**CIS4500 Term Project**

**Name: Abir Abbas**

**Student ID: 0955448**

**Date: 11/28/2019**

**Introduction**

The problem is to classify movie reviews as positive or negative where positive represents a review that is supporting the movie and negative represents a review that is criticizing the movie. This classification problem can help solve many requirements for large movie reviewing websites, it can rank movies according to their movie sentiment and promote movies accordingly. Furthermore, it provides valuable information for the audience to quickly analyze the sentiment on a movie and make decisions on whether the movie is worth watching or not. For example, if our model has classified 70% of the total movie reviews as negative then it is very likely the movie is not worth watching and IMDB should not promote it.

To solve this problem, we are going to experiment with two feature selection methods (Count vectorizers and TFIDF weighting), three classification methods (Logistic Regression, Ridge Classifier and Nearest Centroid). The dataset we will be using for this model is a movie review dataset released at Cornell University in 2004. The dataset is split up into positive and negative reviews in no order. The format of the files is .txt and will be read in and combined into one giant list of documents and their sentiment.

**Description of methods**

**Count Vectorization**

The count vectorizing feature selection method converts a collection of text documents to a matrix of token counts. The matrix is representative of the tokens present in the list of documents and displays whether each document contains that specific word.

Example:

Corpus = [‘This is a test’, ‘A test it is’]

Tokens = [‘this’, ‘is’, a’, ‘test’, ‘it’]

CountVector = [ [1, 1, 1, 1, 0], [0, 1, 1, 1, 1] ]

**TFIDF Weighting**

The tf-idf weight is a statistical measure to evaluate the importance of a specific token within a document from a collection. The importance of a word is determined by evaluating the number of times the token appears in the current document in comparison to the number of times the word appears in all documents in the collection.

Example:

Corpus = [‘This is a test’, ‘A big test it is’]

In this corpus tokens such as ‘test’ and ‘a’ are common in both datasets and will be given a lower weighting. Words such as ‘this’ and ‘big’ will be given a higher weighting because they only appear exclusively on one document.

**Logistic Regression**

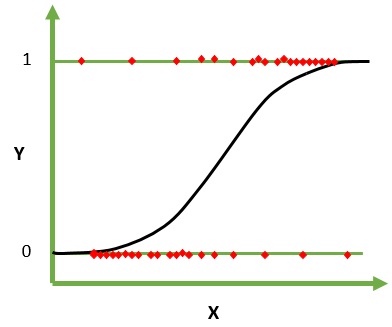
Logistic regression is a statistical approach to solve binary classification problems. It only works if the dependent can be one of two classes (1 or 0, Yes or No, Positive or Negative…). The logistic regression model uses the sigmoid function to split up mass amounts of data into two possible classes using probabilities. The sigmoid function and logistic regression can be visualized in the following image.

Figure Graph representing the sigmoid function used in logistic regression

**Ridge Classifier**

The ridge classifier is a statistical approach to solve linear problems with high multi-collinearity. The ride regression method is very similar to the linear regression method with the key difference being adding bias to explanatory variables. This means that any two highly correlated variables, the model will add a bias to one of the variables and adjust the model accordingly. This ensures the model is not trying to fit against two highly correlated variables, instead it selects one of the variables based on the circumstances and adjusts the regression accordingly. This ridge regression added bias can be visualized in the image below where bias was added to the green nodes and the model was adjusted accordingly.

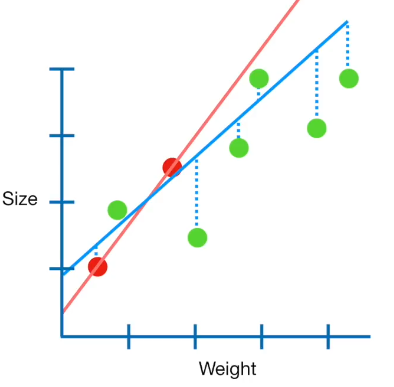


Figure Ridge regression bias demonstrated on the green nodes

**Nearest Centroid**

Nearest centroid is a distance-based approach to solve classification problems. The distance calculation is flexible and different methods can be used. By default, the Euclidian distance calculation method is used by the nearest centroid. The basis of the method is to randomly place a several centroids and iteratively move the centroids towards a group of nodes using Euclidian distance between the centroid and the nodes closest to it. This is continuously done till the centroids are having very little movement if any at all. This can be visualized in the image below where the X represents the centroids and the colors represent the 3 different class.

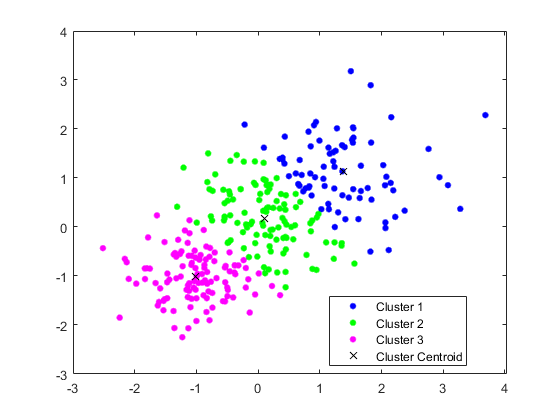
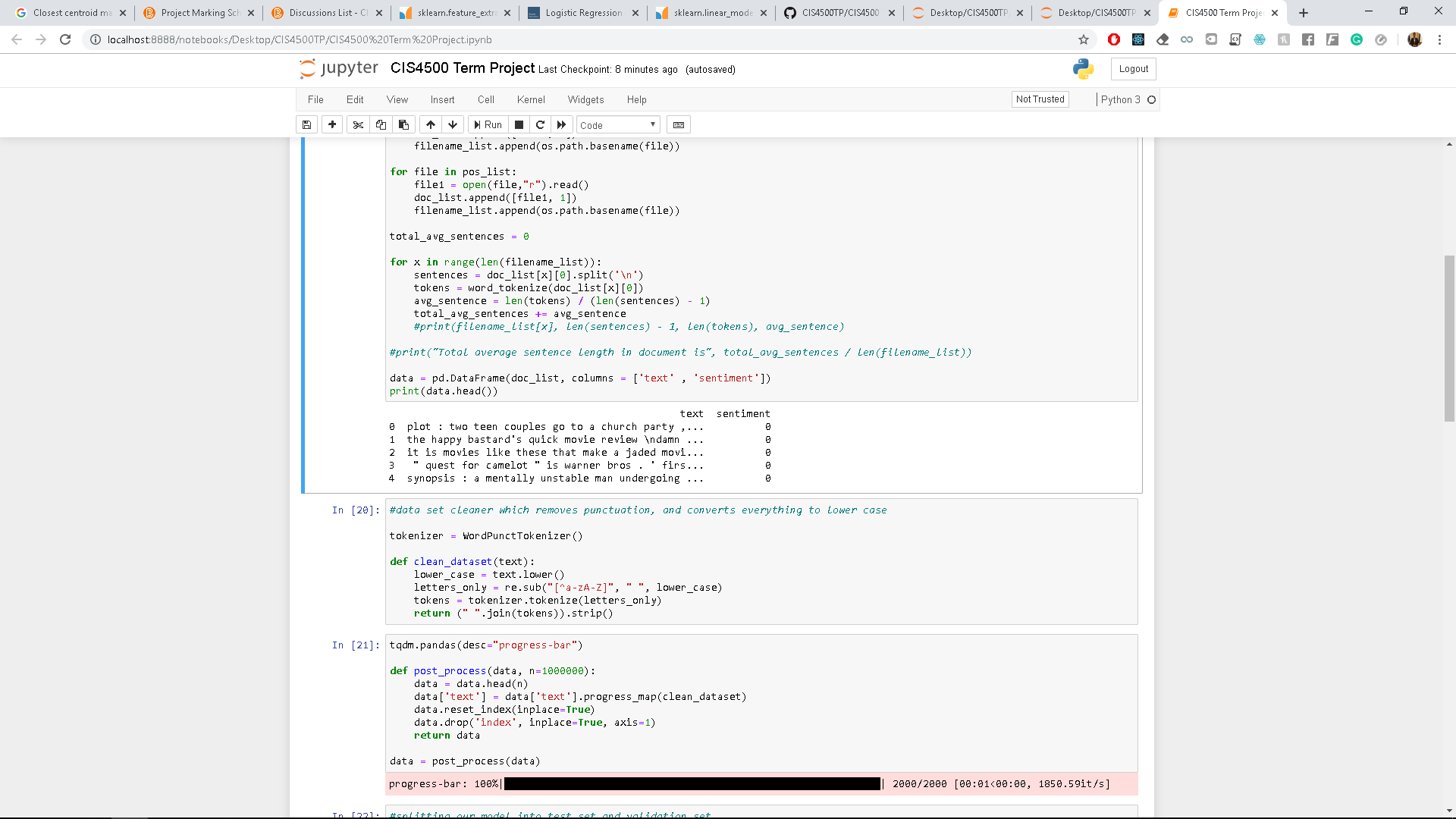


Figure A simple example of the outcome of running several iterations of the Nearest Centroid method

**Highlight of Implementation**

**Data structures**

The data is combined into one large array of the form [[‘text’, ‘sentiment’], **…**]

The dataset is then converted into a Python DataFrame that looks like the following

The data contains 2000 records in total, 1000 negative and 1000 positive reviews respectively.

**Train, Validation and Test sets**

The data is initially split up into train set (85% \* 2000 = 1700 rows) and Validation set (15% \* 2000 = 300 rows)

For cross-validation we will further split up the train set into a new train set (90% \* 1700 = 1530 rows) and test set (10% \* 1700 = 170 rows), this is because we are using 10-fold cross-validation

**Python libraries**

Pandas: A library that provides data structures and methods to read, analyze and manipulate datasets

Natural Language Toolkit: A suite of libraries and programs for symbolic and statistical natural language processing for English written in Python programming language.

SciKit-Learn: Machine learning for python. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, and is designed to interoperate with Python numerical and scientific NumPy and SciPy.

**Assumptions**

* The documents are independent of one another
* The language used in the text is part of the English language
* The documents have been accurately classified as positive or negative

**Limitations**

* There is a limited amount of data (only 2000 documents)
* Any words that are not present in the dataset will significantly impact the performance of the model. I.E. the world horrible does not exist anywhere in the dataset, so the statement “The movie was horrible” would not return an accurate result.
* When using the Python function cross\_val\_score we don’t have access to the trained models used to calculate the scores

**Design decisions**

* Because we don’t have access to the models used to evaluate the cross\_val\_score, we can simply just train one final model with the 85% train set that was allocated initially and will be tested on the validation set.
* We will only remove stop-words as the dataset is already partially cleaned
* A K-Folding of size 10 will be used because we have such a small subset of data.
* We will select Logistic Regression, Ridge Classifier and Nearest Centroid because they are commonly used in natural language processing

**Data, Results and Analysis**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **LG\_CV** | **LG\_IDF** | **RC\_CV** | **RC\_IDF** | **NC\_CV** | **NC\_IDF** |
| **500** | 0.742 | 0.786 | 0.740 | 0.769 | 0.653 | 0.762 |
| **1000** | 0.791 | 0.817 | 0.694 | 0.816 | 0.662 | 0.782 |
| **1500** | 0.799 | 0.840 | 0.685 | 0.831 | 0.664 | 0.797 |
| **2000** | 0.813 | 0.847 | 0.731 | 0.836 | 0.664 | 0.807 |
| **2500** | 0.816 | 0.844 | 0.748 | 0.839 | 0.669 | 0.805 |
| **3000** | 0.812 | 0.841 | 0.765 | 0.836 | 0.673 | 0.802 |

Based on the results it becomes very evident that tf-idf feature selection yields better results than the count vectorization feature selection method.

Furthermore, increasing the number of maximum features selected gradually improves the performance of the models up until the 2000 features mark. Beyond that we see very marginal improvements in performance, and in some cases a decrease in performance. However, increasing the number of features we look at also increases the time and complexity of the model overall. Therefore, our most favorable selection would be somewhere between 2000 and 2500 maximum features.

Between the 3 models that were used it becomes evident that Logistic Regression and Ridge Classifiers both provide very similar performance. Although logistic regression slightly beats out the ridge classifier.

**Judging from our results above we can see that the logistic regression using the tf-idf feature selection with maximum features of 2000 method yields the best result.**

**We can also observe the Ridge classifier using the TFIDF feature selection with maximum features of 2500 also yields some good results as well.**

**Conclusion and Future Improvements**

Using the last two models we selected using our previous analysis, we can observe both models perform well with an accuracy around 83-84%. Both models boast very similar accuracy and performance which can be seen in the two confusion matrices below.

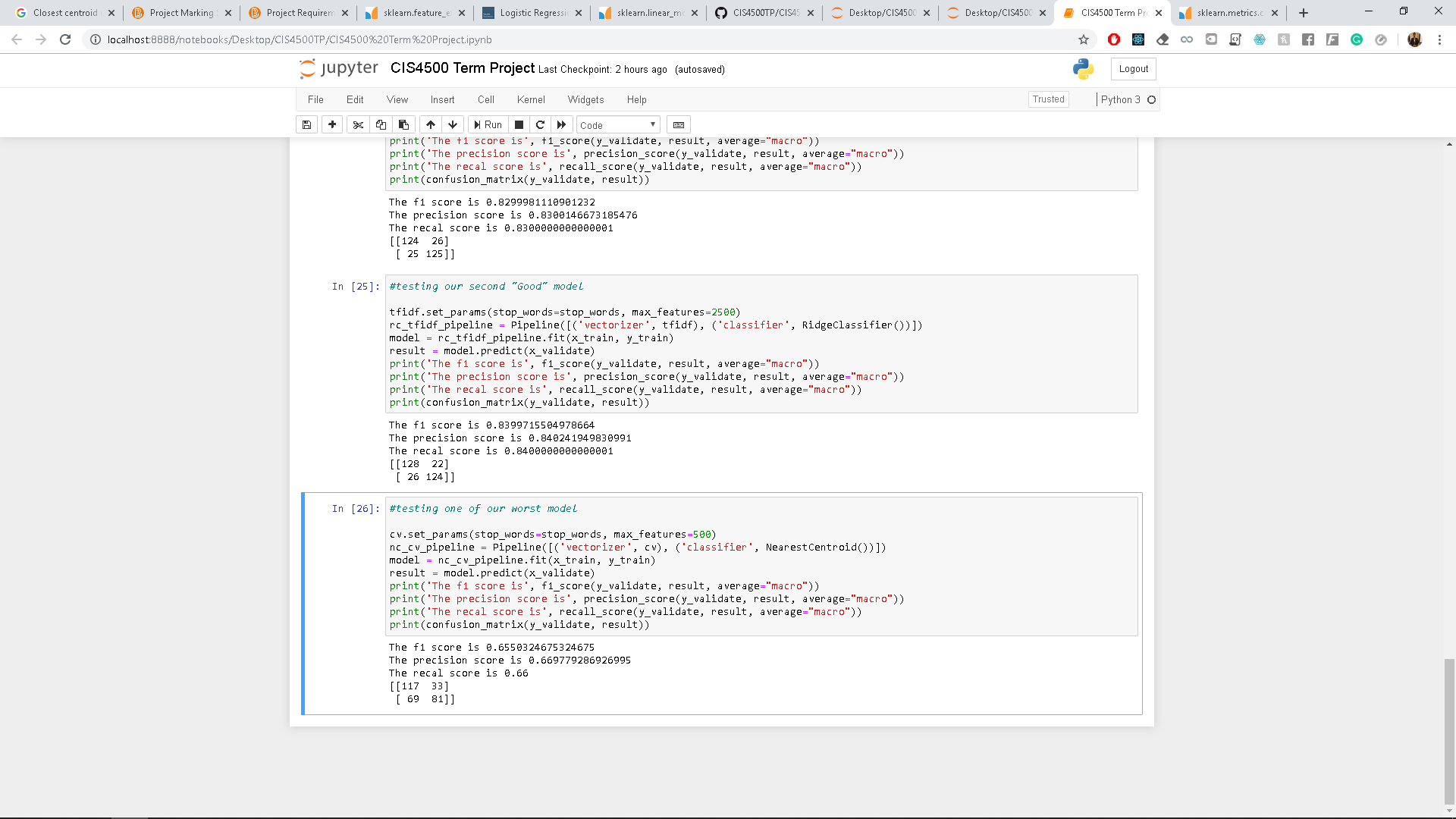


Figure The confusion matrix for the logistic regression model

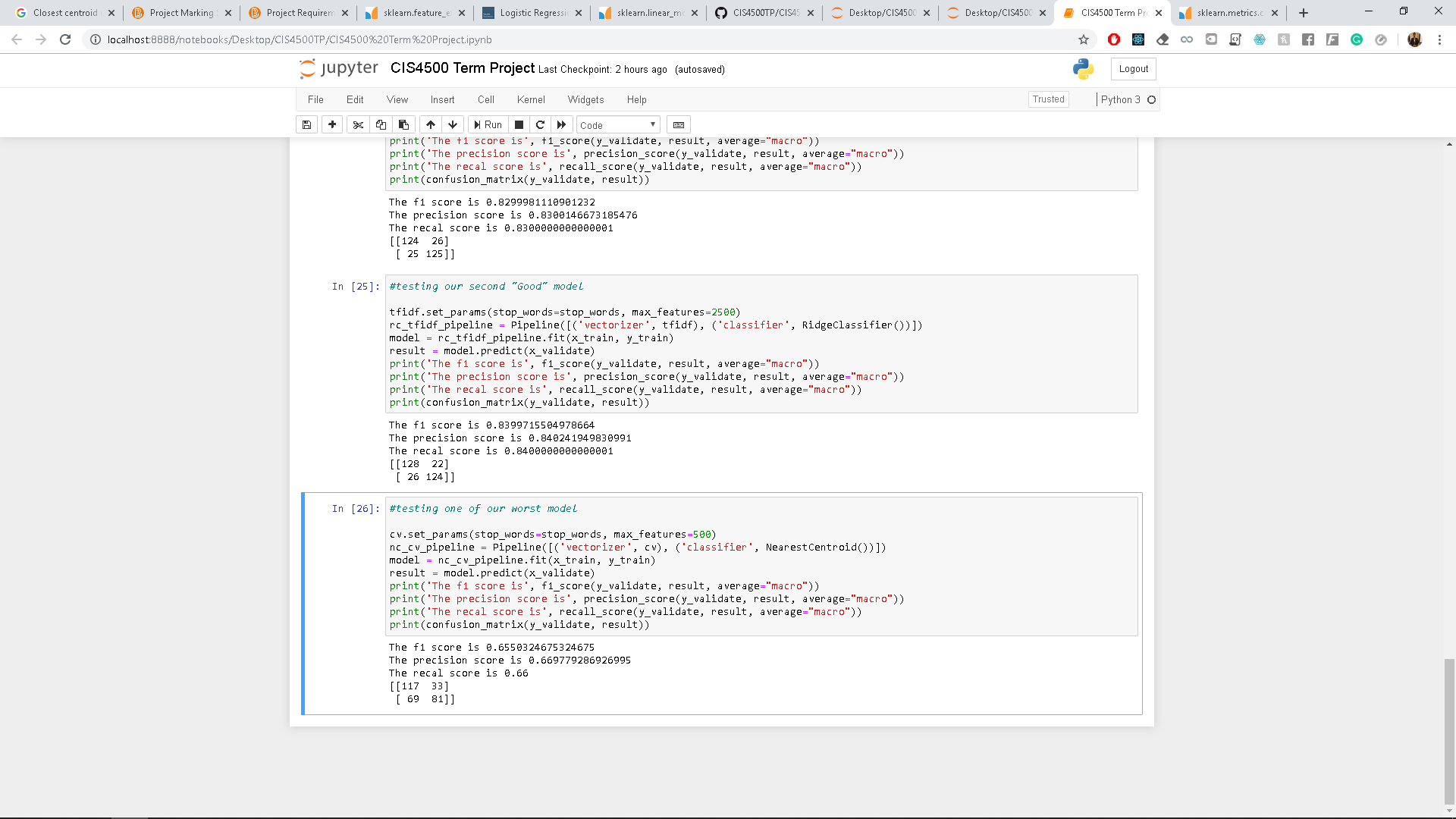


Figure The confusion matrix for the ridge classifier model

Analyzing these results, we can use this model to predict general sentiment of a movie based on its reviews. Although, I would not use this model for predicting a small amount of reviews as it does not perform very well with unknown words. Overall, the model could be useful for general sentiment analysis on a group of reviews but the predictions should not be used individually as the result may not be very useful.

**Build and installation guide**

Requirements:

Python 3.7 : <https://www.python.org/downloads/release/python-370/>

Jupyter Notebook : <https://jupyter.org/>

Anaconda Python Distribution: <https://www.anaconda.com/distribution/> f

Build:

To start Jupyter notebook refer to the Jupyter notebook docs. (On windows you can simply run the application Jupyter notebook and it will open a browser and redirect you to your locally running Jupyter notebook webapp).

Once Jupyter notebook is running you can navigate through the directories using the Jupyter notebook UI and locate the folder which contains the CIS4500\_Term\_Project.ipynb file and open it up.

You can now individually run each cell in the Jupyter notebook and walk through the entire process of selecting and building the model.

In case you have any errors such as “Could not locate module” you must install the respective libraries in your Python distribution. This can be done easily by opening the Anaconda terminal (On windows opening the Anaconda command prompt) and typing “conda install [module name]”

**References**

A Single-Page Tutorial - Information Retrieval and Text Mining. (n.d.). Retrieved from <http://www.tfidf.com/>.

Paul, Paul, Rick, Rick, Hegde, V., Hegde, V., … Ansari, A. (2018, April 25). Difference Between Linear and Logistic Regression (with Comparison Chart). Retrieved from <https://techdifferences.com/difference-between-linear-and-logistic-regression.html>.

Why ridge regression only decreases slope and not increases it? Retrieved from https://stats.stackexchange.com/questions/402889/why-ridge-regression-only-decreases-slope-and-not-increases-it.

Hourrane, O. (2018, September 15). Sentiment Analysis using Python (Part I - Machine learning model comparison). Retrieved from <https://datasciencetoday.net/index.php/en-us/tutorials-top/163-sentiment-analysis-using-python-part-i-machine-learning-model-comparison?fbclid=IwAR3ZA8MN-1NHHnMIBqc3KdaXkka6msPZG13ZoT91sickDOPb_g9l8F1NBfY>.