Homework 2

Francesco Ilario 1469228
Marzio Monticelli 1459333

1 First Part: Ham/Spam

1.1 Dataset

	Training Set	**Test Set**
Spam	122	52
Ham	120	53

Table 1: Data Statistics

1.2 Pipeline Stack

1. **TfIdfVectorizer**: vect.
2. **TruncatedSVD**: dec. This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). Contrary to PCA, this estimator does not center the data before computing the singular value decomposition. This means it can work with scipy.sparse matrices efficiently. In particular, truncated SVD works on term count/tf-idf matrices as returned by the vectorizers in sklearn.feature_extraction.text. In that context, it is known as latent semantic analysis (LSA).
3. **K-Nearest Neighbors**: nbc.

1.3 Parameters to Tune

- **vect___tokenizer**: [None, stemming_tokenizer]
- **vect___ngram_range**: $[(x, y) : x \in [1, 2] \text{ and } y \in [1, 9], \text{ where } x < y]$
- **vect___analyzer**: ['word', 'char']
- **vect___max_df**: np.arange(.6, 1, .1)

- **vect___min_df**: np.arange(0., .3, .1)
- **vect___lowercase**: [True, False]
- **vect___binary**: [True, False]
- **vect___sublinear_tf**: [True, False]
- **vect___stop_words**: [None, stopwords.words("english")]
- **dec___n_components**: $x \in [1, 10]$
- **nbc___n_neighbors**: $x \in [1, 10]$
- **nbc___weights**: ['uniform', 'distance']

1.4 Train-Validation

To perform the train-validation process using more than on CPU Core, we have set the parameter "n_jobs" of the GridSearchCV to the number of core available in the machine. We did not tune this value for the Classifier that have been optimized because of the following warning: "UserWarning: Multiprocessing-backed parallel loops cannot be nested, setting n_jobs=1".

1.5 Best Configuration of Parameters

```
'dec__n_components': 6,
'nbc__n_neighbors': 3
'nbc__weights': 'distance',
'vect__analyzer': 'char',
'vect__binary': True,
'vect__lowercase': True,
'vect__max_df': 0.79999999999999993,
'vect__min_df': 0.0,
'vect__ngram_range': (1, 7),
'vect__stop_words': <nltk english stopwords>,
'vect__sublinear_tf': True,
'vect__tokenizer': <function stemming_tokenizer>
```

1.6 metrics.classification_report

	Precision	Recall	F1-Score	Support
Positive	0.95	1.00	0.97	52
Negative	1.00	0.94	0.97	53
avg/total	0.97	0.97	0.97	105

Table 2: Data Statistics

1.7 Confusion-Matrix

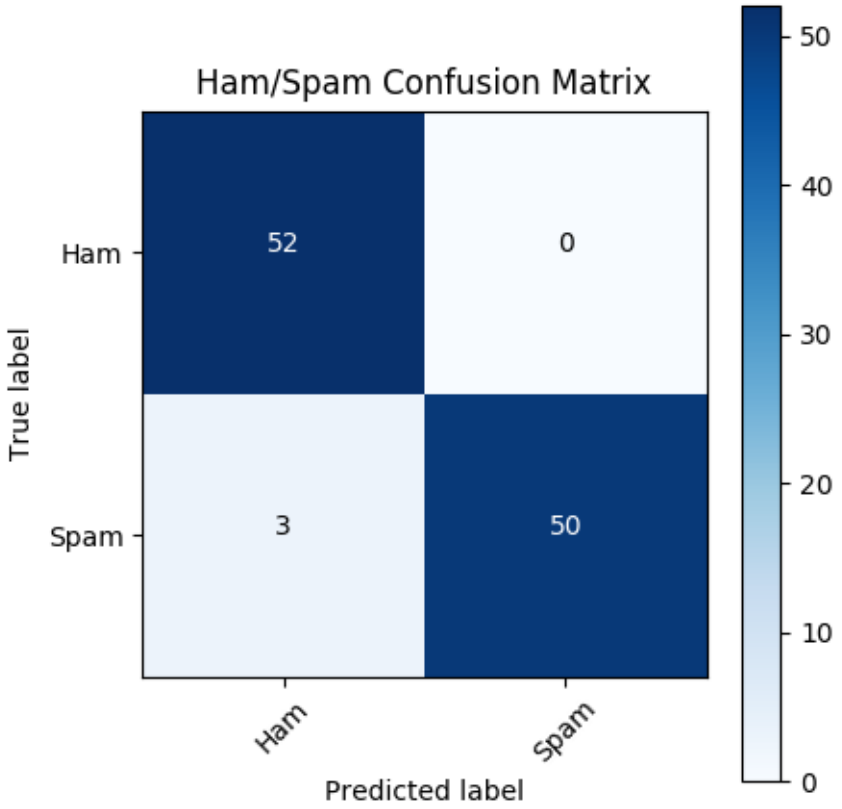

Figure 1: Confusion Matrix

1.7.1 Scores

- **Accuracy Score Normalized**: 0.971428571429
- **Matthews Correlation Coefficient**: 0.944424825187

2 Second Part: Positive/Negative

2.1 Dataset

	Training Set	Test Set
Positive	249	308
Negative	308	250
TOTAL	557	558

Table 3: Data Statistics

2.2 K-Nearest Neighbours

2.2.1 Pipeline Stack

1. **TfIdfVectorizer**: vect
2. **TruncatedSVD**: dec
3. **K-Nearest Neighbors**: nbc

2.2.2 Parameters to Tune

- **dec**: TruncatedSVD()
- **vect___tokenizer**: [None, stemming_tokenizer]
- **vect___ngram_range**: [(1, 2), (1, 3)]
- **vect___analyzer**: ['word', 'char']
- **vect___max_df**: np.arange(.8, 1, .1)
- **vect___min_df**: np.arange(0., .2, .1)
- **vect___binary**: [True, False]
- **vect___lowercase**: [True, False]
- **vect___sublinear_tf**: [True, False]
- **vect___stop_words**: [None, stopwords.words("english")]
- **dev___n_components**: $x \in [10, 15)$ with steps of 2
- **nbc___n_neighbors**: $x \in [3, 6)$ with steps of 1
- **nbc___weights**: ['distance', 'uniform']

2.2.3 Best Configuration of Parameters

```
'vect__analyzer': 'word',
'vect__binary': True,
'vect__lowercase': True,
'vect__max_df': 0.80000000000000004,
'vect__min_df': 0.0,
'vect__ngram_range': (1, 2),
'vect__stop_words': <nltk stopword for English>,
'vect__sublinear_tf': True,
'vect__tokenizer': <function stemming_tokenizer>
'dec__n_components': 14,
'nbc__n_neighbors': 5,
'nbc__weights': 'distance'
```

2.2.4 metrics.classification_report

	Precision	**Recall**	**F1-Score**	**Support**
Positive	0.93	0.93	0.93	308
Negative	0.92	0.91	0.91	250
avg/total	0.92	0.92	0.92	558

Table 4: Data Statistics

2.2.5 Confusion-Matrix

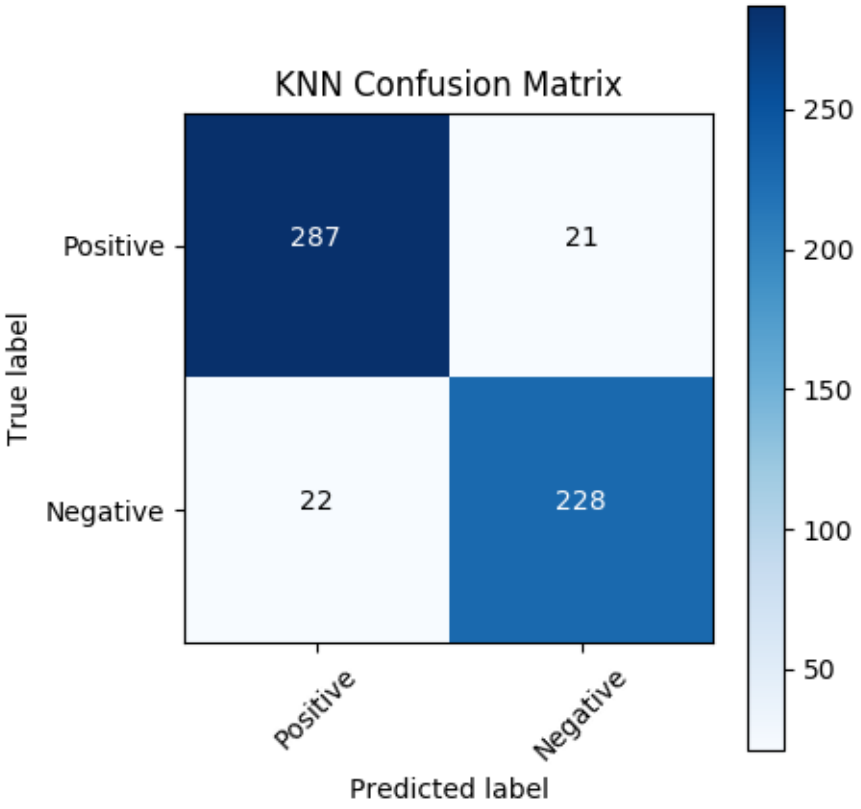

Figure 2: KNN Confusion Matrix

2.2.6 Scores

- **Accuracy Score Normalized**: 0.9229390681
- **Matthews Correlation Coefficient**: 0.84414164871

2.3 Decision Tree

2.3.1 Introduction

This method uses the input data to build a classical Decision Tree. A Decision Tree is a tree where each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf. A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning.

2.3.2 Pipeline Stack

1. **TfIdfVectorizer**: vect
2. **Decision Tree**: nbc

2.3.3 Parameters to Tune

- **vect___tokenizer**: [None, stemming_tokenizer]
- **vect___ngram_range**: [(1, 2), (1, 3)]
- **vect___analyzer**: ['word', 'char']
- **vect___max_df**: np.arange(.8, 1, .1)
- **vect___min_df**: np.arange(0., .2, .1)
- **vect___binary**: [True, False]
- **vect___lowercase**: [True, False]
- **vect___sublinear_tf**: [True, False]
- **vect___stop_words**: [None, stopwords.words("english")]

2.3.4 Best Configuration of Parameters

```
'vect__analyzer': 'word',
'vect__binary': False,
'vect__lowercase': True,
'vect__max_df': 0.80000000000000004,
'vect__min_df': 0.0,
'vect__ngram_range': (1, 2),
'vect__stop_words': <nltk stopword for English>,
'vect__sublinear_tf': False,
'vect__tokenizer': <function stemming_tokenizer>}
```

2.3.5 metrics.classification_report

	Precision	Recall	F1-Score	Support
Positive	0.94	0.99	0.96	308
Negative	0.99	0.92	0.95	250
avg/total	0.96	0.96	0.96	558

Table 5: Data Statistics

2.3.6 Confusion-Matrix

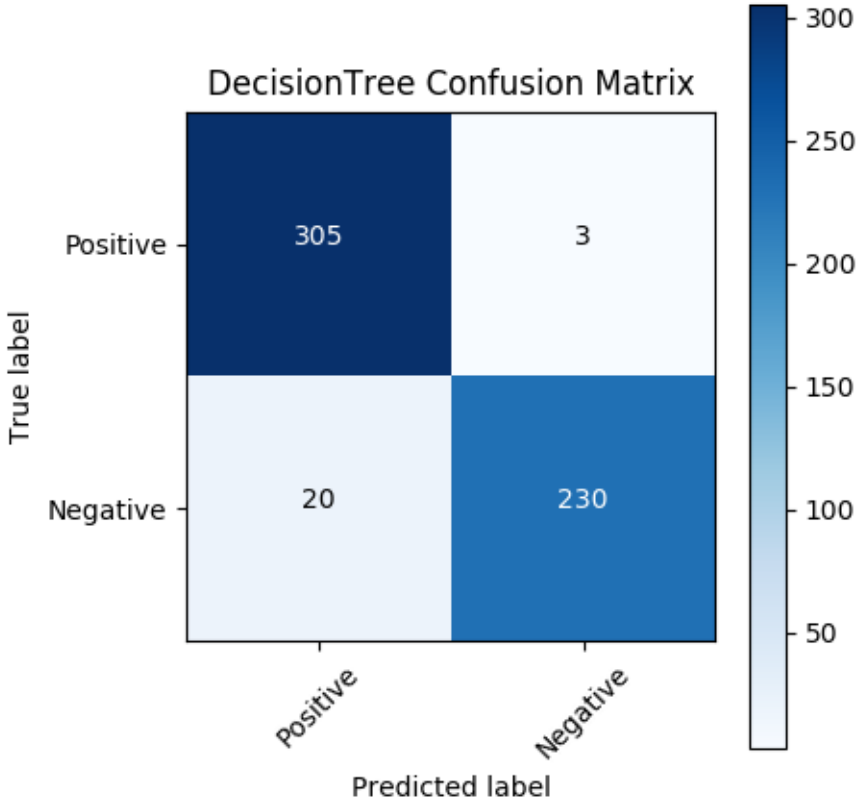

Figure 3: DecisionTree Confusion Matrix

2.3.7 Scores

- **Accuracy Score Normalized**: 0.958781362007
- **Matthews Correlation Coefficient**: 0.917890883358

2.4 SGDClassifier

2.4.1 Introduction

It is a Linear classifier that implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate).

2.4.2 Pipeline Stack

1. **TfIdfVectorizer**: vect
2. **SGD**: nbc

2.4.3 Best Configuration of Parameters

```
'vect__analyzer': 'word',
'vect__binary': False,
'vect__lowercase': False,
'vect__max_df': 0.90000000000000002,
'vect__min_df': 0.0,
'vect__ngram_range': (1, 2),
'vect__stop_words': <nltk stopword for English>,
'vect__sublinear_tf': True,
'vect__tokenizer': <function stemming_tokenizer>}
```

2.4.4 metrics.classification_report

	Precision	**Recall**	**F1-Score**	**Support**
Positive	0.95	0.98	0.96	308
Negative	0.97	0.94	0.96	250
avg/total	0.96	0.96	0.96	558

Table 6: Data Statistics

2.4.5 Confusion-Matrix

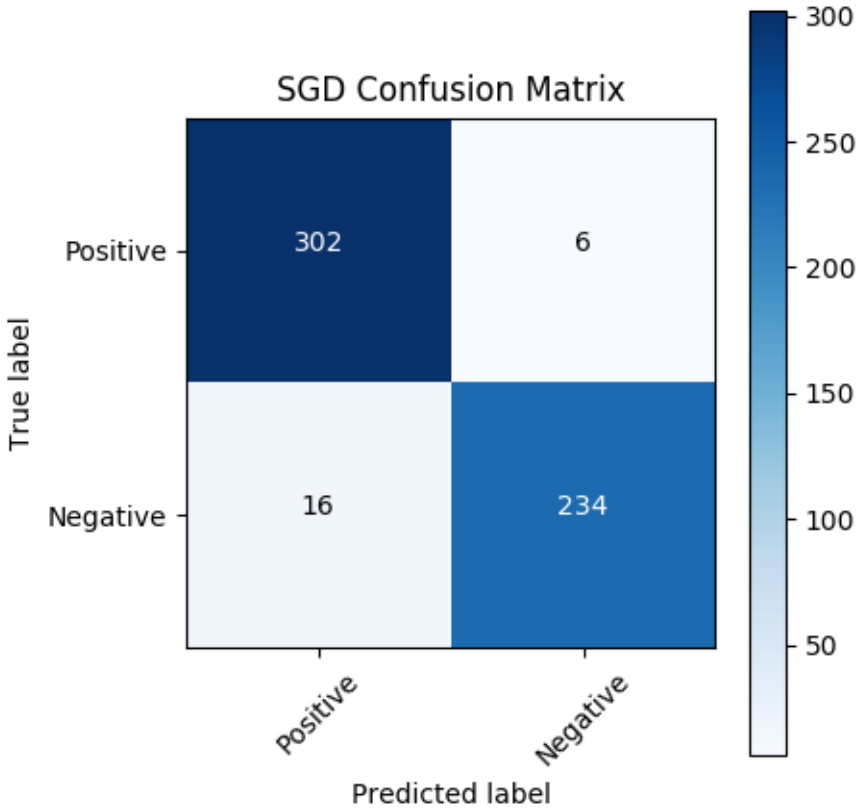

Figure 4: Confusion Matrix

2.4.6 Scores

- **Accuracy Score Normalized**: 0.960573476703
- **Matthews Correlation Coefficient**: 0.920593453055