Natural Language Processing

Homework – 3

עבאס אסמאעיל – 214742025

סלאם קייס - 327876116

At the start of the code, we have all our imports, and we define a few variables:

```
CHUNK_SIZE = 5 # Number of sentences in each chunk
K = 3 # Number of neighbors for KNN
TOP_N_WORDS = 3000  # Number of most used words to consider for custom features
```

We will explain them later, we also suppress some warnings because the pandas module kept printing out that some methods will change in the future.

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import pandas as pd
import numpy as np
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from collections import Counter
```

We also set the random seed.

1- We include the file and import it, then we split it into two groups:

```
# Part 1: Load the data
df = pd.read_json('kneeset_corpus.jsonl', lines=True, encoding='utf-8')
#print(df)
```

We split it into two groups in the next section so it's cleaner.

2- Now we split into two groups, and we divide into chunks:

```
df = get_Chunks(df)
```

```
try:
    def get_Chunks(df):
        chunk_size = CHUNK_SIZE
        combined_rows = []

        # Process committee protocol type
        committee_group = df[df['protocol_type'] == 'committee']
        committee_sentences = committee_group['sentence_text'].tolist()
        num_committee_chunks = len(committee_sentences) // chunk_size
        for i in range(num_committee_chunks):
            chunk_sentences = ' '.join(committee_sentences[i*chunk_size:(i+1)*chunk_size])
            combined_rows.append({'protocol_type': 'committee', 'sentence_text': chunk_sentences})

        # Process plenary protocol type
        plenary_group = df[df['protocol_type'] == 'plenary']
        plenary_sentences = plenary_group['sentence_text'].tolist()
        num_plenary_chunks = len(plenary_sentences) // chunk_size
        for i in range(num_plenary_chunks):
            chunk_sentences = ' '.join(plenary_sentences[i*chunk_size:(i+1)*chunk_size])
            combined_rows.append({'protocol_type': 'plenary', 'sentence_text': chunk_sentences})

        return pd.DataFrame(combined_rows)

except Exception as e:
    print(f'Error in get_Chunks: {e}')
```

We use the chunk_size variable to determine how many sentences we have in a chunk. In both cases we take a list of all the sentences then we calculate how many full chunks we will get, we go over that number in a loop and combine the sentences into chunks, and save them in the format of:

```
{'protocol_type': 'committee', 'sentence_text': chunk_sentences}
```

We combine both groups into one data frame and return it. We don't take incomplete chunks since we get rid of the remainder.

3- Now we begin down-sampling:

```python
# Part 3: down sampling
# Get the indexes of the down sample
committee_indexes = df[df['protocol_type'] == 'committee'].index
plenary_indexes = df[df['protocol_type'] == 'plenary'].index
#print("Len of original committee", len(committee_indexes))
#print("Len of original plenary",len(plenary_indexes))

# Down sample the majority class
down_sampled_indexes = np.random.choice(plenary_indexes, size=len(committee_indexes), replace=False)
#print("Len of new plenary",len(down_sampled_indexes))

# Combine the indexes
combined_indexes = np.concatenate([committee_indexes, down_sampled_indexes])
#print(combined_indexes)
# Get the down sampled dataframe
df = df.loc[combined_indexes].reset_index(drop=True)
#print(df)
```

This part is included in the Main function, we get the indexes of both groups and perform down-sampling with the random seed we included at the start.
These are the results:

```
Len of original committee 8341
Len of original plenary 13606
Len of new plenary 8341
```

4-

1: We were presented with the choice of using Count-Vectorizer or Tfidf so we tried out both. In section 5 we will see the results in detail, to sum it up: Count-Vectorizer simply sums up the count of the words, while Tfidf accounts for the importance of each word.
This tells us that using Tfidf can lead to more accurate results, so we expect it to perform better. Code:

```
# Part 4.1: Feature vector

chunks = df['sentence_text']
kneeset_numbers = df['knesset_number']
count_vectorizer = CountVectorizer()
count_vectorizer.fit(chunks)
count_features = count_vectorizer.transform(chunks)


tfid_vectorizer = TfidfVectorizer()
tfid_vectorizer.fit(chunks)
tfid_features = tfid_vectorizer.transform(chunks)
```

We fit the chunks to both vectorizers and then we get their features to test

In part 5.

2: **Method 1:**

We tried out many different methods, we thought about checking the 10 most common terms in both files to check if we have a specific pattern. That didn't work since they were both similar. We tried checking the highest percentage of use for each term and the highest count. We also tried including chunk length.

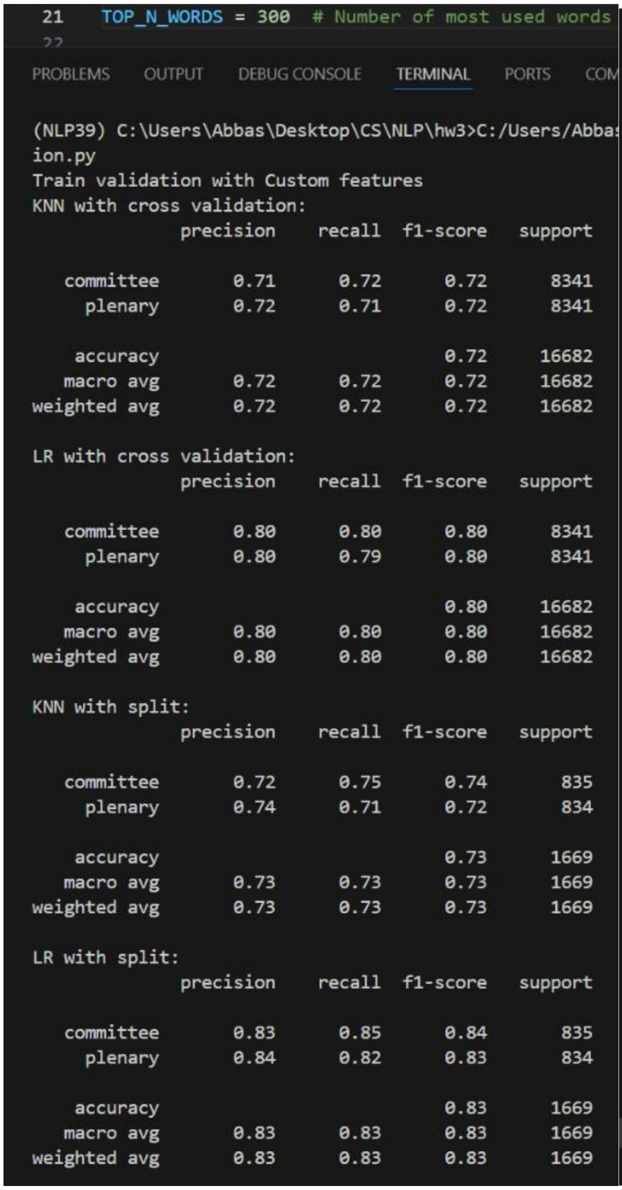
```
# Calculate top words of different protocol types to find our features
top_words = calculate_top_words(df['sentence_text'])
```

```
try:
    def calculate_word_counts(text):
        all_words = ' '.join(text).split()
        word_counts = Counter(all_words)
        return word_counts

except Exception as e:
    print(f'Error in calculate_word_counts: {e}')


try:
    def calculate_top_words(text, top_n=TOP_N_WORDS):
        all_words = ' '.join(text).split()
        word_counts = Counter(all_words)
        top_words = [word for word, _ in word_counts.most_common(top_n)]
        return top_words

except Exception as e:
    print(f'Error in calculate_top_words: {e}')
```

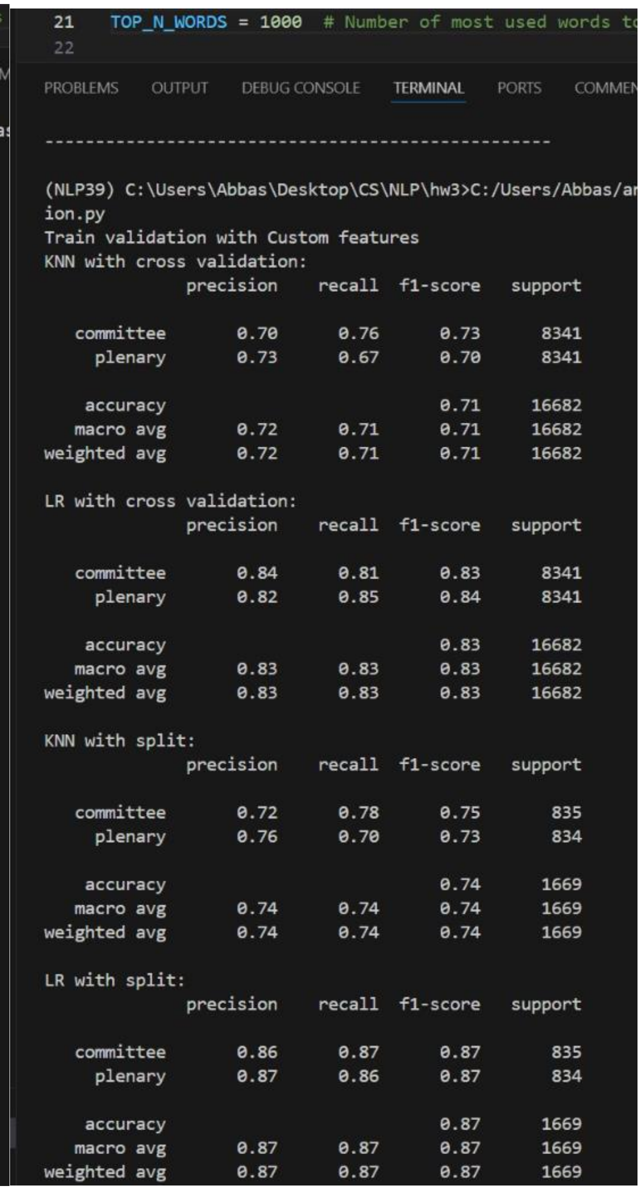
TOP_N_WORDS is a variable that holds the number of most common words of both files. Our current feature vector counts the number of occurrences of each of the top_words of both files

```
try:
    def extract_custom_features(chunks, custom_words):
        features = []
        # Calculate number of occurrences of custom words in each sentence

        # Calculate average length of chunk
        avg_length = sum(len(chunk.split()) for chunk in chunks) / len(chunks)

        for chunk in chunks:
            word_counts = Counter(chunk.split())
            chunk_features = [word_counts.get(word, 0) for word in custom_words]
            #chunk_features.append(avg_length) # Average length of chunk
            chunk_features.append(len(chunk)) # Length of chunk
            features.append(chunk_features)

        return np.array(features)

except Exception as e:
    print(f'Error in extract_custom_features: {e}')
```

We experienced with many different sizes during the testing phase, and these are some results:



```
21    TOP_N_WORDS = 300  # Number of most used words

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COM

(NLP39) C:\Users\Abbas\Desktop\CS\NLP\hw3>C:/Users/Abbas
ion.py
Train validation with Custom features
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.71      0.72      0.72      8341
     plenary       0.72      0.71      0.72      8341

    accuracy                           0.72     16682
   macro avg       0.72      0.72      0.72     16682
weighted avg       0.72      0.72      0.72     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.80      0.80      0.80      8341
     plenary       0.80      0.79      0.80      8341

    accuracy                           0.80     16682
   macro avg       0.80      0.80      0.80     16682
weighted avg       0.80      0.80      0.80     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.72      0.75      0.74       835
     plenary       0.74      0.71      0.72       834

    accuracy                           0.73      1669
   macro avg       0.73      0.73      0.73      1669
weighted avg       0.73      0.73      0.73      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.83      0.85      0.84       835
     plenary       0.84      0.82      0.83       834

    accuracy                           0.83      1669
   macro avg       0.83      0.83      0.83      1669
weighted avg       0.83      0.83      0.83      1669
```



```
21    TOP_N_WORDS = 1000  # Number of most used words to
22

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMEN

------------------------------------------------

(NLP39) C:\Users\Abbas\Desktop\CS\NLP\hw3>C:/Users/Abbas/ar
ion.py
Train validation with Custom features
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.70      0.76      0.73      8341
     plenary       0.73      0.67      0.70      8341

    accuracy                           0.71     16682
   macro avg       0.72      0.71      0.71     16682
weighted avg       0.72      0.71      0.71     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.84      0.81      0.83      8341
     plenary       0.82      0.85      0.84      8341

    accuracy                           0.83     16682
   macro avg       0.83      0.83      0.83     16682
weighted avg       0.83      0.83      0.83     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.72      0.78      0.75       835
     plenary       0.76      0.70      0.73       834

    accuracy                           0.74      1669
   macro avg       0.74      0.74      0.74      1669
weighted avg       0.74      0.74      0.74      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.86      0.87      0.87       835
     plenary       0.87      0.86      0.87       834

    accuracy                           0.87      1669
   macro avg       0.87      0.87      0.87      1669
weighted avg       0.87      0.87      0.87      1669
```

Increasing the size even further:

```
21    TOP_N_WORDS = 2000  # Number of most used words
22

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMM

---------------------------------------------------
Train validation with TFID features
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.77      0.79      0.78      8341
     plenary       0.79      0.76      0.77      8341

    accuracy                           0.78     16682
   macro avg       0.78      0.78      0.78     16682
weighted avg       0.78      0.78      0.78     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.84      0.82      0.83      8341
     plenary       0.82      0.85      0.83      8341

    accuracy                           0.83     16682
   macro avg       0.83      0.83      0.83     16682
weighted avg       0.83      0.83      0.83     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.81      0.92      0.86       835
     plenary       0.90      0.79      0.84       834

    accuracy                           0.85      1669
   macro avg       0.86      0.85      0.85      1669
weighted avg       0.86      0.85      0.85      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.87      0.90      0.88       835
     plenary       0.89      0.87      0.88       834

    accuracy                           0.88      1669
   macro avg       0.88      0.88      0.88      1669
weighted avg       0.88      0.88      0.88      1669
```

```
21    TOP_N_WORDS = 10000  # Number of most used wor
22

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COM

ion.py
Train validation with Custom features
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.66      0.76      0.71      8341
     plenary       0.72      0.61      0.66      8341

    accuracy                           0.69     16682
   macro avg       0.69      0.69      0.68     16682
weighted avg       0.69      0.69      0.68     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.89      0.81      0.85      8341
     plenary       0.83      0.90      0.86      8341

    accuracy                           0.85     16682
   macro avg       0.86      0.85      0.85     16682
weighted avg       0.86      0.85      0.85     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.68      0.72      0.70       835
     plenary       0.70      0.67      0.69       834

    accuracy                           0.69      1669
   macro avg       0.69      0.69      0.69      1669
weighted avg       0.69      0.69      0.69      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.91      0.91      0.91       835
     plenary       0.91      0.91      0.91       834

    accuracy                           0.91      1669
   macro avg       0.91      0.91      0.91      1669
weighted avg       0.91      0.91      0.91      1669
```

Increasing the number of words barley improved the accuracy and it got closer and closer to counter vectorizer, not to mention that using a huge vector like this may lead to overfitting that will occur if the files have a specific word repeated a lot of times by chance.

Instead, we had to try a new approach.

Method 2:

We only include words that appear significantly higher in one file type.

This code helps us the extract the absolute difference of occurrences per word for both files:

```
# Separate sentences by protocol type
committee_sentences = df[df['protocol_type'] == 'committee']['sentence_text'].tolist()
plenary_sentences = df[df['protocol_type'] == 'plenary']['sentence_text'].tolist()

# Calculate word count for each protocol type
committee_word_counts = calculate_word_counts(committee_sentences)
plenary_word_counts = calculate_word_counts(plenary_sentences)

# Find words with a count difference greater than threshold
significant_words = find_significant_word_differences(committee_word_counts, plenary_word_counts, threshold=0)
# sort the significant words by the difference
significant_words = dict(sorted(significant_words.items(), key=lambda item: item[1]['difference'], reverse=True))

# Print and save the significant words
with open('significant_words2.txt', 'w', encoding='utf-8') as file:
    for word, data in significant_words.items():
        line = (f'{word}: Committee={data["committee"]}, '
                f'Plenary={data["plenary"]}, Difference={data["difference"]}')
        file.write(line + '\n')
```

We take the committee and plenary sentences separately after down sampling, then we calculate the word counts for each group.

After that we use this function to extract the “Significant words”:

```
try:
    # Function to find the words with a count difference greater than 20
    def find_significant_word_differences(committee_counts, plenary_counts, threshold=20):
        significant_words = {}
        for word, count in committee_counts.items():
            if word in plenary_counts:
                diff = abs(count - plenary_counts[word])
                if diff > threshold:
                    significant_words[word] = {
                        'committee': count,
                        'plenary': plenary_counts[word],
                        'difference': diff
                    }
        return significant_words

except Exception as e:
    print(f'Error in find_significant_word_differences: {e}')
```

We check if the absolute difference between the counts for each word is over a certain threshold and we save them to a file.

Produced file:

```
,: Committee=34755, Plenary=52922, Difference=18167
הכנסת: Committee=1232, Plenary=8529, Difference=7297
זה: Committee=9014, Plenary=4616, Difference=4398
חבר: Committee=657, Plenary=4288, Difference=3631
לא: Committee=11458, Plenary=8954, Difference=2504
-: Committee=2241, Plenary=4546, Difference=2305
אדוני: Committee=517, Plenary=2673, Difference=2156
חברי: Committee=320, Plenary=2210, Difference=1890
.: Committee=36524, Plenary=34833, Difference=1691
אנחנו: Committee=3967, Plenary=2332, Difference=1635
היושב-ראש: Committee=173, Plenary=1588, Difference=1415
את: Committee=11725, Plenary=10329, Difference=1396
יש: Committee=3724, Plenary=2508, Difference=1216
אני: Committee=8472, Plenary=7296, Difference=1176
```

- This code is commented in the code, so it won't create the txt file

We analyzed the file and decided to include a few specific words with a high difference that we believe will have a big impact. We saved them in a list and we can use the previous custom_features function:

```
# Part 4.2: Custom features
custom_words = ['הכנסת', 'חבר', 'אדוני','חברי', 'אנחנו','היושב-ראש', 'אני','הצעת', 'חוק','ישראל','הממשלה','השר','צריך','בבקשה']
#custom_features = extract_custom_features(sentences, top_words)
custom_features = extract_custom_features(sentences, custom_words)
```

While testing in the next section we also tried including the length of the chunk as a feature for each chunk, but it slightly made the accuracy worse, so we removed it.

Results without length:

```
Train validation with Custom features
KNN cross-validation scores: 0.663589458070942
LR cross-validation scores: 0.7281520543229785
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.61      0.94      0.74      8341
     plenary       0.86      0.39      0.54      8341

    accuracy                           0.66     16682
   macro avg       0.73      0.66      0.64     16682
weighted avg       0.73      0.66      0.64     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.69      0.84      0.75      8341
     plenary       0.79      0.62      0.70      8341

    accuracy                           0.73     16682
   macro avg       0.74      0.73      0.72     16682
weighted avg       0.74      0.73      0.72     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.66      0.80      0.72       835
     plenary       0.75      0.58      0.65       834

    accuracy                           0.69      1669
   macro avg       0.70      0.69      0.69      1669
weighted avg       0.70      0.69      0.69      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.69      0.84      0.75       835
     plenary       0.79      0.62      0.69       834

    accuracy                           0.73      1669
   macro avg       0.74      0.73      0.72      1669
weighted avg       0.74      0.73      0.72      1669
```

Results with length:

```
Train validation with Custom features
KNN cross-validation scores: 0.6539390483417881
LR cross-validation scores: 0.7286314909714422
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.62      0.79      0.70      8341
     plenary       0.71      0.52      0.60      8341

    accuracy                           0.65     16682
   macro avg       0.67      0.65      0.65     16682
weighted avg       0.67      0.65      0.65     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.69      0.83      0.75      8341
     plenary       0.79      0.62      0.70      8341

    accuracy                           0.73     16682
   macro avg       0.74      0.73      0.73     16682
weighted avg       0.74      0.73      0.73     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.62      0.71      0.66       835
     plenary       0.66      0.57      0.61       834

    accuracy                           0.64      1669
   macro avg       0.64      0.64      0.63      1669
weighted avg       0.64      0.64      0.63      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.69      0.83      0.75       835
     plenary       0.79      0.62      0.69       834

    accuracy                           0.73      1669
   macro avg       0.74      0.73      0.72      1669
weighted avg       0.74      0.73      0.72      1669
```

We wanted to include more features from the corpus to improve the accuracy, we wouldn't include the names since that will ruin the point of using the model. That means were left with speaker names or Knesset number.

Using the speaker's name wouldn't help much. First it will cause overfitting, for example if we only have the name "Mike" in plenary and we give the model a new from a committee file with the same speaker name it will be misclassified. Secondly it doesn't make much sense because in the classification task we don't have the speaker's name attached to the chunk.

The same can be said for the Knesset number, but let's add it anyway for the sake of testing.

Results:

```
Train validation with Custom features
KNN cross-validation scores: 0.6681472861866335
LR cross-validation scores: 0.7592797922285486
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.66      0.70      0.68      8341
     plenary       0.68      0.63      0.66      8341

    accuracy                           0.67     16682
   macro avg       0.67      0.67      0.67     16682
weighted avg       0.67      0.67      0.67     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.78      0.73      0.75      8341
     plenary       0.74      0.79      0.77      8341

    accuracy                           0.76     16682
   macro avg       0.76      0.76      0.76     16682
weighted avg       0.76      0.76      0.76     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.78      0.81      0.80       835
     plenary       0.80      0.77      0.79       834

    accuracy                           0.79      1669
   macro avg       0.79      0.79      0.79      1669
weighted avg       0.79      0.79      0.79      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.81      0.81      0.81       835
     plenary       0.81      0.81      0.81       834

    accuracy                           0.81      1669
   macro avg       0.81      0.81      0.81      1669
weighted avg       0.81      0.81      0.81      1669
```

We can see that the accuracy improved as expected, but so did the overfitting.

5- This is how we train our models, for every feature vector we run a KNN model and a LR model, once with cross-validation and once with train-test-splitting.

```
for current_features in [custom_features, count_features, tfid_features]:
    # Part 5: Training models
    labels = df['protocol_type']
    if current_features is count_features:
        curr = 'Count features'
    elif current_features is tfid_features:
        curr = 'TFID features'
    else:
        curr = 'Custom features'

    print('Train validation with '+curr)


    # Models
    KNN = KNeighborsClassifier(K)
    LR = LogisticRegression(max_iter=10000)  # Added max_iter to ensure convergence
```

We go over each vector and print the corresponding message, and we define the models.

```
# 5 fold cross validation

knn_scores = cross_val_score(KNN, current_features, labels, cv=5)
lr_scores = cross_val_score(LR, current_features, labels, cv=5)

print(f'KNN cross-validation scores: {knn_scores.mean()}')
print(f'LR cross-validation scores: {lr_scores.mean()}')

y_pred = cross_val_predict(KNN, current_features, labels, cv=5)
print(f'KNN with cross validation:')
print(classification_report(labels, y_pred))

y_pred = cross_val_predict(LR, current_features, labels, cv=5)
print(f'LR with cross validation:')
print(classification_report(labels, y_pred))
```

Here we use 5-fold cross validation over both the models.

```python
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(current_features, labels, test_size=0.1, random_state=42,stratify=labels)
KNN.fit(X_train,y_train)
LR.fit(X_train, y_train)
if current_features is custom_features:
    best_model = LR

print(f'KNN with split:')
y_pred = KNN.predict(X_test)
print(classification_report(y_test, y_pred))

print(f'LR with split:')
y_pred = LR.predict(X_test)
print(classification_report(y_test, y_pred))
print('--------------------------------------------------')
```

Here we used train-test-split over both the models.

We will only focus on Tfidf and Count-Vectorizer since they had the best results, we tried different k's using this code:

```python
for k in [3,5,10,15,20,25,50,100]:
    print(f'KNN with {k} neighbors and count features:')
    KNN = KNeighborsClassifier(k)
    knn_scores = cross_val_score(KNN, count_features, labels, cv=5)
    print(f'KNN with {k} neighbors cross-validation scores: {knn_scores.mean()}')

    X_train, X_test, y_train, y_test = train_test_split(count_features, labels, test_size=0.1, random_state=42,stratify=labels)
    KNN.fit(X_train,y_train)
    print(f'KNN with {k} neighbors with split:')
    y_pred = KNN.predict(X_test)
    print(classification_report(y_test, y_pred))

    print('--------------------------------------------------')

    print(f'KNN with {k} neighbors and tfidf features:')
    KNN = KNeighborsClassifier(k)
    knn_scores = cross_val_score(KNN, tfid_features, labels, cv=5)
    print(f'KNN with {k} neighbors cross-validation scores: {knn_scores.mean()}')

    X_train, X_test, y_train, y_test = train_test_split(tfid_features, labels, test_size=0.1, random_state=42,stratify=labels)
    KNN.fit(X_train,y_train)
    print(f'KNN with {k} neighbors with split:')
    y_pred = KNN.predict(X_test)
    print(classification_report(y_test, y_pred))

    print('--------------------------------------------------')
```

Based on the results included in the next page we can see that increasing k doesn't help us. We left it at 3.

```
--------------------------------------------------
KNN with 3 neighbors and count features:
KNN with 3 neighbors cross-validation scores: 0.6405108337663104
KNN with 3 neighbors with split:
              precision    recall  f1-score   support

   committee       0.59      0.88      0.71       835
     plenary       0.76      0.39      0.52       834

    accuracy                           0.63      1669
   macro avg       0.68      0.63      0.61      1669
weighted avg       0.68      0.63      0.61      1669

--------------------------------------------------
KNN with 3 neighbors and tfidf features:
KNN with 3 neighbors cross-validation scores: 0.7446382899673669
KNN with 3 neighbors with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

--------------------------------------------------
```

```
--------------------------------------------------
KNN with 5 neighbors and count features:
KNN with 5 neighbors cross-validation scores: 0.650462171467501
KNN with 5 neighbors with split:
              precision    recall  f1-score   support

   committee       0.60      0.92      0.72       835
     plenary       0.83      0.38      0.52       834

    accuracy                           0.65      1669
   macro avg       0.71      0.65      0.62      1669
weighted avg       0.71      0.65      0.62      1669

--------------------------------------------------
KNN with 5 neighbors and tfidf features:
KNN with 5 neighbors cross-validation scores: 0.7565671646081398
KNN with 5 neighbors with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.85       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

--------------------------------------------------
```

```
--------------------------------------------------
KNN with 10 neighbors and count features:
KNN with 10 neighbors cross-validation scores: 0.6415900962179013
KNN with 10 neighbors with split:
              precision    recall  f1-score   support

   committee       0.58      0.96      0.73       835
     plenary       0.89      0.30      0.45       834

    accuracy                           0.63      1669
   macro avg       0.74      0.63      0.59      1669
weighted avg       0.74      0.63      0.59      1669

--------------------------------------------------
KNN with 10 neighbors and tfidf features:
KNN with 10 neighbors cross-validation scores: 0.7780879701393215
KNN with 10 neighbors with split:
              precision    recall  f1-score   support

   committee       0.81      0.92      0.86       835
     plenary       0.90      0.79      0.84       834

    accuracy                           0.85      1669
   macro avg       0.86      0.85      0.85      1669
weighted avg       0.86      0.85      0.85      1669

--------------------------------------------------
```

```
--------------------------------------------------
KNN with 15 neighbors and tfidf features:
KNN with 15 neighbors cross-validation scores: 0.7817444336409805
KNN with 15 neighbors with split:
              precision    recall  f1-score   support

   committee       0.84      0.89      0.86       835
     plenary       0.88      0.83      0.85       834

    accuracy                           0.86      1669
   macro avg       0.86      0.86      0.86      1669
weighted avg       0.86      0.86      0.86      1669

--------------------------------------------------
KNN with 20 neighbors and count features:
KNN with 20 neighbors cross-validation scores: 0.6419498982773625
KNN with 20 neighbors with split:
              precision    recall  f1-score   support

   committee       0.58      0.97      0.72       835
     plenary       0.90      0.30      0.45       834

    accuracy                           0.63      1669
   macro avg       0.74      0.63      0.59      1669
weighted avg       0.74      0.63      0.59      1669

--------------------------------------------------
```

```
--------------------------------------------------
KNN with 20 neighbors and count features:
KNN with 20 neighbors cross-validation scores: 0.6419498982773625
KNN with 20 neighbors with split:
              precision    recall  f1-score   support

   committee       0.58      0.97      0.72       835
     plenary       0.90      0.30      0.45       834

    accuracy                           0.63      1669
   macro avg       0.74      0.63      0.59      1669
weighted avg       0.74      0.63      0.59      1669

--------------------------------------------------
KNN with 20 neighbors and tfidf features:
KNN with 20 neighbors cross-validation scores: 0.7865400757009017
KNN with 20 neighbors with split:
              precision    recall  f1-score   support

   committee       0.81      0.91      0.86       835
     plenary       0.90      0.79      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

--------------------------------------------------
```

```
--------------------------------------------------
KNN with 100 neighbors and count features:
KNN with 100 neighbors cross-validation scores: 0.6327782784261053
KNN with 100 neighbors with split:
              precision    recall  f1-score   support

   committee       0.57      0.97      0.72       835
     plenary       0.91      0.27      0.41       834

    accuracy                           0.62      1669
   macro avg       0.74      0.62      0.57      1669
weighted avg       0.74      0.62      0.57      1669

--------------------------------------------------
KNN with 100 neighbors and tfidf features:
KNN with 100 neighbors cross-validation scores: 0.7859398007515475
KNN with 100 neighbors with split:
              precision    recall  f1-score   support

   committee       0.78      0.91      0.84       835
     plenary       0.89      0.74      0.81       834

    accuracy                           0.82      1669
   macro avg       0.83      0.82      0.82      1669
weighted avg       0.83      0.82      0.82      1669

--------------------------------------------------
```

We tried a few parameters for LR and it made it worse so removed them:

Before:

```
Train validation with TFID features
KNN cross-validation scores: 0.7446382899673669
LR cross-validation scores: 0.8319781154399226
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.79      0.67      0.72      8341
     plenary       0.71      0.82      0.76      8341

    accuracy                           0.74     16682
   macro avg       0.75      0.74      0.74     16682
weighted avg       0.75      0.74      0.74     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.84      0.82      0.83      8341
     plenary       0.82      0.85      0.83      8341

    accuracy                           0.83     16682
   macro avg       0.83      0.83      0.83     16682
weighted avg       0.83      0.83      0.83     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.87      0.90      0.88       835
     plenary       0.89      0.87      0.88       834

    accuracy                           0.88      1669
   macro avg       0.88      0.88      0.88      1669
weighted avg       0.88      0.88      0.88      1669
```

After:

```
Train validation with TFID features
KNN cross-validation scores: 0.7446382899673669
LR cross-validation scores: 0.8107580941539846
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.79      0.67      0.72      8341
     plenary       0.71      0.82      0.76      8341

    accuracy                           0.74     16682
   macro avg       0.75      0.74      0.74     16682
weighted avg       0.75      0.74      0.74     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.82      0.80      0.81      8341
     plenary       0.80      0.83      0.81      8341

    accuracy                           0.81     16682
   macro avg       0.81      0.81      0.81     16682
weighted avg       0.81      0.81      0.81     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.84      0.86      0.85       835
     plenary       0.86      0.84      0.85       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669
```

So far here's our results for this section:

```
Train validation with Count features
KNN cross-validation scores: 0.6405108337663104
LR cross-validation scores: 0.8292810282789651
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.59      0.90      0.71      8341
     plenary       0.79      0.38      0.51      8341

    accuracy                           0.64     16682
   macro avg       0.69      0.64      0.61     16682
weighted avg       0.69      0.64      0.61     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.86      0.79      0.82      8341
     plenary       0.80      0.87      0.84      8341

    accuracy                           0.83     16682
   macro avg       0.83      0.83      0.83     16682
weighted avg       0.83      0.83      0.83     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.59      0.88      0.71       835
     plenary       0.76      0.39      0.52       834

    accuracy                           0.63      1669
   macro avg       0.68      0.63      0.61      1669
weighted avg       0.68      0.63      0.61      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.89      0.90      0.89       835
     plenary       0.90      0.89      0.89       834

    accuracy                           0.89      1669
   macro avg       0.89      0.89      0.89      1669
weighted avg       0.89      0.89      0.89      1669
```

```
Train validation with TFID features
KNN cross-validation scores: 0.7446382899673669
LR cross-validation scores: 0.8319781154399226
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.79      0.67      0.72      8341
     plenary       0.71      0.82      0.76      8341

    accuracy                           0.74     16682
   macro avg       0.75      0.74      0.74     16682
weighted avg       0.75      0.74      0.74     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.84      0.82      0.83      8341
     plenary       0.82      0.85      0.83      8341

    accuracy                           0.83     16682
   macro avg       0.83      0.83      0.83     16682
weighted avg       0.83      0.83      0.83     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.87      0.90      0.88       835
     plenary       0.89      0.87      0.88       834

    accuracy                           0.88      1669
   macro avg       0.88      0.88      0.88      1669
weighted avg       0.88      0.88      0.88      1669
```

6- We had to make a choice between using the Tfidf features or the Count-Vectorizer. In the first 3 models Tfidf beats Count-Vectorizer and it only gets beaten in the last model, therefore we will use Tfidf since it had a higher overall average accuracy.

We used Tfidf in combination with LR:

```
# Part 6: Classification
# Choose the best model and feature vector

with open(sentences_file, 'r', encoding='utf-8') as file:
    sentences = file.readlines()
    new_count_features = tfid_vectorizer.transform(sentences)
    predictions = best_model.predict(new_count_features)
    text = ''

for i, prediction in enumerate(predictions):
    text += prediction + '\n'
with open(os.path.join(output_dir, 'classification_results.txt'), 'w') as write_file:
    write_file.write(text)
```

We open the txt file, and we extract the features vectors then we give them to our model to predict. After that we simply write them down.

7- 1- Using the Chunk size variable to test different values, here’s what we got:

```
19    CHUNK_SIZE = 2 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CC

Train validation with TFID features
KNN cross-validation scores: 0.6866794665905689
LR cross-validation scores: 0.7716759358661668
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.70      0.65      0.67     20854
     plenary       0.67      0.73      0.70     20854

    accuracy                           0.69     41708
   macro avg       0.69      0.69      0.69     41708
weighted avg       0.69      0.69      0.69     41708

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.78      0.75      0.77     20854
     plenary       0.76      0.79      0.78     20854

    accuracy                           0.77     41708
   macro avg       0.77      0.77      0.77     41708
weighted avg       0.77      0.77      0.77     41708

KNN with split:
              precision    recall  f1-score   support

   committee       0.74      0.80      0.77      2086
     plenary       0.79      0.72      0.75      2085

    accuracy                           0.76      4171
   macro avg       0.77      0.76      0.76      4171
weighted avg       0.77      0.76      0.76      4171

LR with split:
              precision    recall  f1-score   support

   committee       0.80      0.85      0.82      2086
     plenary       0.84      0.78      0.81      2085

    accuracy                           0.82      4171
   macro avg       0.82      0.82      0.82      4171
weighted avg       0.82      0.82      0.82      4171
```

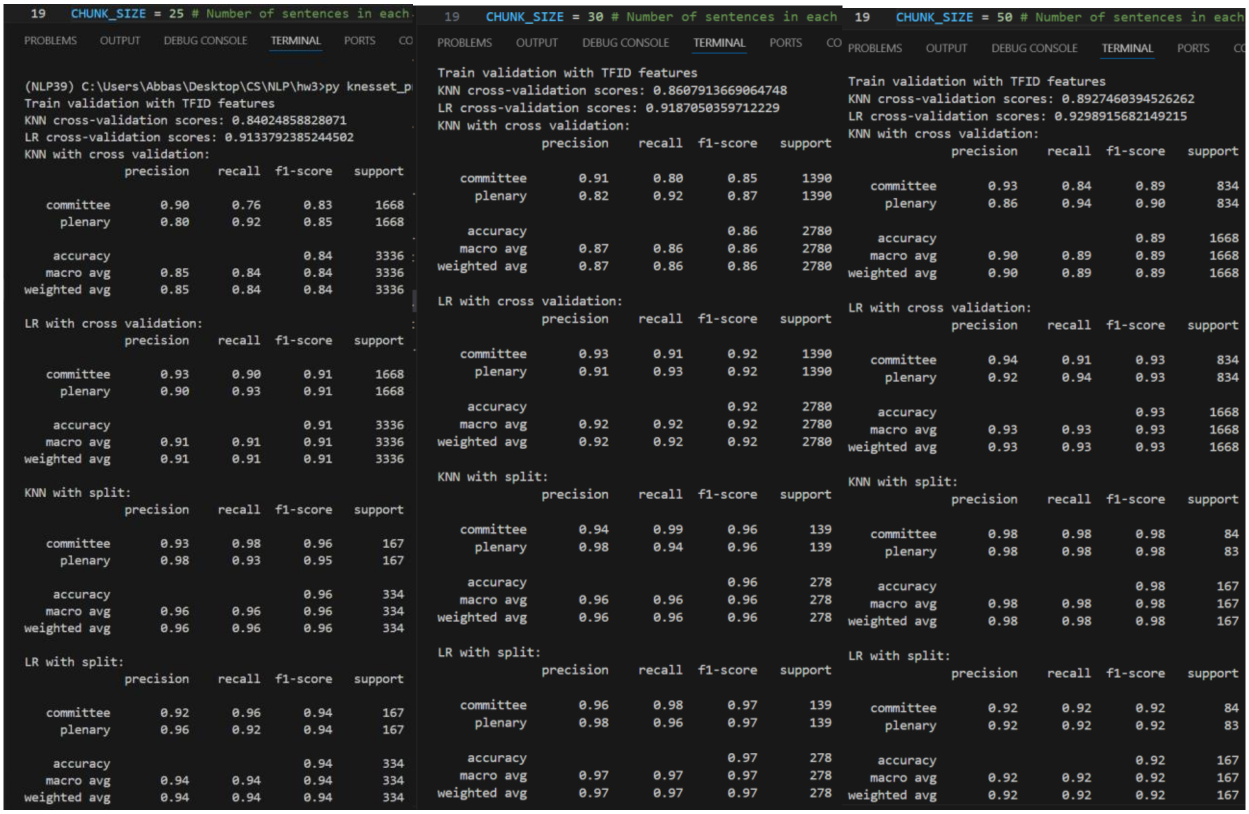
```
19    CHUNK_SIZE = 5 # Number of sentences in each c
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COI

--------------------------------------------------
Train validation with TFID features
KNN cross-validation scores: 0.7446382899673669
LR cross-validation scores: 0.8319781154399226
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.79      0.67      0.72      8341
     plenary       0.71      0.82      0.76      8341

    accuracy                           0.74     16682
   macro avg       0.75      0.74      0.74     16682
weighted avg       0.75      0.74      0.74     16682

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.84      0.82      0.83      8341
     plenary       0.82      0.85      0.83      8341

    accuracy                           0.83     16682
   macro avg       0.83      0.83      0.83     16682
weighted avg       0.83      0.83      0.83     16682

KNN with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669

LR with split:
              precision    recall  f1-score   support

   committee       0.87      0.90      0.88       835
     plenary       0.89      0.87      0.88       834

    accuracy                           0.88      1669
   macro avg       0.88      0.88      0.88      1669
weighted avg       0.88      0.88      0.88      1669
```

```
19    CHUNK_SIZE = 9 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CO

Train validation with TFID features
KNN cross-validation scores: 0.7780616988341016
LR cross-validation scores: 0.8672933072757025
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.84      0.69      0.76      4634
     plenary       0.74      0.87      0.80      4634

    accuracy                           0.78      9268
   macro avg       0.79      0.78      0.78      9268
weighted avg       0.79      0.78      0.78      9268

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.88      0.85      0.87      4634
     plenary       0.86      0.88      0.87      4634

    accuracy                           0.87      9268
   macro avg       0.87      0.87      0.87      9268
weighted avg       0.87      0.87      0.87      9268

KNN with split:
              precision    recall  f1-score   support

   committee       0.88      0.92      0.90       464
     plenary       0.91      0.88      0.90       463

    accuracy                           0.90       927
   macro avg       0.90      0.90      0.90       927
weighted avg       0.90      0.90      0.90       927

LR with split:
              precision    recall  f1-score   support

   committee       0.89      0.95      0.92       464
     plenary       0.94      0.89      0.91       463

    accuracy                           0.92       927
   macro avg       0.92      0.92      0.92       927
weighted avg       0.92      0.92      0.92       927
```

```
19    CHUNK_SIZE = 11 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COI

Train validation with TFID features
KNN cross-validation scores: 0.7937325961008309
LR cross-validation scores: 0.8753728630490327
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.86      0.70      0.77      3791
     plenary       0.75      0.88      0.81      3791

    accuracy                           0.79      7582
   macro avg       0.80      0.79      0.79      7582
weighted avg       0.80      0.79      0.79      7582

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.89      0.86      0.87      3791
     plenary       0.86      0.89      0.88      3791

    accuracy                           0.88      7582
   macro avg       0.88      0.88      0.88      7582
weighted avg       0.88      0.88      0.88      7582

KNN with split:
              precision    recall  f1-score   support

   committee       0.89      0.93      0.91       380
     plenary       0.93      0.88      0.91       379

    accuracy                           0.91       759
   macro avg       0.91      0.91      0.91       759
weighted avg       0.91      0.91      0.91       759

LR with split:
              precision    recall  f1-score   support

   committee       0.91      0.95      0.93       380
     plenary       0.94      0.91      0.92       379

    accuracy                           0.93       759
   macro avg       0.93      0.93      0.93       759
weighted avg       0.93      0.93      0.93       759
```

```
19    CHUNK_SIZE = 15 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CC

Train validation with TFID features
KNN cross-validation scores: 0.8138489208633093
LR cross-validation scores: 0.89568345323741
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.88      0.73      0.80      2780
     plenary       0.77      0.90      0.83      2780

    accuracy                           0.81      5560
   macro avg       0.82      0.81      0.81      5560
weighted avg       0.82      0.81      0.81      5560

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.91      0.88      0.89      2780
     plenary       0.88      0.91      0.90      2780

    accuracy                           0.90      5560
   macro avg       0.90      0.90      0.90      5560
weighted avg       0.90      0.90      0.90      5560

KNN with split:
              precision    recall  f1-score   support

   committee       0.94      0.96      0.95       278
     plenary       0.96      0.94      0.95       278

    accuracy                           0.95       556
   macro avg       0.95      0.95      0.95       556
weighted avg       0.95      0.95      0.95       556

LR with split:
              precision    recall  f1-score   support

   committee       0.94      0.96      0.95       278
     plenary       0.96      0.94      0.95       278

    accuracy                           0.95       556
   macro avg       0.95      0.95      0.95       556
weighted avg       0.95      0.95      0.95       556
```

```
19    CHUNK_SIZE = 20 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CC

Train validation with TFID features
KNN cross-validation scores: 0.834052757793765
LR cross-validation scores: 0.9004796163069544
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.90      0.75      0.82      2085
     plenary       0.79      0.91      0.85      2085

    accuracy                           0.83      4170
   macro avg       0.84      0.83      0.83      4170
weighted avg       0.84      0.83      0.83      4170

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.91      0.89      0.90      2085
     plenary       0.89      0.91      0.90      2085

    accuracy                           0.90      4170
   macro avg       0.90      0.90      0.90      4170
weighted avg       0.90      0.90      0.90      4170

KNN with split:
              precision    recall  f1-score   support

   committee       0.96      0.95      0.95       209
     plenary       0.95      0.96      0.95       208

    accuracy                           0.95       417
   macro avg       0.95      0.95      0.95       417
weighted avg       0.95      0.95      0.95       417

LR with split:
              precision    recall  f1-score   support

   committee       0.90      0.95      0.92       209
     plenary       0.95      0.89      0.92       208

    accuracy                           0.92       417
   macro avg       0.92      0.92      0.92       417
weighted avg       0.92      0.92      0.92       417
```



```
19   CHUNK_SIZE = 25 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CO

(NLP39) C:\Users\Abbas\Desktop\CS\NLP\hw3>py knesset_p
Train validation with TFID features
KNN cross-validation scores: 0.84024858828071
LR cross-validation scores: 0.9133792385244502
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.90      0.76      0.83      1668
     plenary       0.80      0.92      0.85      1668

    accuracy                           0.84      3336
   macro avg       0.85      0.84      0.84      3336
weighted avg       0.85      0.84      0.84      3336

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.93      0.90      0.91      1668
     plenary       0.90      0.93      0.91      1668

    accuracy                           0.91      3336
   macro avg       0.91      0.91      0.91      3336
weighted avg       0.91      0.91      0.91      3336

KNN with split:
              precision    recall  f1-score   support

   committee       0.93      0.98      0.96       167
     plenary       0.98      0.93      0.95       167

    accuracy                           0.96       334
   macro avg       0.96      0.96      0.96       334
weighted avg       0.96      0.96      0.96       334

LR with split:
              precision    recall  f1-score   support

   committee       0.92      0.96      0.94       167
     plenary       0.96      0.92      0.94       167

    accuracy                           0.94       334
   macro avg       0.94      0.94      0.94       334
weighted avg       0.94      0.94      0.94       334
```

```
19   CHUNK_SIZE = 30 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CO

Train validation with TFID features
KNN cross-validation scores: 0.8607913669064748
LR cross-validation scores: 0.9187050359712229
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.91      0.80      0.85      1390
     plenary       0.82      0.92      0.87      1390

    accuracy                           0.86      2780
   macro avg       0.87      0.86      0.86      2780
weighted avg       0.87      0.86      0.86      2780

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.93      0.91      0.92      1390
     plenary       0.91      0.93      0.92      1390

    accuracy                           0.92      2780
   macro avg       0.92      0.92      0.92      2780
weighted avg       0.92      0.92      0.92      2780

KNN with split:
              precision    recall  f1-score   support

   committee       0.94      0.99      0.96       139
     plenary       0.98      0.94      0.96       139

    accuracy                           0.96       278
   macro avg       0.96      0.96      0.96       278
weighted avg       0.96      0.96      0.96       278

LR with split:
              precision    recall  f1-score   support

   committee       0.96      0.98      0.97       139
     plenary       0.98      0.96      0.97       139

    accuracy                           0.97       278
   macro avg       0.97      0.97      0.97       278
weighted avg       0.97      0.97      0.97       278
```

```
19   CHUNK_SIZE = 50 # Number of sentences in each
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CO

Train validation with TFID features
KNN cross-validation scores: 0.8927460394526262
LR cross-validation scores: 0.9298915682149215
KNN with cross validation:
              precision    recall  f1-score   support

   committee       0.93      0.84      0.89       834
     plenary       0.86      0.94      0.90       834

    accuracy                           0.89      1668
   macro avg       0.90      0.89      0.89      1668
weighted avg       0.90      0.89      0.89      1668

LR with cross validation:
              precision    recall  f1-score   support

   committee       0.94      0.91      0.93       834
     plenary       0.92      0.94      0.93       834

    accuracy                           0.93      1668
   macro avg       0.93      0.93      0.93      1668
weighted avg       0.93      0.93      0.93      1668

KNN with split:
              precision    recall  f1-score   support

   committee       0.98      0.98      0.98        84
     plenary       0.98      0.98      0.98        83

    accuracy                           0.98       167
   macro avg       0.98      0.98      0.98       167
weighted avg       0.98      0.98      0.98       167

LR with split:
              precision    recall  f1-score   support

   committee       0.92      0.92      0.92        84
     plenary       0.92      0.92      0.92        83

    accuracy                           0.92       167
   macro avg       0.92      0.92      0.92       167
weighted avg       0.92      0.92      0.92       167
```

In conclusion the best accuracy was obtained via our best model (LR) with chunk size of 15, we must also be careful not to overfit/underfit the model to the data, so we didn't look for anything higher. We will explain why that chunk size may be the best in the next part.

2- If we drastically increase the size of the chunks, we will get less chunks in total which means we have less data to work with, this may result in overfitting since we don't have a lot of data to work with and that if our chunk contains more patterns then our model will be overfitted. On the plus side we will be able to capture patterns better since we have a larger chunk overall which means the entire pattern is highly likely to be inside one chunk.

On the other hand, if we decrease the size of the chunks, we will get the opposite of that. We will get way more chunks in total which on paper should deal better with overfitting, more seen data = better predictions on unseen data. However, we may not be able to detect patterns, a pattern could be split across 5 separate sentences but if we read 2 at a time, we won't be able to detect it appropriately, in other words we may underfit the data.

Having a chunk size of 15 may be the sweet spot. When we used chunk size up to 11, we got 0.93%, moving to 15 we get 0.95%, but when we try to increase it again to 20, we get a lower accuracy of 0.92% which may be a sign of overfitting.

Even though chunk size of 30 gave us 0.97% we won't pick it as the best since a lower number (20) may be overfitting, this can be inferred since we have less data and the sudden drop in accuracy.

Questions:

1- We would maximize the recall for the committee, this metric is smaller when we have more false positives and in our case, we don't want those, so by maximizing this metric we minimize the false positives and therefore we try to maximize the recall.
   We would use KNN with Count-Vectorizer (KNN with split got the highest recall) that we saw previously since it has the highest recall for committee:

```
--------------------------------------------------
KNN with 20 neighbors and count features:
KNN with 20 neighbors cross-validation scores: 0.6419498982773625
KNN with 20 neighbors with split:
              precision    recall  f1-score   support

   committee       0.58      0.97      0.72       835
     plenary       0.90      0.30      0.45       834

    accuracy                           0.63      1669
   macro avg       0.74      0.63      0.59      1669
weighted avg       0.74      0.63      0.59      1669

--------------------------------------------------
```

- In this case looking at the recall is more important/correct than just the accuracy, even though we have low accuracy, we only focus on the recall.

2- Here we want to get correct classification for everything, we don't want to get a single misclassification, this is exactly the accuracy, if we maximize the accuracy, we maximize the number of correct classifications for both plenary and committee and we also minimize the incorrect classifications for both.
We Will use the LR model with Tfidf (chunk size of 5):

LR with split:

	precision	recall	f1-score	support
committee	0.87	0.90	0.88	835
plenary	0.89	0.87	0.88	834
accuracy			0.88	1669
macro avg	0.88	0.88	0.88	1669
weighted avg	0.88	0.88	0.88	1669

3- Based on our tests they were very similar, but train-test-split had slightly higher accuracy, sometimes it wasn't very noticeable but sometimes it was, generally I believe cross validation should provide more reliable and accurate results since it considers multiple different sets to estimate how good a model is. This gives us a "stronger" estimate of how good our model is.

   If one of the splits had a high accuracy while the rest had lower accuracy, then train-test could have higher accuracy, and since cross validation considers every different set, it will be much lower. I believe this is why it happened. Also, test-train split is more prone to overfitting which could explain the results. Overfitting could happen in cases where the train test has no outliers or drastically different points, but they are included in the test set.

4- Starting with KNN: the advantages include simpleness and ease of use, KNN doesn't need a training phase and it's easy to understand the process, we just compare words and find the closest.

   Disadvantages include slow runtime, in datasets with a lot of points we will have to go over every single point which isn't very ideal. Also scaling plays a huge role in this model and it can greatly impact the results based on scaling factors such as compressing only one axis.
   Not only that but we also need to store all the dataset unlike other models that come up with an equation or other types of identifiers, and we must not forget that storing large datasets take up a lot of memory.

   Now for LR, the advantages include fast runtime for large datasets as well as the simplicity of the intuition behind it's idea.
   Disadvantages include that the model assumes that the data is linearly separable which could be an advantage in specific cases, this will cause issues if we have inseparable data, this model cannot detect complex relations or patterns in the data since it tries to separate it using a line.

   I believe in our case using LR is better than using KNN, it's significantly faster than KNN and after we checked the most common words we found a lot of similarities between both file types. KNN might not be able to detect that since the points will be close to each other and get classified in the same cluster.
   LR on the other hand might find a few features that will be sufficient to separate two different data points. We also saw this throughout our testing.