Natural Language processing – HW 3

Mias Ghantous – 213461692

Faisal Omari - 325616894

Section 1,2.

As we see, we split the data according to his type after making the chunks

```
random.seed(42)
np.random.seed(42)
# part 1,2
df = pd.read_csv('knesset_corpus.csv',index_col=None)
df=make_chunks(df)

#we need the indexes in down sample
committee_data = df.loc[df['protocol_type'] == 'committee'].reset_index(drop=True)
plenary_data = df.loc[df['protocol_type'] == 'plenary'].reset_index(drop=True)
```

Make_chunks:

1. we divide the data into groups by using the *groupby* function which, takes the name of the columns as a parameter and then make a group for every possible compensation of values from those columns.

2. second we use *apply* which takes a function as a parameter and do this function for each group, we sent *process* which extract the relevant data from the group and put them into a data frame.

3. apply concatenate the result by herself.

4. in the code we group by 'protocol_type' and 'protocol_name' because we want the keenest number in the classification

Section 3:

As we see, we call down sample for both of the data and then connect them and then randomize it:

```
#part 3
committee_data = down_sample(committee_data,len(committee_data)-len(plenary_data))
plenary_data = down_sample(plenary_data,len(plenary_data)-len(committee_data))

#connect the 2 types with randomness
data = pd.concat([committee_data,plenary_data])
data = data.sample(frac=1,random_state=42).reset_index(drop = True)
```

For chunk size 5 we get that:

The size of the committee data before the down sample was 5872 and also after.

The size of the plenary data 14095 and after the down sample we have 5872

Section 4.

1. We used TFIDF vectorizer because he can give more accurate predictions:
   (first is IFIDF and the is the counter)

```
BoW train validation
KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.80      0.96      0.87      5872
     plenary       0.95      0.77      0.85      5872

    accuracy                           0.86     11744
   macro avg       0.88      0.86      0.86     11744
weighted avg       0.88      0.86      0.86     11744

SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.91      0.93      0.92      5872
     plenary       0.92      0.91      0.92      5872

    accuracy                           0.92     11744
   macro avg       0.92      0.92      0.92     11744
weighted avg       0.92      0.92      0.92     11744

KNN with split:
              precision    recall  f1-score   support

   committee       0.81      0.96      0.88       588
     plenary       0.95      0.78      0.86       587

    accuracy                           0.87      1175
   macro avg       0.88      0.87      0.87      1175
weighted avg       0.88      0.87      0.87      1175

SVM with split:
              precision    recall  f1-score   support

   committee       0.91      0.93      0.92       588
     plenary       0.93      0.91      0.92       587

    accuracy                           0.92      1175
   macro avg       0.92      0.92      0.92      1175
weighted avg       0.92      0.92      0.92      1175
```

```
BoW train validation
KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.63      0.88      0.74      5872
     plenary       0.80      0.49      0.61      5872

    accuracy                           0.69     11744
   macro avg       0.72      0.69      0.67     11744
weighted avg       0.72      0.69      0.67     11744

SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.89      0.91      0.90      5872
     plenary       0.91      0.89      0.90      5872

    accuracy                           0.90     11744
   macro avg       0.90      0.90      0.90     11744
weighted avg       0.90      0.90      0.90     11744

KNN with split:
              precision    recall  f1-score   support

   committee       0.65      0.85      0.73       588
     plenary       0.78      0.53      0.63       587

    accuracy                           0.69      1175
   macro avg       0.71      0.69      0.68      1175
weighted avg       0.71      0.69      0.68      1175

SVM with split:
              precision    recall  f1-score   support

   committee       0.90      0.92      0.91       588
     plenary       0.92      0.89      0.91       587

    accuracy                           0.91      1175
   macro avg       0.91      0.91      0.91      1175
weighted avg       0.91      0.91      0.91      1175
```

2. for our vector we used 8 words:

```
word_list = ['חוק','תודה','חברי','ראש','אדוני','חבר','הכנסת','הצעת']
```

Because those words have a lot of occurrences in plenary documents compared to committee, how did I know? I wrote code that counts all the occurrences of all the word and took the words that has 3 times appearances in plenary and appear more than a 1000 times in all the plenary data: here is the code:

```
'''
p_Counter = CountVectorizer(vocabulary=vectorizer.vocabulary_)
c_Counter = CountVectorizer(vocabulary=vectorizer.vocabulary_)

P = p_Counter.fit_transform(plenary_data['sentence_text'])
C = c_Counter.fit_transform(committee_data['sentence_text'])

dic = {word: P[:,vectorizer.vocabulary_.get(word)].sum() /C[:,vectorizer.vocabulary_.get(word,1)].sum() if P[:,vectorizer.v
big_list = heapq.nlargest(30, dic, key=dic.get)
for word in big_list:

    print(f'{word}:  {str(dic[word])} = {P[:,vectorizer.vocabulary_.get(word)].sum()} / {C[:,vectorizer.vocabulary_.get(wor
'''
```

Also we used the *knesset_number* because we realized that the numbers are not evenly distributed between the 2 type, helped the accuracy a lot:
(first is before and the second is after we use it):

```
our vecotr test validation
Our KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.72      0.83      0.77      5872
     plenary       0.80      0.68      0.74      5872

    accuracy                           0.75     11744
   macro avg       0.76      0.75      0.75     11744
weighted avg       0.76      0.75      0.75     11744

our SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.77      0.78      0.78      5872
     plenary       0.78      0.76      0.77      5872

    accuracy                           0.77     11744
   macro avg       0.77      0.77      0.77     11744
weighted avg       0.77      0.77      0.77     11744

our KNN with split:
              precision    recall  f1-score   support

   committee       0.72      0.79      0.76       588
     plenary       0.77      0.70      0.73       587

    accuracy                           0.74      1175
   macro avg       0.75      0.74      0.74      1175
weighted avg       0.75      0.74      0.74      1175

our SVM with split:
              precision    recall  f1-score   support

   committee       0.78      0.82      0.80       588
     plenary       0.81      0.76      0.78       587

    accuracy                           0.79      1175
   macro avg       0.79      0.79      0.79      1175
weighted avg       0.79      0.79      0.79      1175
```

```
our vecotr test validation
Our KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.81      0.89      0.85      5872
     plenary       0.87      0.79      0.83      5872

    accuracy                           0.84     11744
   macro avg       0.84      0.84      0.84     11744
weighted avg       0.84      0.84      0.84     11744

our SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.80      0.88      0.84      5872
     plenary       0.86      0.78      0.82      5872

    accuracy                           0.83     11744
   macro avg       0.83      0.83      0.83     11744
weighted avg       0.83      0.83      0.83     11744

our KNN with split:
              precision    recall  f1-score   support

   committee       0.81      0.93      0.87       588
     plenary       0.92      0.78      0.85       587

    accuracy                           0.86      1175
   macro avg       0.87      0.86      0.86      1175
weighted avg       0.87      0.86      0.86      1175

our SVM with split:
              precision    recall  f1-score   support

   committee       0.81      0.90      0.85       588
     plenary       0.88      0.78      0.83       587

    accuracy                           0.84      1175
   macro avg       0.84      0.84      0.84      1175
weighted avg       0.84      0.84      0.84      1175
```

Also we tried to include the average length to the feature vector but it made the accuracy worst:

```
our vecotr test validation
Our KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.74      0.90      0.81      5872
     plenary       0.87      0.69      0.77      5872

    accuracy                           0.79     11744
   macro avg       0.80      0.79      0.79     11744
weighted avg       0.80      0.79      0.79     11744

our SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.78      0.72      0.75      5872
     plenary       0.74      0.79      0.77      5872

    accuracy                           0.76     11744
   macro avg       0.76      0.76      0.76     11744
weighted avg       0.76      0.76      0.76     11744

our KNN with split:
              precision    recall  f1-score   support

   committee       0.73      0.90      0.81       588
     plenary       0.87      0.66      0.75       587

    accuracy                           0.78      1175
   macro avg       0.80      0.78      0.78      1175
weighted avg       0.80      0.78      0.78      1175

our SVM with split:
              precision    recall  f1-score   support

   committee       0.79      0.70      0.74       588
     plenary       0.73      0.81      0.77       587

    accuracy                           0.75      1175
   macro avg       0.76      0.75      0.75      1175
weighted avg       0.76      0.75      0.75      1175
```

Section 5:

1. $k = 10$ in knn also we used linear kernel for the SVM to classify BoW and rbf kernel for our feature vector.
2. And here are our results:
   For BoW feature vector we get:

```
BoW train validation
KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.80      0.96      0.87      5872
     plenary       0.95      0.77      0.85      5872

    accuracy                           0.86     11744
   macro avg       0.88      0.86      0.86     11744
weighted avg       0.88      0.86      0.86     11744

SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.91      0.93      0.92      5872
     plenary       0.92      0.91      0.92      5872

    accuracy                           0.92     11744
   macro avg       0.92      0.92      0.92     11744
weighted avg       0.92      0.92      0.92     11744

KNN with split:
              precision    recall  f1-score   support

   committee       0.81      0.96      0.88       588
     plenary       0.95      0.78      0.86       587

    accuracy                           0.87      1175
   macro avg       0.88      0.87      0.87      1175
weighted avg       0.88      0.87      0.87      1175

SVM with split:
              precision    recall  f1-score   support

   committee       0.91      0.93      0.92       588
     plenary       0.93      0.91      0.92       587

    accuracy                           0.92      1175
   macro avg       0.92      0.92      0.92      1175
weighted avg       0.92      0.92      0.92      1175
```

And for our feature vector we get:

```
our vecotr test validation
Our KNN with corss validation:
              precision    recall  f1-score   support

   committee       0.81      0.89      0.85      5872
     plenary       0.87      0.79      0.83      5872

    accuracy                           0.84     11744
   macro avg       0.84      0.84      0.84     11744
weighted avg       0.84      0.84      0.84     11744

our SVM with corss validation:
              precision    recall  f1-score   support

   committee       0.80      0.88      0.84      5872
     plenary       0.86      0.78      0.82      5872

    accuracy                           0.83     11744
   macro avg       0.83      0.83      0.83     11744
weighted avg       0.83      0.83      0.83     11744

our KNN with split:
              precision    recall  f1-score   support

   committee       0.81      0.93      0.87       588
     plenary       0.92      0.78      0.85       587

    accuracy                           0.86      1175
   macro avg       0.87      0.86      0.86      1175
weighted avg       0.87      0.86      0.86      1175

our SVM with split:
              precision    recall  f1-score   support

   committee       0.81      0.90      0.85       588
     plenary       0.88      0.78      0.83       587

    accuracy                           0.84      1175
   macro avg       0.84      0.84      0.84      1175
weighted avg       0.84      0.84      0.84      1175
```

Section 6:

We predicted using the SVM that belongs to the BoW because he has the best accuracy.

Section 7 (questions):

1. If the recall is smaller the false-negative is bigger, and if the precision is smaller the false-positive is bigger, we see that SVM on BoW has almost the same recall and precision, which means that it can identify them equally, but KNN of BoW (also for both classifiers of our feature vector) the committee precision is smaller which means it can identify committee with lower accuracy, and that's why the recall is also smaller for plenary (because if it does not identify it as committee then it gives plenary which will increase the false-negative).
2. We see that we have almost the same accuracy and confusion (recall and precision), and that is because we have the same number of test and train data with same distribution over the data, and the difference distribution over the dataset is stable and there is no big differences between batches, because of that cross-validation didn't give a higher accuracy.
3. SVM has better performance for BoW and worst performance for our feature vector, also it requires a lot more time to be trained comparing to KNN, but for prediction it requires less time, and that is because the way it works (KNN needs no training).

KNN has weaker performance for BoW while a better performance for our feature vector comparing to SVM, it takes long time for predication, but less time training comparing to SVM.
If we look at the features above for both of the classifiers we see that features of SVM is better because we are going to train one time and then predict a lot more and it gave better accuracy for BoW than KNN gave for our feature vector.

4.

Big chunk size: First the feature matrix would be smaller which means the run time would be less, and the accuracy get bigger (near perfect and even perfect for 10% test if big enough) and that is because of overfitting, why there is overfitting? Because we would look at a lot less vectors which means that we would have less variation and more close vectors which means we have less diverse data which automatically lead to overfitting.

Small chunk size: if we have small chunk size then we would have bigger feature matrix which is worst for run time, also each and every chunk would have features that doesn't identify the types good enough which make the classification task harder.

5. $Chunk\ size\ =\ 10$ is the best size for our data.
We tried 5 and we have no over fitting, and then we tried 20 but we had an accuracy of 0.99 which were an overfitting (according to the test accuracy and the escalading of the training process), and we tried something in the middle (13) and we got 0.97 (for split SVM) which feels like there is an overfitting, and then we tried 10 which give us an accuracy of 0.96 (in split SVM) which sounds like the best accuracy without overfitting for this data.
Note: after we finished, we returned the size to 5.