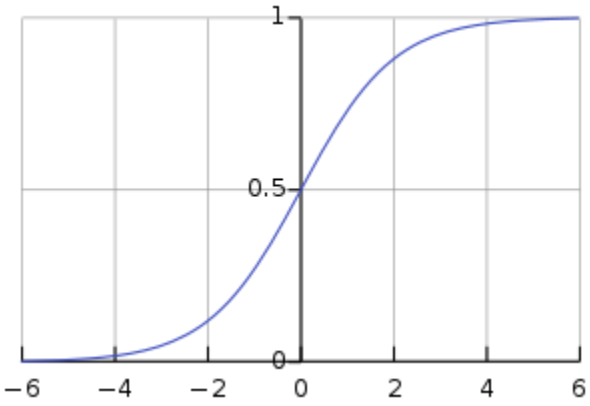
Algorithms and Techniques

Logistic Regression Algorithm:

Logistic regression is named for the function used at the core of the method, the sigmoid function which was developed by statisticians to describe properties of growth in different areas. It is am S-shaped curve which can take any real-valued number and map it into a value between 0 and 1, but not exactly at those boundaries.

$$\frac{1}{1 + e^{-x}}$$

Here e is the base of the natural logarithms and x is the actual numerical value that you want to transform. The graphical form of the above equation can be represented as:



I implemented Logistic Regression model with Stochastic Gradient Descent, which is an optimization algorithm that usually updates the model's weights using one sample at one shot. The hyper parameters to consider here are "alpha" the learning rate with defaults to 0.0001, "penalty" a regularization term (l2), and "n_iter" that the number of passes over the training data.

Support Vector Machine (SVM): Support Vector Machine algorithm is a supervised machine learning algorithm which is designed for classification and regression as well. In this algorithm, we plot each data sample as a point in the n-dimensional space with the value of each feature being the value of a

particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Tuning hyper parameter values for SVM will improves the performance, the important parameters having impact on model performance are “kernel”, “gamma” and “C”.

Where kernel takes low dimensional input space and transforms it to a higher dimensional space. Higher the gamma value, lower the generalization error and reduces over fitting problem. Finally the “C” parameter, it controls the trade-off between smooth and decision boundary and classifying the training points correctly.

Naïve Baye’s Algorithm: Naïve Bayes assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. This algorithm is known to outperform even highly sophisticated classification methods. Naïve Bayes classifier has limited options for parameter tuing like alpha=1 for smoothing and fit_prior=[True|False] to learn class prior probabilities or not. By using this algorithm it is easy and fast to predict the class of test data set. On the other side naïve bayes is also known as bad estimator.

This algorithm provides a way to calculate posterior probability P(c|x) form P(c), P(x) and P(x|c).

$$P(c|x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Where P(c|x) is the target (c) of class given attributes (x), P(c) is the prior probability of class, P(x|c) is the likelihood which is the probability of predictor given class and P(x) is the prior probability of predictor.

Bag of Words: I preferred Bag of Words technique to process my text into vectors as raw text cannot be fed into my model. I implemented **tokenizing** strings and giving an integer id and **counting** the occurrences of tokens in each document finally normalizing and weighting with diminishing important tokens that occur in the majority of the documents.

HashingVectorizer: To perform out-of-core scaling. The main advantage of using this class is we can learn from data that does not fit into the computer’s

main memory. The main agenda of implementing out-of-core scaling is to stream data to the estimator in mini-batches and by using HashingVectorizer the each mini-batch is vectorized. There is no data limit that can be handled by this technique, and the learning time is often limited by the CPU time one wants to spend on the task.

SQLite Database: SQLite is an embedded SQL database engine. Unlike other databases, it doesn't have a separate server process. It reads/writes directly to ordinary disk files.

Flask: Flask is a micro-framework for python to develop web applications. It is called micro-framework because it does not require particular tools or libraries. And also it has a wide range of extensions to add-on

Benchmark

The benchmark model of my algorithm is to predict the opinion in a submitted review with good accuracy. Hence a simple logistic regression model with 60% of classification accuracy will be considered as the Benchmark Model.

For the classification accuracy my goal was to achieve at least of 60% of accuracy. However I reached my goal I implemented a Classification model with 86% of classification accuracy.

III. Methodology

Data Preprocessing

The preprocessing is done in the prepare data notebook which consists of the following steps:

1. Unzipped data is loaded to process
2. Each text file present in the subdirectories of the dataset are loaded and appended to data-frame with label as well.
3. The appended data is columned with label names
4. The data-frame data is randomly shuffled with permutations function

5. The shuffled data-frame is converted into a single movie_reviews_data.csv file.

The prepared data is ready to load into the movie review classifier notebook for further classification and evaluation.

Implementation

The implementation process can be split into three main stages:

1. The classification stage
2. SQLite setup
3. The web application development stage

During the first stage, the classifier was trained on the preprocessed and cleaned data. This was done in a Jupyter notebook (titled "movie review classifier"), the following steps are processed in this section:

i. Load the preprocessed dataset into memory
ii. Plot the polarity distribution of data points
iii. Download stop-words of English language
iv. Implement text processing functions:
    a. tokenizer(….): Removes unwanted symbols and punctuations present in the text and converts into tokens
    b. stream(….): Reads in and returns one document at a time
    c. minibatch(….): Returns a particular number of documents specified by the size parameter in a single stream.
v. Define HashingVectorizer to convert tokenized to words into vectors.
vi. Define Classification models
vii. Loading stream of data
viii. Vectorize and train the models
ix. Evaluating the accuracies
x. Makeing predictions with test data
xi. Finding precision, recall and f-scores
xii. Plot the scores and comparing them
xiii. Choosing a efficient model
xiv. Save and freeze the current trained model

In the second step we will setup SQLite database for our web application, it includes the following steps:

i. Load the saved objects and de-serialize
ii. Connect to SQLite database
iii. Execute a table creation Quire
iv. Close the connection

In the final step will create a Flask web application to predict the polarity of entered review.

i. Create a Flask Web Application Template
ii. Load all the de-serialized objects into app.py
iii. Load database
iv. Define classifier
v. Train classifier
vi. Insert user entered reviews into database
vii. Implement a text area to enter reviews
viii. Predict and display the prediction in outcomes.py
ix. Return back to enter another revie

Refinement

As I mentioned in the Benchmark section, I implemented a Logistic regression model with GridSearchCV with five fold cross validation. Which produces a accuracy of 60% I unable to provide results as only one time executed and it took about 3hours with lots of kernel crashes. I also implemented SVM model in Stochastic Gradient Decent Classifier, where I like to use 'penalty', 'random_state' and 'n_iter' hyper parameters in my model which produced classification accuracy of 0.87 with default values. And in Naïve Bayes classification models I don't want to tune any hyper parameters as it is that important in this model it produced 0.8 of classification accuracy.

For my final model I implemented Logistic Regression using Stochastic Gradient Decent Classifier I tried tuning the learning rate 'alpha' with different learning rates the results are tabulated below, but I achieved maximum

classification accuracy of 0.87 with default learning rate. I set 'n_iter' parameter to 1 as I am using 'partial_fit' to fit my model, that 'n_iter' parameter is the number of epochs over the training data. The scores of the three models used in the project are compared in the below fig 3.

Table 1: Logistic Regression classification accuracies with different learning rates

Learning Rate	Classification Accuracy
0.1	0.71
0.00001	0.51
0.0001 (Default)	0.86

Fig 2: Classification accuracies of the three classification models

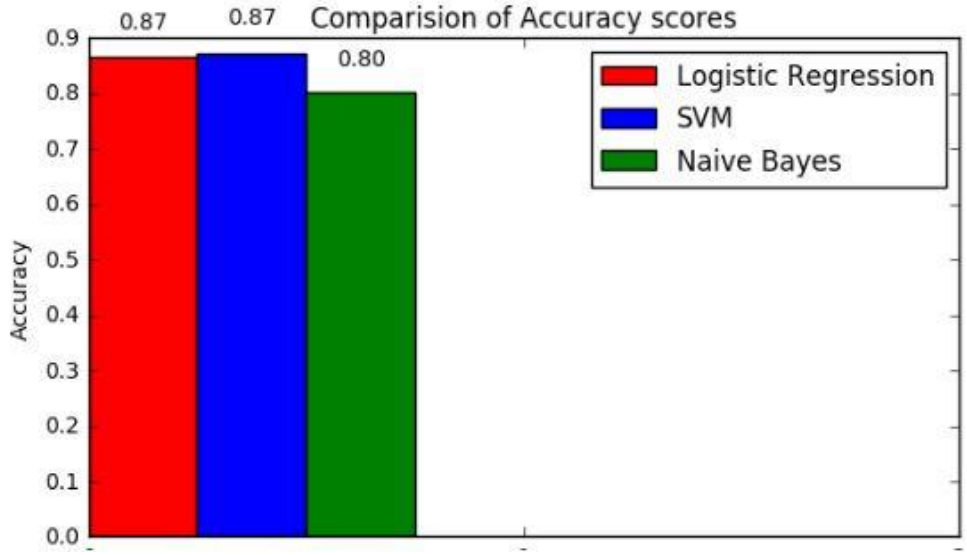
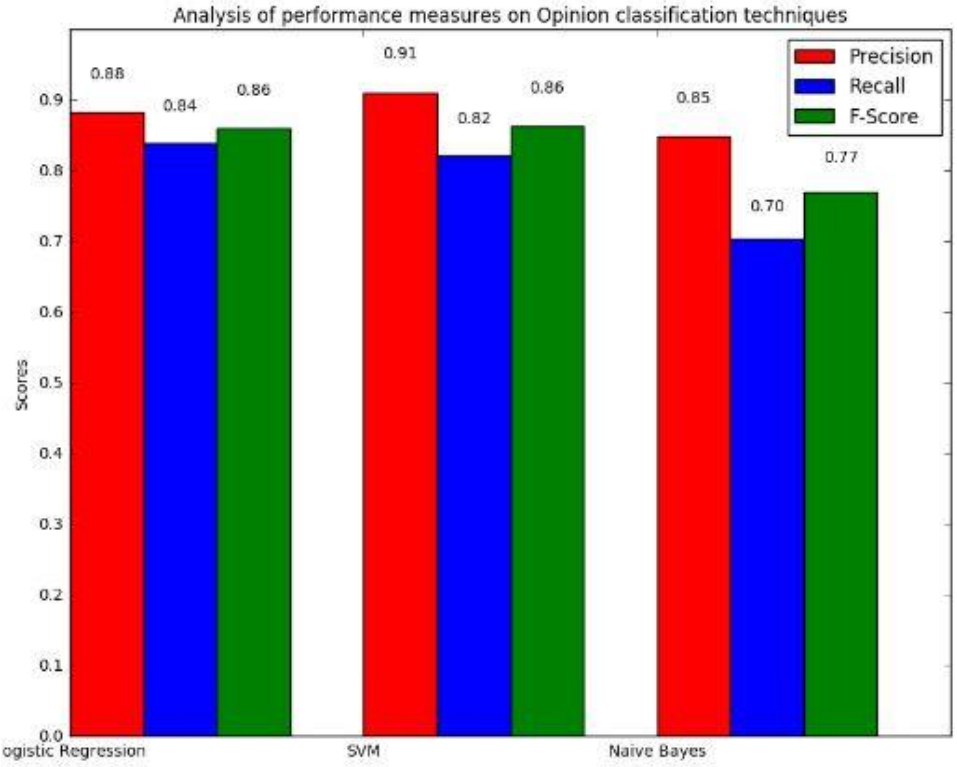


Fig 3: Scores produced by the Logistic Regression, SVM and Naïve Bayes models.



IV. Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The final model is chosen as it is predicting the opinions with better accuracy.

- The tokenized text is converted into vectors using HashingVectorizer class, which consists of 2**21 n_features.
- The Logistic Regression model with default learning rate is initialized
- The 45,000 documents of the 50,000 documents of dataset are used to train our model.
- Mini batches of 1000 documents are iterated for total 45 iterations.

- The remaining 5000 documents are used to predict the accuracy of our model using as a test set.
- It produced a classification accuracy of 86%.

To verify the robustness of the final model, I tested with different inputs in my web application. The results are included in fig 5. The following results are based on the results of the test:

- The classifier can reliably predicts the opinion in the entered reviews
- Tried with neutral reviews, but the classifier predicted (positive or negative) based on the weights of the words used in review.
- The accuracy in each prediction is meaningful

Some of the predictions made by the classifier are shown in below figures.

Justification

Using a simple Logistic Regression Model on a PC with normal CPU, I got these results:

- The classification delay is of few minutes, which is better than the benchmark model.
- The classification accuracy is of 86%, which is also greater than the proposed Benchmark model.
- The application is making good predictions of the user opinions as well.

To understand how successful the final application is, we need test different type of reviews on the model, which is illustrated in the below figures.

This application is useful in a limited domain, but to predict in more accurate way we need to drain the model using deep learning techniques with better hardware.

V. Conclusion

Free-Form Visualization

Fig2: The precision, recall and f-scores calculated for the Logistic Regression, SVM and Naïve Bayes classification models. I choose Logistic Regression model for my further classification as it showed better scores compared to remaining models.

Fig 4: Comparing scores of the three models used.

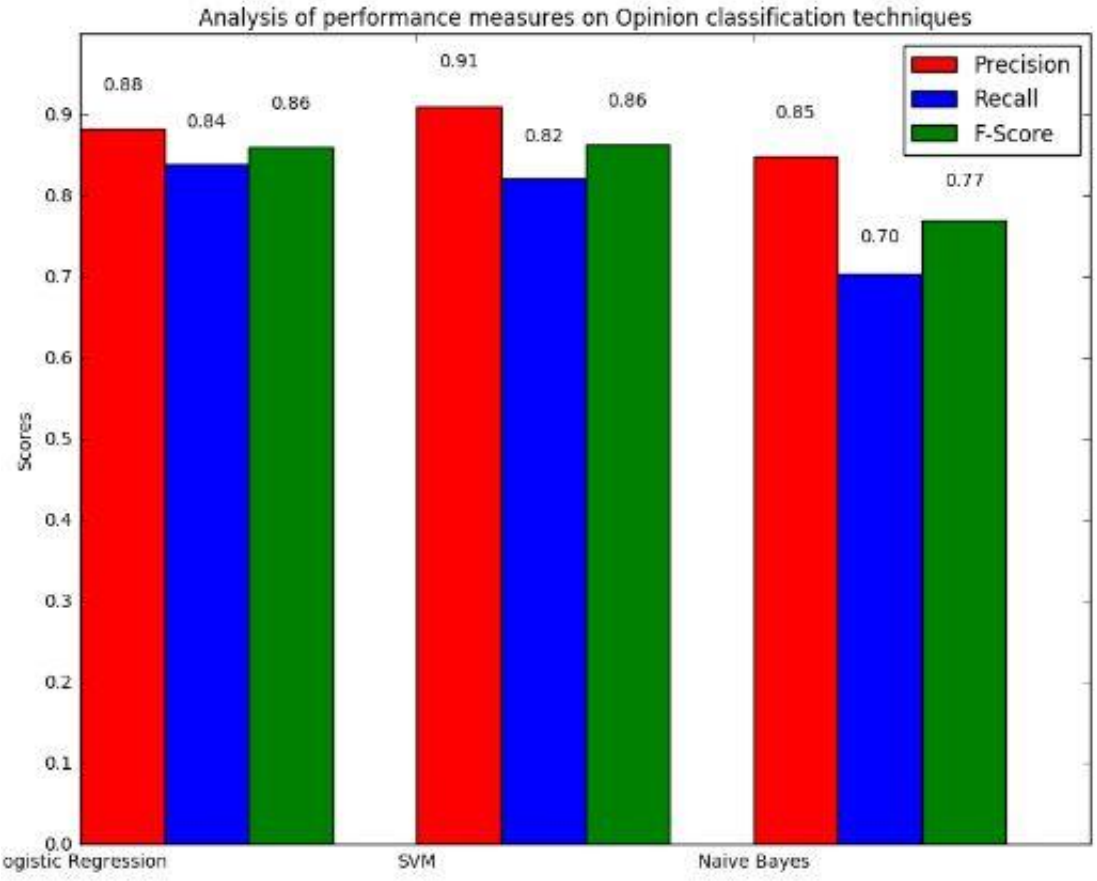
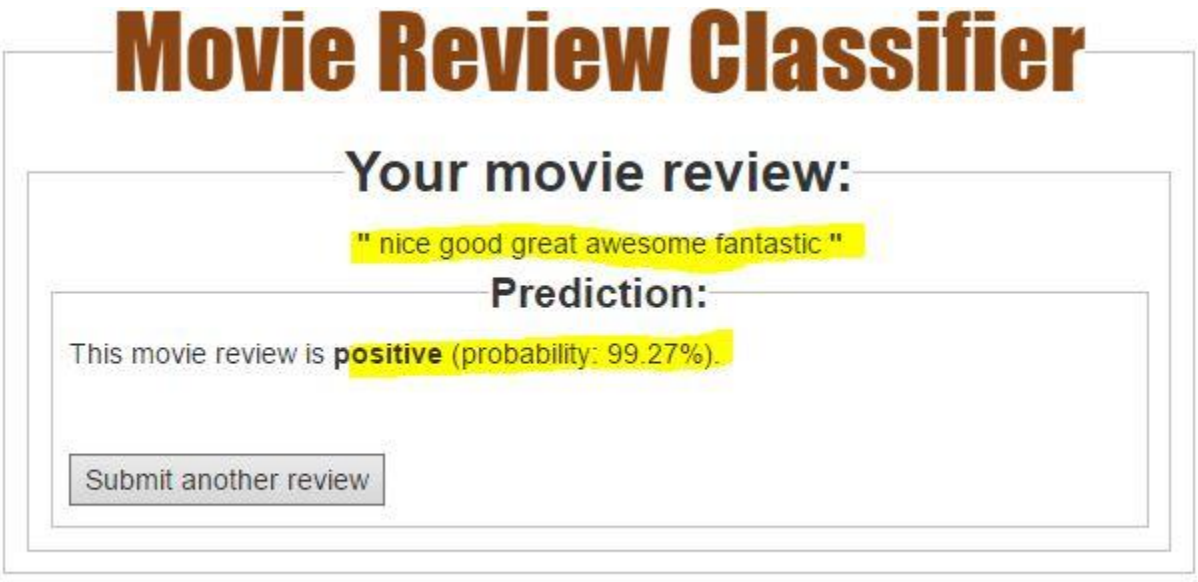
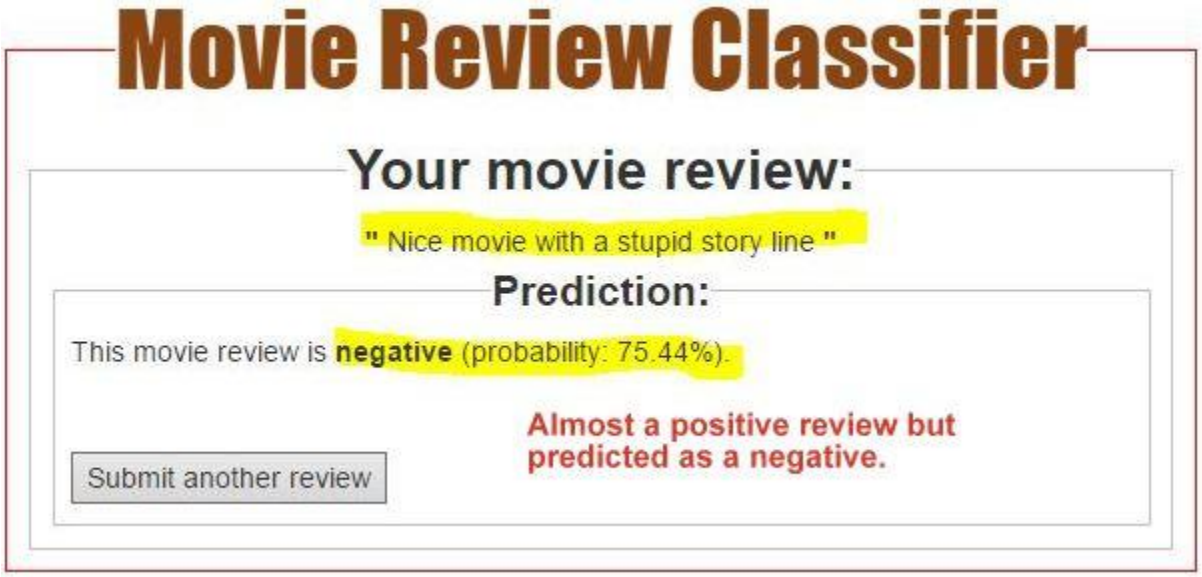


Fig 5: This figures shows the final models predictions made on a Flask Web Application





Reflection

The process used for this project can be summarized using following steps:

1. A good task and relevant public datasets were found
2. The data was downloaded and preprocessed
3. A benchmark was set for the classifier
4. The classifier was trained using the data until it reaches the desired output.

5. The model is saved to local disk for further use without recompiling
6. The database is set using SQLite
7. A Movie Review Classifier Flask web application is created to predict the polarity of newly entered reviews.

I found difficulty in training Benchmark model as my hardware didn't supported much for my complex model and I felt interesting in fifth step as there are no more recompilations to run a model. Before this project I found more difficulty in Natural Language Processing, but after this project I am very confident on NLP and I want to research more on it use deep learning techniques.

Improvement

To achieve optimal user experience, using more capable hardware like GPUs and by using word2vec or GloVe for vectorization of tokens and instead of linear models like Logistic Regression using Random Forest or Convolution Neural Networks will be a good option to produce more prudent results. The same model can be implemented in other languages other than English to predict the reviews of movies, restaurants, products and lots more.

If I consider this model as a new Benchmark model, I can surpass this by using deep learning techniques like Convolution Neural Networks with less classification delay and with more accuracy.