Shivangi Mundhra
SUID - 842548148

Homework 8

Shivangi Mundhra | SUID – 842548148 | 11/04/2022

SECTION 1 - INTRODUCTION

In this homework, I am going to try to test the claim that machine learning algorithms can figure out whether a person is lying or not while also trying to predict the sentiment of a statement. I am going to walk through all the aspects of this assignment including what the assignment is and what approaches I am using. In the analysis itself, I will try to explain all reasoning and thought process behind each significant step. The intent is that any person, especially one without technical knowledge and that is not familiar with this problem statement, should be able to read this report and follow along.

About the assignment –

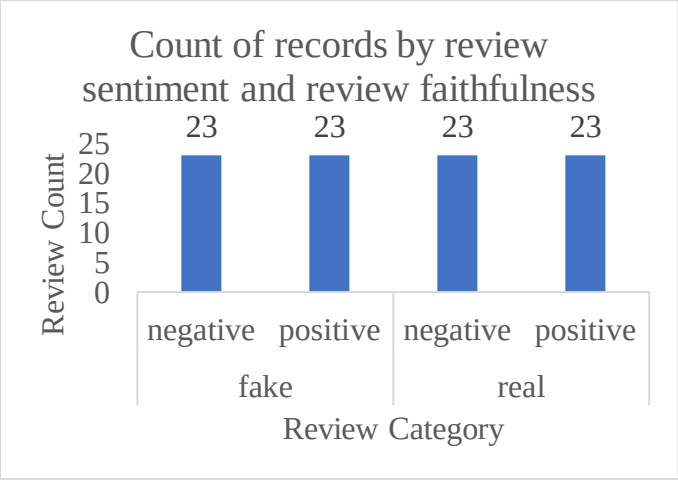
The task at hand, in this assignment, is to create ML algorithms that

1. be able to recognize true reviews from the fake ones, and,
2. be able recognize positive reviews from the negative ones.

For both tasks, we will create ML models using 2 algorithms –

3. The Naïve Bayes algorithm, and,
4. The Support Vector Machine algorithm

The data set consists of a collection of customer reviews — some true and some false, some positive and some negative. The distribution of records in the data set is shown below.



After we create each model, we will examine the model's macro accuracy, precision and recall scores. We will report on each of the model's metrics and find which model best predicts solves our purpose.

About the Naïve Bayes algorithm –

The Naïve Bayes classifiers are a family of probabilistic ML classifiers based on Bayes theorem which calculates conditional probabilities – probability that an event will occur given that

something else has already occurred. Naïve Bayes assumes that the features in the data set are independent of each other. These algorithms are usually fast, accurate and reliable.

About the Support Vector Machine algorithm –

The SVM classification algorithm is a supervised machine learning classifier for two-group classification problems. It creates a divider/hyperplane that separates two groups of labels/targets. Since, this is a two-group classifier, it creates multiple hyperplanes between multiple target pairs for a model where we have more than 2 distinct values in target.

Analysis and Data Pre-processing –

We start with cleaning the data –

1. We removed the blank lines between each consecutive record in the data set.
2. We removed two records from the data set that had reviews as ‘’, because this seems to be an error. This reduces the number of records from 92 to 90.
3. The column “lie” in our data set, consists of [‘fake’, True] values. We change this the records with True to ‘real’ so that the value options for that column become [‘real’, ‘fake’] and make more sense.

I used scikit-learn machine learning libraries to create these models in python.

1. I start with importing different packages like sklearn, pandas, numpy, etc. Some of these, like sklearn, help us with creation and implementation of the models, some, like matplotlib and graphviz help in creating visualizations and others just help in data manipulation.
2. I split the data in X = “reviews”, y = “sentiment” and z = “lie”.
3. I split the data set into training and testing in 80:20 ratio. We will use 80% of the data to train our models and 20% to test these trained models.

Evaluation Method –

For evaluating these models, we will compare their macro accuracy, precision and recall scores. Accuracy score represents the percent of correct prediction on test data when compared to the actual labels in test data. Precision refers to the percent of correct predictions relative to total predictions of a certain category. Recall refers to the actual values predicted accurately as a percent of total actual values of a certain category. We will also look at their confusion matrix whenever possible.

SECTION 2 – SENTIMENT ANALYSIS

1. We vectorize the training and test reviews using 4 commonly used vectorizers –
    a. Default count vectorizer – uses word frequency in the corpus
    b. Count vectorizer with min_df = 5 – ignores words/features that have document frequency (number of documents the term is in) lower than 5.

Shivangi Mundhra
SUID - 842548148

   c. Default TFIDF vectorizer – Term Frequency – Inverse Document Frequency. Usually considered a better alternative to count or Boolean vectorizers as it assigns importance to feature vectors and not just assigns a count frequency.
   d. TFIDF vectorizer with min_df = 5 – ignores words that have a document frequency less than 5.
2. We use count vectorizers because Naïve Bayes models use frequency base for their calculation, so in theory these models should work better with count vectorizers. We use TFIDF model because SVM creates hyperplanes between target variables and since TFIDF imparts a sense of importance to the words as well, it should in theory help create better hyperplane placement. So, theoretically SVM models should produce better results with the TFIDF vectorizers.
3. We use the word vectors to train the model on sentiment target values (positive, negative) using base MNB and SVM classifiers.
4. We measure the accuracy, precision and recall scores. The best results for base models were obtained with MNB with default TFIDF vectorizer. The confusion matrix is shown below.

		Prediction		
		negative	positive	Precision
Actual	negative	8	0	100%
	positive	3	7	70%
	Recall	73%	100%	

   a. Accuracy = 83.33%
   b. Precision = 86%
   c. Recall = 85%
5. We use GridSearchCV from sklearn module to tune parameters for all these models.
   a. To tune an MNB model, we use GridSearchCV to find best values for alpha and fit_prior. Alpha is an adaptive smoothing parameter and fit_prior decides whether to learn prior probabilities or not.
   b. To tune a SVM model, we use GridSearchCV to find best values for C, gamma and kernel. C is a regularization parameter, kernel decides whether to use a linear, polynomial or rbf kernel and gamma is the kernel coefficient.
6. The best results for tuned models were obtained with MNB with default TFIDF vectorizer. The tuned parameters are alpha = 1.3, fit_prior = TRUE. The confusion matrix is shown below. Clearly, the tuning had no effect.

		Prediction		
		negative	positive	Precision
Actual	negative	8	0	100%
	positive	3	7	70%
	Recall	73%	100%	

   a. Accuracy = 83.33%
   b. Precision = 86%
   c. Recall = 85%

Shivangi Mundhra
SUID - 842548148

7. The tuning did not enhance the performance of model. The final model has an accuracy of 83.33% which is not very high but not low as well. This is a good model.

SECTION 3 – FAKE REVIEW DETECTION

1. We vectorize the training and test reviews using 4 commonly used vectorizers –
    a. Default count vectorizer – uses word frequency in the corpus
    b. Count vectorizer with min_df = 5 – ignores words/features that have document frequency (number of documents the term is in) lower than 5.
    c. Default TFIDF vectorizer – Term Frequency – Inverse Document Frequency. Usually considered a better alternative to count or Boolean vectorizers as it assigns importance to feature vectors and not just assigns a count frequency.
    d. TFIDF vectorizer with min_df = 5 – ignores words that have a document frequency less than 5.
2. We use count vectorizers because Naïve Bayes models use frequency base for their calculation, so in theory these models should work better with count vectorizers. We use TFIDF model because SVM creates hyperplanes between target variables and since TFIDF imparts a sense of importance to the words as well, it should in theory help create better hyperplane placement. So, theoretically SVM models should produce better results with the TFIDF vectorizers.
3. We use the word vectors to train the model on "lie" target values (fake, real) using base MNB and SVM classifiers.
4. We measure the accuracy, precision and recall scores. The best results for base models were obtained with MNB with default TFIDF vectorizer and with SVM with default TFIDF vectorizer. The confusion matrix is shown below.

		Prediction		
		negative	**positive**	**Precision**
Actual	**negative**	**4**	6	40%
	positive	2	**6**	75%
	Recall	67%	50%	

    a. Accuracy = 55.56%
    b. Precision = 58%
    c. Recall = 57%
5. We use GridSearchCV from sklearn module to tune parameters for all these models.
    a. To tune an MNB model, we use GridSearchCV to find best values for alpha and fit_prior. Alpha is an adaptive smoothing parameter and fit_prior decides whether to learn prior probabilities or not.
    b. To tune a SVM model, we use GridSearchCV to find best values for C, gamma and kernel. C is a regularization parameter, kernel decides whether to use a linear, polynomial or rbf kernel and gamma is the kernel coefficient.
6. The best results for tuned models were obtained with SVM with min_df = 5 count vectorizer. The tuned parameters were C = 1, gamma = 1 and kernel = 'rbf'. The confusion

Shivangi Mundhra
SUID - 842548148

matrix is shown below. The parameter tuning increased the model performance slightly.

		Prediction		
		negative	positive	Precision
Actual	negative	7	3	70%
	positive	3	5	63%
	Recall	70%	63%	

a. Accuracy = 66.67%
b. Precision = 66%
c. Recall = 66%

7. The tuning helps in increasing the performance of model slightly but the final model has an accuracy of 67% which is not good for a prediction model.

SECTION 4 – ALGORITHM PERFORMANCE COMPARISON

Target	Model	Vectorization Type	Vectorization Settings	Parameter Settings	Accuracy	Precision	Recall
Sentiment	MNB	Count Vectorization	Default	Default	72.22%	72%	72%
Sentiment	MNB	Count Vectorization	min_df = 5	Default	83.33%	83%	84%
Sentiment	MNB	TFIDF Vectorization	Default	Default	83.33%	86%	85%
Sentiment	MNB	TFIDF Vectorization	min_df = 5	Default	83.33%	83%	84%
Sentiment	MNB	Count Vectorization	Default	alpha = 1.3, fit_prior = TRUE	72.22%	72%	72%
Sentiment	MNB	Count Vectorization	min_df = 5	alpha = 1.3, fit_prior = TRUE	83.33%	83%	84%
Sentiment	MNB	TFIDF Vectorization	Default	alpha = 1.3, fit_prior = TRUE	83.33%	86%	85%
Sentiment	MNB	TFIDF Vectorization	min_df = 5	alpha = 1.3, fit_prior = TRUE	83.33%	83%	84%
Sentiment	SVM	Count Vectorization	Default	Default	66.66%	67%	65%
Sentiment	SVM	Count Vectorization	min_df = 5	Default	66.67%	68%	68%
Sentiment	SVM	TFIDF Vectorization	Default	Default	72.22%	81%	75%
Sentiment	SVM	TFIDF Vectorization	min_df = 5	Default	77.78%	79%	79%
Sentiment	SVM	Count Vectorization	Default	C = 0.1, gamma = 0.0001, kernel = 'linear'	72.22%	72%	72%
Sentiment	SVM	Count Vectorization	min_df = 5	C = 10, gamma = 0.001, kernel = 'rbf'	72.22%	72%	72%
Sentiment	SVM	TFIDF Vectorization	Default	C = 1, gamma = 0.0001, kernel = 'linear'	77.78%	79%	79%
Sentiment	SVM	TFIDF Vectorization	min_df = 5	C = 1, gamma = 1, kernel = 'rbf'	77.78%	79%	79%
Sentiment	Linear SVM	TFIDF Vectorization	Default	Default	77.78%	79%	79%
Lie Detection	MNB	Count Vectorization	Default	Default	55.56%	56%	56%
Lie Detection	MNB	Count Vectorization	min_df = 5	Default	55.56%	58%	57%
Lie Detection	MNB	TFIDF Vectorization	Default	Default	55.56%	56%	56%
Lie Detection	MNB	TFIDF Vectorization	min_df = 5	Default	55.56%	56%	56%
Lie Detection	MNB	Count Vectorization	Default	alpha = 1.1, fit_prior = TRUE	55.56%	56%	56%
Lie Detection	MNB	Count Vectorization	min_df = 5	alpha = 1.3, fit_prior = TRUE	55.56%	58%	57%
Lie Detection	MNB	TFIDF Vectorization	Default	alpha = 0.5, fit_prior = TRUE	55.56%	56%	56%
Lie Detection	MNB	TFIDF Vectorization	min_df = 5	alpha = 0.5, fit_prior = TRUE	55.56%	56%	56%
Lie Detection	SVM	Count Vectorization	Default	Default	44.44%	46%	47%
Lie Detection	SVM	Count Vectorization	min_df = 5	Default	38.89%	39%	41%
Lie Detection	SVM	TFIDF Vectorization	Default	Default	55.55%	58%	57%
Lie Detection	SVM	TFIDF Vectorization	min_df = 5	Default	44.44%	45%	45%
Lie Detection	SVM	Count Vectorization	Default	C = 1, gamma = 0.1, kernel = 'rbf'	55.56%	58%	57%
Lie Detection	SVM	Count Vectorization	min_df = 5	C = 1, gamma = 1, kernel = 'rbf'	66.67%	66%	66%
Lie Detection	SVM	TFIDF Vectorization	Default	C = 1, gamma = 1, kernel = 'rbf'	55.56%	58%	57%
Lie Detection	SVM	TFIDF Vectorization	min_df = 5	C = 0.0001, gamma = 0.001, kernel = 'poly'	38.89%	39%	39%

The above table shows the accuracies, precisions and recalls of each model that we have tested in this homework for both, lie detection and sentiment prediction using all different settings and vectorization options. We have obtained a reasonably good model for sentiment prediction but none of the models above are good enough for any practical lie detection. The main cause that

stands out from this exercise is that words can theoretically be associated with positive or negative sentiment; however, it is illogical to associate words with truth or lie.