Problem Set - 4

Please read all of the guidelines carefully before submitting the problem set. (Unless specified) each question is ***20 points*** and there are ***100 points*** in total.

Due date: Sunday, February 20, 11:59 PM. Late submissions will be accepted with a penalty! (10% reduction per day – no submissions accepted two days after the deadline.)

Guidelines – Before You Start

1) **You should complete the problem set on your own.** Discussing ideas is fine; but, sharing answers and sharing code will be considered as plagiarism.
2) You will be using the **`Python`** programming language. You need to write your codes in an empty **.ipynb** file.
3) Make sure that you provide many comments to describe your code and the variables that you created.
4) Please use **LaTeX** or **MS Word** to submit your written responses (hand-written responses will not be graded).
5) For some of the coding exercises, you may need to do a little bit of "**Googling**" or review the documentation.

Deliverables:

1) The code of the problem set in **.ipynb** format (one file)
2) Short answers written with ***LaTeX*** or **MS Word** and exported in **.pdf** format (one file)

Questions

For (most of) the questions below, please use the fake news dataset uploaded on *BlackBoard* (called '`corona_fake.csv`'). You can find the file under *'Data'* tab.

Please include your code also in your .pdf file (in code blocks).

Data Pre-Processing (40 points)

1) **[20 points]** Using the `pandas` package for `Python`, import the **corona_fake.csv** dataset, and do the following:
   a) **[5 points]** Import the `nltk` package. Check the documentation: https://www.nltk.org/
   b) **[15 points]** Take a look at the ***text*** column in the dataset, and do the following:
      i. **[3 points]** Using `nltk.word_tokenize()`, **tokenize the text.**
      ii. **[3 points]** Using the POS-tagging feature (`nltk.pos_tag`), POS-tag the tokenized words.
      iii. **[3 points]** Using `WordNetLemmatizer (from nltk.stem import WordNetLemmatizer)` lemmatize the pos-tagged words you obtained above. (*Hint*: If there is no available tag, append the token as is; else, use the tag to lemmatize the token)

iv. **[3 points]** Using the list of stop words that can be imported (`nltk.corpus import stopwords`), remove the stopwords in lemmatized text [Note: the language needs to be set as 'english'.].

v. **[3 points]** Finally, also **remove numbers, words that are shorter than 2 characters, punctuation, links and emojis**. Finally, convert the obtained list of tokenized+tagged+lemmatized+cleaned list of words back into a joined string (joined by space ' ') and add the result as ***text_clean*** column to your dataset.

2) **[20 points]** Let's vectorize the data we produced above by using two approaches: Bag of Words (BOW) and TF-IDF; and, at the end, we will make a prediction:

a. **[5 points]** Read the following page: https://en.wikipedia.org/wiki/N-gram. Explain what an 'n-gram' is and why it is helpful in max. 200 words.

b. **[5 points]** Import `CountVectorizer` and `TfidfVectorizer`:
`from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer`

c. **[5 points]** Using `CountVectorizer`, create three vectorized representations of ***text_clean*** [set **`lowercase=True`**]:

i. One vectorized representation where `ngram_range = (1,1)`
ii. One vectorized representation where `ngram_range = (1,2)`
iii. One vectorized representation where `ngram_range = (1,3)`

d. **[5 points]** Using `TfidfVectorizer`, create three vectorized representations of ***text_clean*** [set **`lowercase=True`**]:

i. One vectorized representation where `ngram_range = (1,1)`
ii. One vectorized representation where `ngram_range = (1,2)`
iii. One vectorized representation where `ngram_range = (1,3)`

Prediction (20 points)

3) **[20 points]** Now, let's use `sklearn.linear_model.LogisticRegressionCV` to do some predictions. Set `cv = 5`, `random_state = 265`, and `max_iter = 1000`, and `n_jobs = -1` (other parameters should be left as default) [Note: training size is 70%, test size is 30%, split by `random_state = 265`].

a. **[10 points]** By using the **three (3)** different versions of the `CountVectorizer` dataset you created above, run logistic regression to predict class labels (fake, true). Report **three (3)** accuracy values associated with each of the regressions.

b. **[10 points]** By using the **three (3)** different versions of the `TfidfVectorizer` dataset you created above, run logistic regression to predict class labels (fake, true). Report **three (3)** accuracy values associated with each of the regressions.

c. Combine and report all accuracy values in a table (6 values in total).

Theoretical question (40 points)

4) **[40 points]** Check the optimizer (solver) functions used by `sklearn.linear_model.LogisticRegressionCV`. For each function, explain in around 100 words what they mean; specifically:
   a. **[8 points]** What does ***newton-cg*** mean?
   b. **[8 points]** What does ***lbfgs*** mean?
   c. **[8 points]** What does ***liblinear*** mean?
   d. **[8 points]** What does ***sag*** mean?
   e. **[8 points]** What does ***saga*** mean?

   Note: For this question you might need to do some online research. It is your job to find out how they work. You are also welcome to use formulas / matrices in your description.