Text Classification - Distinguish facts from fake facts about animals

Sagar Nandeshwar - Student ID: 260920948

Objective:
In this project, I investigated whether linear classifiers could distinguish real facts from fake facts about animals and compare three linear classifiers, to know whether the choice of linear classifier matters or not.

Dataset:
I collected 100 real facts and 100 fake facts about each of the four animals - cats, dogs, owls and penguins - using ChatGPT 4.0. I save them in facts.txt and fakes.txt file.

Experimental Procedure:
Step 1: Load the data from facts.txt and fakes.txt file.
Step 2: Split and shuffle the dataset into Train and Test set.
Step 3: Pre-process the dataset by first tokenizing them. Then, performing combinations of lemmatization, stemming, removing stop words and removing punctuation. Lastly, detokenize the dataset back to continues sentence.
Step 4: Convert the dataset content into numerical feature vectors using CountVectorizer and TfidfTransformer.
Step 5: Used GridSearchCV to tune hyperparameters for all the three models
Step 6: Train the three linear classifiers.
step 7: Predict the output of test dataset and compare the accuracy and F1 score of three classifiers.

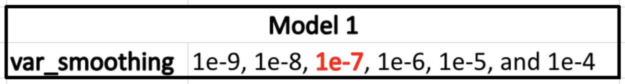
Pre-processing:

- **Tokenization**: Splits a sentence into individual words (token). For example, “I like cat” to “I” “like” and “cat”. I used NLTK word_tokenize to tokenize the data.
- **Detokenization**: Converting a list of tokens back into continuous text string. For example, “I” “like” and “cat” to “I like cat”. I used NLTK TreebankWordDetokenizer to detokenize the data.
- **Remove Stop words:** Stop words refer to the most common words in any natural language. They do not add much significant meaning to a sentence. For example, "and", "the", "is", "of", "to" and "in", etc. I used NLTK stopwords to remove all stop words.
- **Remove Punctuation:** I used string.punctuation condition to remove punctuation.
- **Lemmatization:** Reduces words to their base or dictionary form, known as a lemma. For example, the lemma of "am", "are", and "is" is "be". I used NLTK WordNetLemmatizer function to pre-process lemmatization.
- **Stemming:** Remove morphological affixes from words, leaving only the word stem. For example, “jumping” to “jump”. I used NLTK PorterStemmer function to pre-process stemming.
- **Feature Vectors**: Convert a collection of text documents to a matrix of token counts using sklearn’s CountVectorizer
- **Normalization**: Transform a count matrix to a normalized tf or tf-idf representation using sklearn’s TfidfTransformer. Tf means term-frequency while tf-idf means term-frequency times inverse document-frequency.

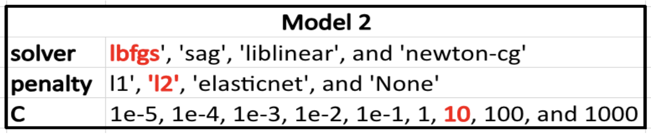
Linear Classifiers:

- **Naive Bayes (Model 1):** Naive Bayes is a probabilistic machine learning algorithm that's based on the Bayes' theorem.
- **Logistic Regression (Model 2)** logistic regression finds the best fitting model to describe the relationship between the dichotomous characteristic of interest and a set of independent variables.
- **Support Vector Machines SVM (Model 3):** The primary goal of SVM is to find a hyperplane (or decision boundary) that best divides a dataset into classes.

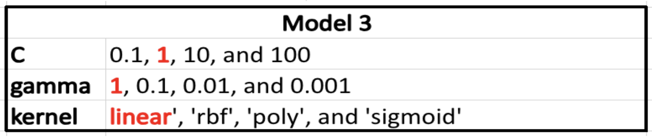
Parameter Settings:
I tested the model’s performance with a different combination of pre-processing techniques. I also tested the model’s performance with different sizes of training and test dataset. I also tested the following hyperparameter settings for all three models and compared the performance against the default model setting. I highlighted the best parameter settings for each model in red.



Model 1	
var_smoothing	1e-9, 1e-8, 1e-7, 1e-6, 1e-5, and 1e-4

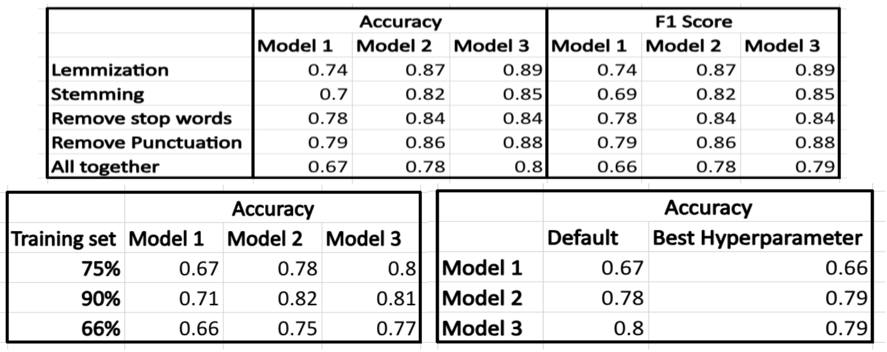


Model 2	
solver	lbfgs', 'sag', 'liblinear', and 'newton-cg'
penalty	l1', 'l2', 'elasticnet', and 'None'
C	1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, and 1000



Model 3	
C	0.1, 1, 10, and 100
gamma	1, 0.1, 0.01, and 0.001
kernel	linear', 'rbf', 'poly', and 'sigmoid'

Result:
I used sklearn accuracy_score, which is fraction of correctly classified samples, and F1 score, which is F1 = 2 * (precision * recall) / (precision + recall), to compare model’s performance.



	Accuracy			F1 Score		
	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
Lemmization	0.74	0.87	0.89	0.74	0.87	0.89
Stemming	0.7	0.82	0.85	0.69	0.82	0.85
Remove stop words	0.78	0.84	0.84	0.78	0.84	0.84
Remove Punctuation	0.79	0.86	0.88	0.79	0.86	0.88
All together	0.67	0.78	0.8	0.66	0.78	0.79

	Accuracy		
Training set	Model 1	Model 2	Model 3
75%	0.67	0.78	0.8
90%	0.71	0.82	0.81
66%	0.66	0.75	0.77

	Accuracy	
	Default	Best Hyperparameter
Model 1	0.67	0.66
Model 2	0.78	0.79
Model 3	0.8	0.79

Conclusion:
Since Model 2 and Model 3 were able to get an accuracy of higher than 0.85 with selective pre-processing technique, we can say that linear classifier can distinguish real facts from fake facts about the animals. Additionally, since Model 2 and Model 3 perform significantly better than model 1 we also say that the choice of linear classifier does matter.

Removing stop words and punctuation impacted much more than lemmization and stemming, however putting them all together reduces the performance of the three models. Also, increasing the training data does improve the performance for model 1 and 2, but it may lead to overfitting, while model 3 does not have much impact of training dataset size. Lastly, I did not see any significant difference in changing the model's hypermeter from their default setting.

Limitation:
First, we made a big assumption about decision boundary that the text is either fact or fake, but there may be situations where due to certain exceptions some facts might not be true. We also assume a linear relationship between the features and the output, however, for many real-world problems, the relationship between the input features and the output can be nonlinear. Lastly, Linear classifiers assume that features are independent. This is rarely the case in text data, where the context can drastically change the meaning; they don't inherently understand the semantic relationship between words.