Project Description

Train a `LinearSVC()` classifier and `SVC()` classifiers with, respectively, the Linear ('linear') and RBF ('rbf') kernels provided by Sci-kit Learn (package sklearn) to predict `'Priority'` labels for the Java Development Tools Bug dataset. (Note that `LinearSVC()` and `SVC()` with a linear kernel are not the same thing due to different internal algorithmic settings.) Your project consists of the following tasks:

1. Clean and remove short bug reports as in `TextClassification.ipynb`. Organize the remaining data to a data frame called `df`.
2. Generate a balanced data frame called `df_balanced` from `df` by restricting the number of entries in the `P3` category to 5000.
3. Lemmatize words and use only '`nav`' lemmas for classifications, where '`nav`' stands for nouns, pronouns, adjectives, adverbs, and verbs as in FeatureExtraction.
4. Use grid search with 5-fold cross validation on `LinearSVC()` to determine the best hyperparameters on the following grid with a pipeline on `tfidf` and `LinearSVC`:

```
grid_param = [{
    'tfidf__min_df': [5, 10],
    'tfidf__ngram_range': [(1, 3), (1, 6)],
    'LinearSVC__penalty': ['l2'],
    'LinearSVC__loss': ['hinge'],
    'LinearSVC__max_iter': [10000]
}, {
    'tfidf__min_df': [5, 10],
    'tfidf__ngram_range': [(1, 3), (1, 6)],
    'LinearSVC__C': [1, 10],
    'LinearSVC__tol': [1e-2, 1e-3]
}]
```

   where "`tfidf`" represents the method to generate tfidf vectorization on '`nav`' lemmas for each bug report in `df_balanced`, and "`LinearSVC`" represents `LinearSVC()`.

5. Use grid search with 5-fold cross validation on `SVC()` to determine the best hyperparameters on the following grid with a pipeline on `tfidf` and `SVC`:

```
tuned_parameters = [{
'tfidf__min_df': [5, 10],
    'tfidf__ngram_range': [(1, 3), (1, 6)],
    "SVC__C": [1,10,100,1000],
    "SVC__kernel": ['rbf', 'linear'],
    "SVC__gamma": [1e-3, 1e-4],
    "SVC__tol": [1e-2, 1e-10]
}]
```

6. Use an 80-20 train-test split on `df_balanced` as your training data and test data. Train `LinearSVC()` and `SVC()` on the hyperparameters you obtained above at Step 4 and Step 5, respectively, over the training data.
7. Use the trained `LinearSVC()` as the baseline, compare your `SVC()` classifiers with the baseline on accuracy, precision, recall, and F1 scores over the test data and determine which classifier is the best.
8. Use the best hyperparameters obtained at Step 5 except the kernel types to train `SVC(kernel='linear')` and `SVC(kernel='rbf')` over the training data and compare their F1 scores over the test data. Determine if the best `SVC()` trained at Step 6 is the same as the one you obtained here.
9. For each classifier, find out the most important words using LIME and ELI5 methods and visualize the outcomes. Draw up your explanations.

You may borrow code from `TextClassification.ipynb` and `Prob_LIME_ELI5.ipynb`.

Note:

- As always, do not include irrelevant code in your submission. Irrelevant code is code that is not needed for executing your tasks and so is junk code. Be clean and concise in your submission.
- It may take hours to train SVC classifiers using grid search to obtain the best hyperparameters on a laptop computer. Also, keep your sample size moderate and the number of features small when performing LIME analysis. Budget your time wisely.
- In addition to submitting your code and report in one notebook file that is, enter code in code cell, and text and figures in text cells, please also submit a video presentation not to exceed 10 minutes of your project. Your submission should be organized according to the task list. Name your notebook file and video file in the following format: <your last name>-Project.
- Students taking COMP.5500 are required to give an oral presentation in class.