Argument Quality Assessment

Feature selection:

- **TF-IDF (Term Frequency – Inverse Document Frequency)**
    - It is a mathematical measure that assesses the relevance of a word in a collection of texts. This is achieved by multiplying two metrics: the number of times a word appears in a document and the inverse document frequency of the term over a collection of documents.
    - Why do I use this feature? The answer is that TF-IDF is particularly beneficial in encoding text data as vectors, and it has the potential to emphasize the value of words and weight unique and informative words accordingly.
    - The other features such as "bag-of-words", in that it considers every word equally, is given a document, certain words are common to be repeated more frequently than others. In TF-IDF feature, it's intended to indicate how significant a term is to a document in a collection of text or corpus.
    - As the main task is to assess the quality of argument, I used TF-IDF feature to be extracted.

- **Model Selection:**
    - **Support Vector Machine (SVM):**
    - When compared to alternative classifiers, such as the Naive Bayes model, SVM provides very high accuracy. Many people like it since it produces noticeable correctness with less compute power in a fair amount of time.
    - I utilized SVM because it can discover complex associations in my data without requiring to perform numerous changes.
    - One more strong reason to use TF-IDF + SVM is that SVM has performed very well on text classification as per the comparison done by [Pawar et al., 2012]
        - Pawar, Pratiksha Y., and S. H. Gawande. "A comparative study on different types of approaches to text categorization." International Journal of Machine Learning and Computing 2.4 (2012): 423-426.

Prerequisites for the code:

1. Libraries used: pandas, numpy, nltk and sklearn
2. Installed libraries using "pip install [library name]" command

Code step-by-step:

Step 1: Data Splitting

- By comparing the “id” field from both the datasets i.e., ‘.csv file’ and ‘.json file’, pulled all fields and bifurcated into Train and Test set.

Step 2: Data Pre-processing

- Blank spaces removal: I removed different special characters which are stored in “Separators” variable in the code.
- Converted some float data into string format for easy evaluation
- Lowercase conversion of all strings in list

Step 3: Token Vectorization of corpus and Feature Extraction using TF-IDF method

- Instead of manually implementing TF-IDF, I utilized the class given by sklearn.

```
#Vectorization of corpus using TF-IDF, Features extraction
Tfidf_vec = TfidfVectorizer()
Tfidf_vec.fit(Train_X)

Train_X_Tfidf = Tfidf_vec.transform(Train_X)
Test_X_Tfidf = Tfidf_vec.transform(Test_X)
```

Step 4: Parameter tuning for SVM machine:

•To find the best parameters for SVM, I defined a method called “finding_best_params” as shown below:

```
#following function was used to tunning the parameters of SVM machine
def finding_best_params(X_train,y_train,X_test,y_test):
    param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid','linear']}
    grid = GridSearchCV(svm.SVC(),param_grid,refit=True,verbose=2,cv=2)
    grid.fit(X_train,y_train)
    print(grid.best_estimator_)
    grid_predictions = grid.predict(X_test)
    print(confusion_matrix(y_test,grid_predictions))
    print(classification_report(y_test,grid_predictions))
```

- Using this step the function derived the best parameters which passed into SVM model definition.
- Used GridSearchCV function from scikitlearn for parameter tuning

Step 4: Implementation of machine learning model

- First, import the SVM module and create support vector classifier object by passing argument kernel as the sigmoid kernel (similar to the Neural Network) in SVC() function.
- Then, fit the model on train set using fit() and perform prediction on the test set using predict().

```
#SVM model
SVM = svm.SVC(C=1, gamma=1, kernel='sigmoid')
SVM.fit(Train_X_Tfidf, Train_Y)
predictions = SVM.predict(Test_X_Tfidf)
# print(predictions)
```

Step 5: After all the execution, exported all predictions to the “predictions.json” file.

```
1    [{"id": "373", "confirmation_bias": true}, {"id": "61", "confirmation_bias": true}, {"id": "180", "confirmation_b
```


For better readability, I formatted the json as below:


```
[{"id": "373", "confirmation_bias": true}, {"id": "61", "confirmation_bias": true}, {"id": "180", "confirmation_bias": true}, {"id": "211", "confirmation_bias": true}, {"id": "229", "confirmation_bias": true}, {"id": "278", "confirmation_bias": true}, {"id": "218",
"confirmation_bias": true}, {"id": "221", "confirmation_bias": true}, {"id": "306", "confirmation_bias": true}, {"id": "301", "confirmation_bias": true}, {"id": "21", "confirmation_bias": true}, {"id": "182", "confirmation_bias": true}, {"id": "289",
"confirmation_bias": true}, {"id": "364", "confirmation_bias": true}, {"id": "243", "confirmation_bias": true}, {"id": "108", "confirmation_bias": true}, {"id": "104", "confirmation_bias": false}, {"id": "136", "confirmation_bias": true}, {"id": "393",
"confirmation_bias": true}, {"id": "259", "confirmation_bias": true}, {"id": "72", "confirmation_bias": true}, {"id": "169", "confirmation_bias": true}, {"id": "98", "confirmation_bias": true}, {"id": "316", "confirmation_bias": true}, {"id": "382",
"confirmation_bias": true}, {"id": "103", "confirmation_bias": true}, {"id": "287", "confirmation_bias": true}, {"id": "352", "confirmation_bias": true}, {"id": "328", "confirmation_bias": true}, {"id": "202", "confirmation_bias": true}, {"id": "266",
"confirmation_bias": true}, {"id": "386", "confirmation_bias": true}, {"id": "142", "confirmation_bias": false}, {"id": "192", "confirmation_bias": true}, {"id": "4", "confirmation_bias": true}, {"id": "359", "confirmation_bias": true}, {"id": "117",
"confirmation_bias": true}, {"id": "331", "confirmation_bias": true}, {"id": "160", "confirmation_bias": true}, {"id": "310", "confirmation_bias": true}, {"id": "68", "confirmation_bias": true}, {"id": "241", "confirmation_bias": true}, {"id": "265",
"confirmation_bias": true}, {"id": "6", "confirmation_bias": true}, {"id": "86", "confirmation_bias": true}, {"id": "252", "confirmation_bias": true}, {"id": "255", "confirmation_bias": false}, {"id": "119", "confirmation_bias": false}, {"id": "227",
"confirmation_bias": false}, {"id": "204", "confirmation_bias": true}, {"id": "71", "confirmation_bias": false}, {"id": "234", "confirmation_bias": true}, {"id": "245", "confirmation_bias": true}, {"id": "154", "confirmation_bias": true}, {"id": "126",
"confirmation_bias": true}, {"id": "129", "confirmation_bias": true}, {"id": "42", "confirmation_bias": true}, {"id": "187", "confirmation_bias": true}, {"id": "240", "confirmation_bias": true}, {"id": "322", "confirmation_bias": true}, {"id": "193",
"confirmation_bias": true}, {"id": "97", "confirmation_bias": true}, {"id": "172", "confirmation_bias": true}, {"id": "5", "confirmation_bias": true}, {"id": "52", "confirmation_bias": false}, {"id": "163", "confirmation_bias": true}, {"id": "139",
"confirmation_bias": true}, {"id": "149", "confirmation_bias": true}, {"id": "277", "confirmation_bias": true}, {"id": "341", "confirmation_bias": true}, {"id": "212", "confirmation_bias": true}, {"id": "199", "confirmation_bias": true}, {"id": "77",
"confirmation_bias": true}, {"id": "348", "confirmation_bias": true}, {"id": "220", "confirmation_bias": true}, {"id": "82", "confirmation_bias": true}, {"id": "398", "confirmation_bias": true}, {"id": "335", "confirmation_bias": false}, {"id": "91",
"confirmation_bias": false}, {"id": "355", "confirmation_bias": false}]
```


Step 6: Evaluation phase

- Exported our “predictions.json” file into given evaluation.py file to retrieve scores on the test set for our predicitons.
- If the evaluation code does not run then add parameter encoding = 'utf-8' while loading the corpus “json.file” in evaluation.py
- As mentioned below our F1-score beats the baseline score.

With Tf_idf + naïve bayes:

F1-score: 0779

With Tf-idf + SVM: → **Better model than Naïve Bayes**

F1-score: 0843