Russian Disinformation: Naive Bayes & Support Vector Machine Results

Presentation by: Tobby Lie

Github Link: https://github.com/tobby-lie/Russion-Disinformation-Project

Objective & Approach

- The initial approach taken was to utilize two supervised machine learning classifiers (Naive Bayes and Support Vector Machines) in order to determine if the data based on the **'description'** field included **'anger'**, **'fear'** or **both** of these within the **tag** field.
- The models utilized:
    - **Naive Bayes:** A simple probabilistic classifier based on Bayes' theorem (probability of an event, based on prior knowledge of conditions that might be related to the event) with strong independence assumptions between the features.[1]
    - **Support Vector Machines:** Supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other. It is a non-probabilistic binary linear classifier.[2]

1. "Naive Bayes Classifier." *Wikipedia*, Wikimedia Foundation, 27 Nov. 2019, https://en.wikipedia.org/wiki/Naive_Bayes_classifier.
2. "Support-Vector Machine." *Wikipedia*, Wikimedia Foundation, 1 Dec. 2019, https://en.wikipedia.org/wiki/Support-vector_machine.

Data-preprocessing

- From an initial data frame containing the original contents of the given .csv file, we generated three new data frames. One for anger, fear and both classifications. Each data frame had a new column called ‘label numerical’ and ‘label word’ containing 0 or 1 and ‘none’ or ‘anger’ respectively depending on if the tags came up positive for that classification. This is how we created our training data.
- The data was split into train and test sets in order to validate our results.
- Our collection of text documents from the ‘description’ field were converted into a matrix of word frequencies for each word for each document. This is what we call a bag of words (BoW). An example of this will be included in the next slide

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
>>> vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2))
>>> X2 = vectorizer2.fit_transform(corpus)
>>> print(vectorizer2.get_feature_names())
['and this', 'document is', 'first document', 'is the', 'is this',
'second document', 'the first', 'the second', 'the third', 'third one',
 'this document', 'this is', 'this the']
 >>> print(X2.toarray())
 [[0 0 1 1 0 0 1 0 0 0 0 1 0]
 [0 1 0 1 0 1 0 1 0 0 1 0 0]
 [1 0 0 1 0 0 0 0 1 1 0 1 0]
 [0 0 1 0 1 0 1 0 0 0 0 0 1]]
```

Hide prompts and outputs

Image from scikit-learn documentation for CountVectorizer which implements a bag of words

Dealing with Imbalanced Data

- After training with our original data we found that we were facing class imbalance. For anger, fear and both classification we had **781**, **317** and **186** positive samples respectively for the total dataset size of **3012** so our negative sample set in all cases were much larger.
- We implemented undersampling and oversampling in an effort to balance our dataset.
    - **Undersampling:** We reduced the number of negative samples from our dataset so that this number matched the number of samples from our positive samples.
    - **Oversampling:** We utilized random over sampling in order to over-sample our minority class (positive) by picking samples at random with replacement.

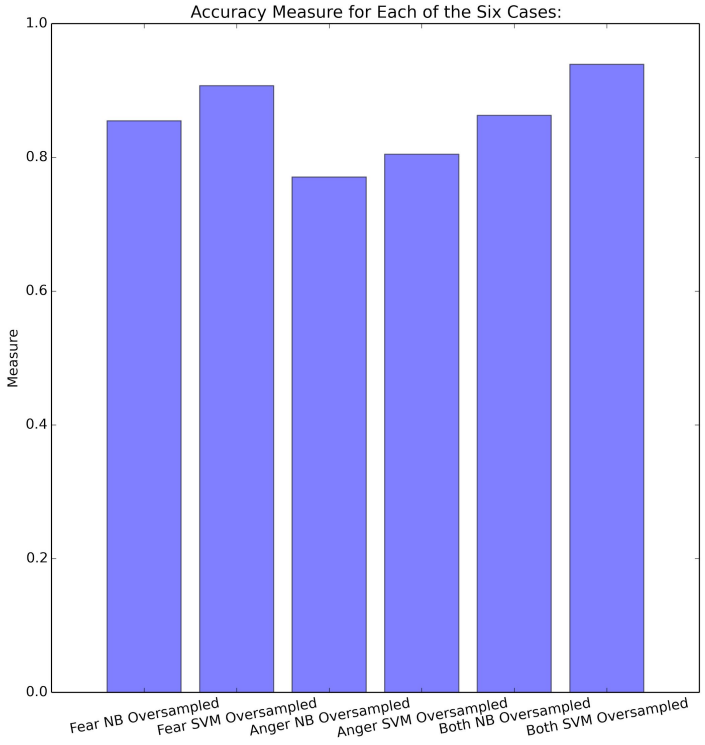
Metrics

- Before we take a look at the results let's take a look at some **metrics** and what they represent:
- **TP, FP, TN, FN** = True Positives, False Positives, True Negatives, False Negatives
    - **Accuracy:** (TP + TN)/(TP + TN + FP + FN) -- Percentage of correctly classified instances.
    - **Precision:** TP / TP + FP -- Accuracy of positive predictions.
    - **Recall:** TP / TP + FN -- Fraction of positives that were correctly identified.
    - **F1 Score:** 2 x (precision x recall) / (precision + recall) -- Takes into consideration precision and recall. Finds the harmonic mean of precision and recall.
    - **Sensitivity:** TP / TP + FN -- Measures proportion of actual positives that are correctly identified as such.
    - **Specitivity:** TN / TN + FP -- Measures the proportion of actual negatives that are correctly identified as such.
    - **Classification Report:** Report that includes Precision, Recall and F1-Score.
    - **Confusion Matrix:** Allows you to look at matrix of correctly and incorrectly classified predictions.

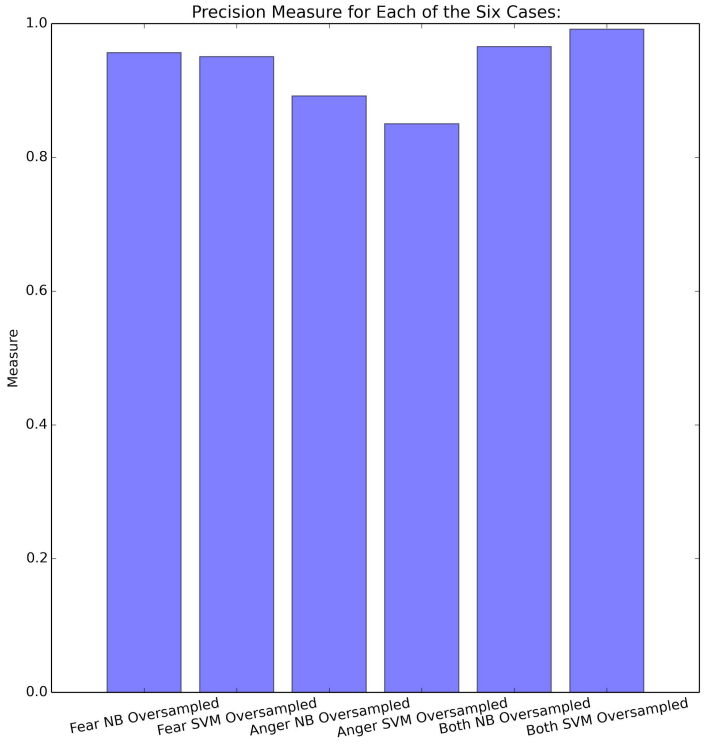
The Six Cases Considered

1. Fear classification with NB
2. Fear classification with SVM
3. Anger classification with NB
4. Anger classification with SVM
5. Fear & Anger (Both) classification with NB
6. Fear & Anger (Both) classification with SVM

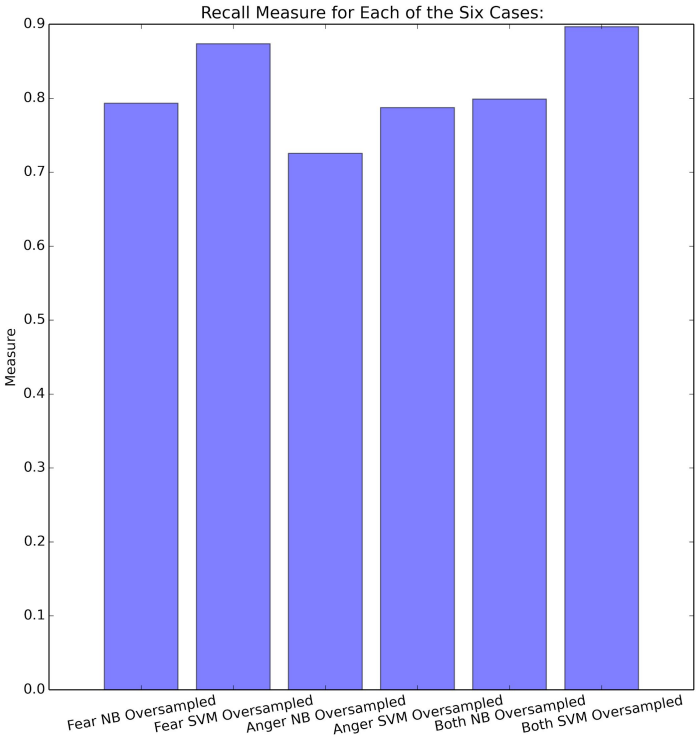
The next couple of slides will include bar charts showing each of our **measures of performance (metrics)**. Each bar will represent the **six cases** we've utilized against our **oversampled** dataset mentioned above.



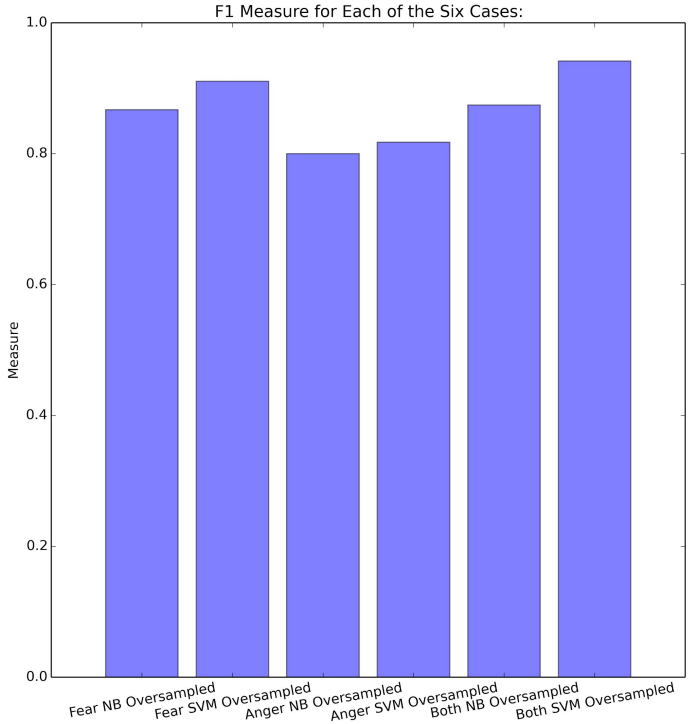
Accuracy Measures for Each of Our 6 Cases



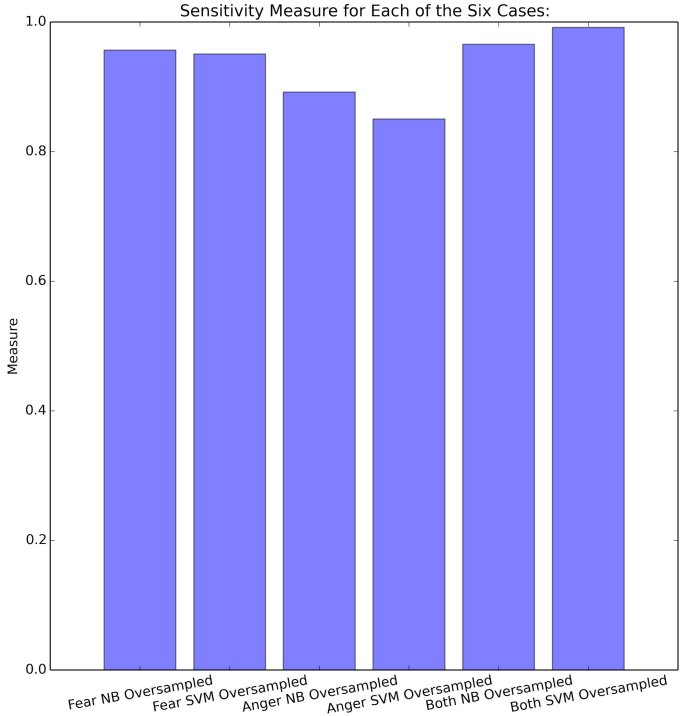
Precision Measures for Each of Our 6 Cases



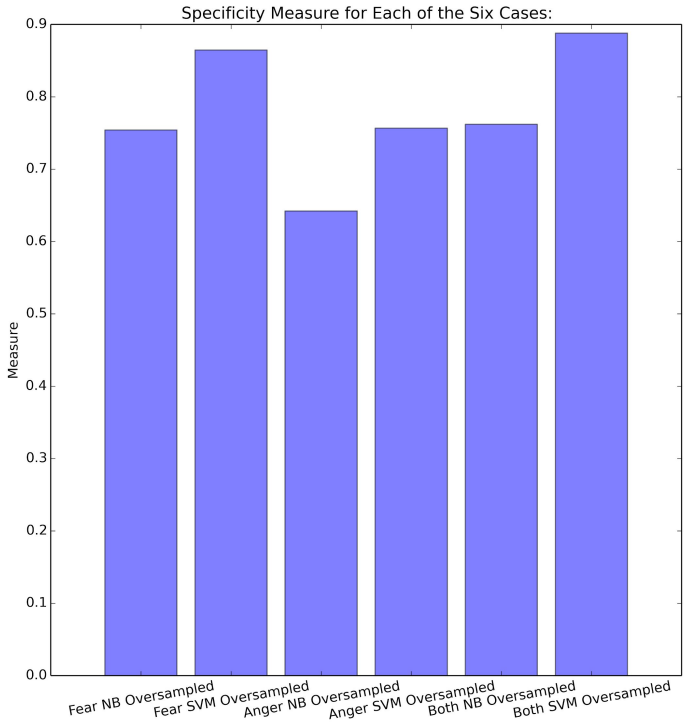
Recall Measures for Each of Our 6 Cases



F1 Measures for Each of Our 6 Cases



Sensitivity Measures for Each of Our 6 Cases



Specificity Measures for Each of Our 6 Cases

Links to More Work & Results

- Links to Jupyter Notebook files for further NB & SVM code and results:
  - Naive Bayes: https://github.com/tobby-lie/Russion-Disinformation-Project/blob/master/naive_bayes.ipynb
  - Support Vector Machine: https://github.com/tobby-lie/Russion-Disinformation-Project/blob/master/support_vector_machine.ipynb