Amazon Reviews For Sentiment Analysis

ML Project

System Vampires
Batch No : 2

April 20,2022

Team Members

19B01A0583 - K.Lovely Srenika - CSE
19B01A0594 - M.Alekhya - CSE
19B01A05A2 - V.Harshita Sai - CSE
19B01A0478 - K.S.L.Katyayini - ECE
19B01A0573 - K.Devi Meghana - CSE

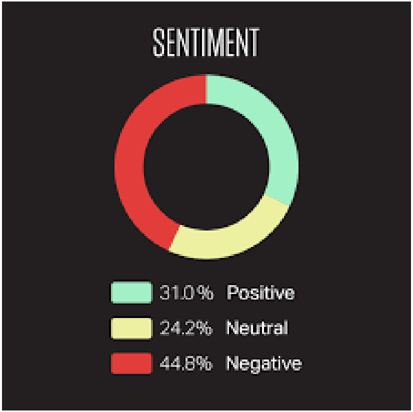
Outline of the project

PROJECT

- Problem Description
- Input
- Output
- Problem Statement
- Architecture Diagram
- Work Schedule
- Tech Stack and Learnings
- Challenges
- Project Demo and Code
- Git Repo Link
- Statistics

Problem Description

The aim of this project is to build a sentiment analysis model on Amazon reviews which will allow us to categorize words based on their sentiments,that is whether they are positive,neutral or negative.



Input

The input dataset consists of a few million Amazon customer reviews (input text) and star ratings (output labels) for learning how to train fastText for sentiment analysis.

The idea here is a dataset is more than a toy - real business data on a reasonable scale - but can be trained in minutes on a modest laptop.

Sentiment analysis

Output

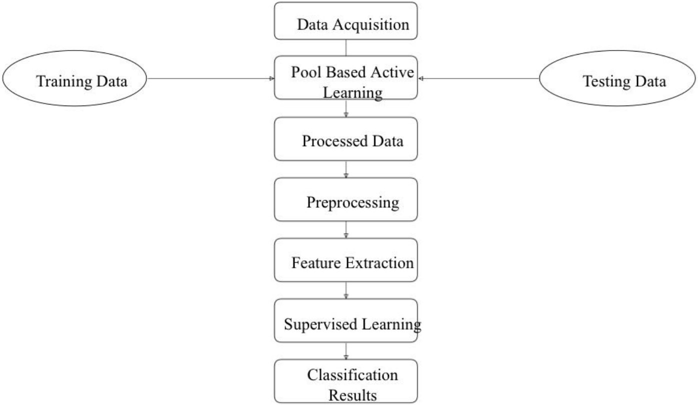
For each amazon customer review the output should be whether the review is positive, neutral or negative.

Problem Statement

From the description of the Amazon Reviews using Sentiment Analysis we understood that it’s a model where the amazon review is given as input and this model should predict whether the review is positive, neutral or negative based on their words.

Supervised Learning: Supervised learning is the types of machine learning in which machines are trained using well ”labelled” training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly

Architecture Diagram



Work Schedule

- First Week and Second week - Problem Solution Identifying

(sentiment analysis, dataset ,supervised learning and its techniques like knn , decision trees, random forest, svm)

And Developed Logistic Regression Model.

- Third week - Researched about the remaining three models - SVM, Random Forest, Decision Tree.

Tech Stack and Learnings

- MikTex and TexMaker are used to make this presentation

Learnings:

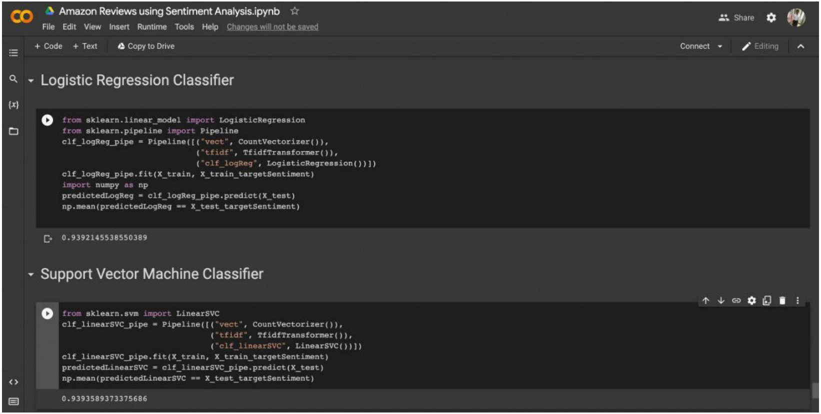
- We learnt how to use latex from the scratch and make presentation out of it.

Challenges we have faced

- Developing all the four models - Overcame by researching about those models.

- Accuracy of the models - Overcame by rectifying our mistakes in training part.

Project Demo and Code


CO
Amazon Reviews using Sentiment Analysis.ipynb ☆
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text
Copy to Drive
Connect
Editing
{x}

Logistic Regression Classifier

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
clf_logReg_pipe = Pipeline([("vect", CountVectorizer()),
                            ("tfidf", TfidfTransformer()),
                            ("clf_logReg", LogisticRegression())])
clf_logReg_pipe.fit(X_train, X_train_targetSentiment)
import numpy as np
predictedLogReg = clf_logReg_pipe.predict(X_test)
np.mean(predictedLogReg == X_test_targetSentiment)
```

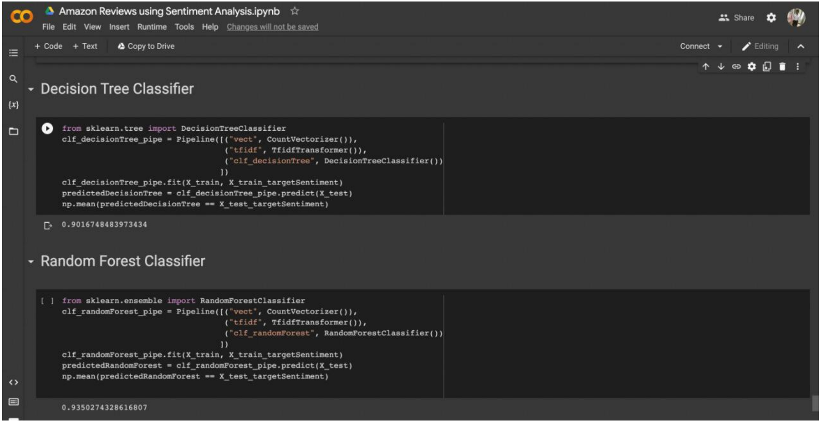
```
0.9392145538550389
```

Support Vector Machine Classifier

```
from sklearn.svm import LinearSVC
clf_linearSVC_pipe = Pipeline([("vect", CountVectorizer()),
                               ("tfidf", TfidfTransformer()),
                               ("clf_linearSVC", LinearSVC())])
clf_linearSVC_pipe.fit(X_train, X_train_targetSentiment)
predictedLinearSVC = clf_linearSVC_pipe.predict(X_test)
np.mean(predictedLinearSVC == X_test_targetSentiment)
```

```
0.9393589373375686
```

Project Demo and Code


Amazon Reviews using Sentiment Analysis.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
clf_decisionTree_pipe = Pipeline([("vect", CountVectorizer()),
                                  ("tfidf", TfidfTransformer()),
                                  ("clf_decisionTree", DecisionTreeClassifier())
                                 ])
clf_decisionTree_pipe.fit(X_train, X_train_targetSentiment)
predictedDecisionTree = clf_decisionTree_pipe.predict(X_test)
np.mean(predictedDecisionTree == X_test_targetSentiment)
```

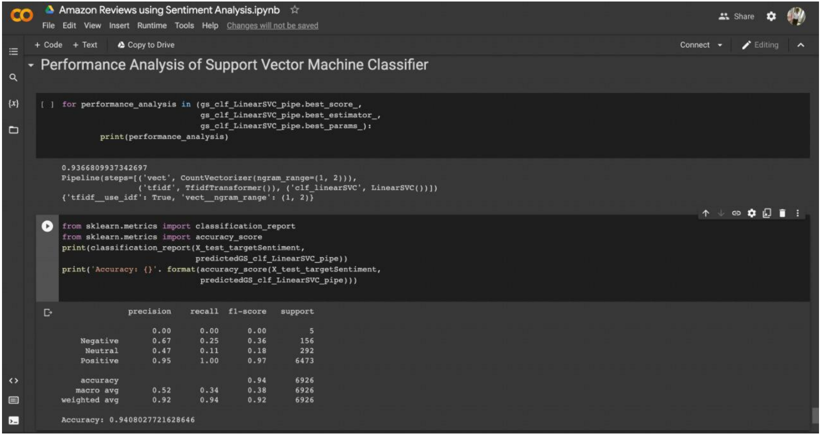
```
0.9016748483973434
```

Random Forest Classifier

```
[ ] from sklearn.ensemble import RandomForestClassifier
    clf_randomForest_pipe = Pipeline([("vect", CountVectorizer()),
                                      ("tfidf", TfidfTransformer()),
                                      ("clf_randomForest", RandomForestClassifier())
                                     ])
    clf_randomForest_pipe.fit(X_train, X_train_targetSentiment)
    predictedRandomForest = clf_randomForest_pipe.predict(X_test)
    np.mean(predictedRandomForest == X_test_targetSentiment)
```

```
0.9350274328616807
```

Project Demo and Code



Performance Analysis of Support Vector Machine Classifier

```
[ ] for performance_analysis in (gs_clf_LinearSVC_pipe.best_score_,
                                 gs_clf_LinearSVC_pipe.best_estimator_,
                                 gs_clf_LinearSVC_pipe.best_params_):
        print(performance_analysis)
```

```
0.9366809937342697
Pipeline(steps=[('vect', CountVectorizer(ngram_range=(1, 2))),
                ('tfidf', TfidfTransformer()), ('clf_linearSVC', LinearSVC())])
{'tfidf__use_idf': True, 'vect__ngram_range': (1, 2)}
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(classification_report(X_test_targetSentiment,
                            predictedGS_clf_LinearSVC_pipe))
print('Accuracy: {}'. format(accuracy_score(X_test_targetSentiment,
                             predictedGS_clf_LinearSVC_pipe)))
```

```
              precision    recall  f1-score   support

                   0.00      0.00      0.00         5
    Negative       0.67      0.25      0.36       156
     Neutral       0.47      0.11      0.18       292
    Positive       0.95      1.00      0.97      6473

    accuracy                           0.94      6926
   macro avg       0.52      0.34      0.38      6926
weighted avg       0.92      0.94      0.92      6926

Accuracy: 0.9408027721628646
```

Git Repo Link

https://gitlab.com/python-pirates/amazon-reviews-using-sentiment-analysis.git

Project Statistics

- Number of Lines of code - 272.
- Number of Algorithms - 4.
- Number of Functions - 1.

The End