Report HW 2 Perceptron Algorithm Machine Learning:

PART 0 (Sentiment Classification Task and Dataset):

1) **Why is each of these steps necessary or helpful for machine learning?**

Ans:

Each of these steps are very important in Machine Learning for sentiment classification for the following:

a) **All the words are lowercased:** This is very important for standardization as it helps in reducing vocabulary size. For example: "Car", "CAR" and "car" will be treated differently, so by lowercasing the words we will be able to help the model treating the same word not differently and will significantly help in reducing the complexity and dimensionality of the data. It will also help in consistency as the model learns a single representation for each word, leading to more accurate and reliable analysis.
b) **Punctuations are split from adjoining words and become separate "words"**: This helps in isolating the words and punctuation marks as separate entities. It will help in more accurate word representation and it also preserves the punctuation values.
c) **Verb contractions and the genitive of nouns are split into their component morphemes, and each morpheme is tagged separately:** From my understanding the words are split because of a process called tokenization, where each word will have meaning which will be considered as a token. It helps the model to clarify the meaning and will make it more explicit and helps in maintaining consistency for the whole dataset.
d) **Single quotes are changed to forward- and backward- quotes and double quotes are changed to double single forward- and backward- quotes:** I feel this is also another form of standardization. It ensures uniformity for the model and ensures all the quoted words is represented consistently and the model will not treat different types of quotation marks as distinct tokens and not adding unnecessary complexity.
e) **Removing Spanish words:** By removing the Spanish words it will make the data more consistent because a model trained for English language will get confused by the introduction of Spanish words and can cause unnecessary complexity as different models have unique ways of expressing sentiment. Hence, it prevents noise and

improves accuracy of the model. We have to remove English words if a model is being trained for Spanish language.

PART 1 (Naive Perceptron Baseline):

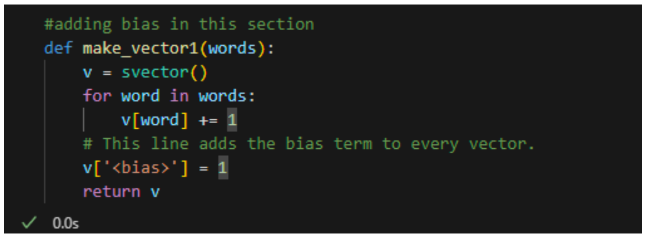
1) **Take a look at svector.py, and briefly explain why it can support addition, subtraction, scalar product, dot product, and negation.**
   Ans:
   The svector.py supports standard vector operations by using python's special overloading methods. From my understanding it is built upon **collections.defaultdict package**, which makes it super memory efficient for all the text data as words (which are features here) that are actually present in a sentence.
   - **Addition:** It is implemented via _add_ and _iadd_ methods. It usually works by iterating through all the words of the sentence in the dataset of each vector and summing the values up.
   - **Subtraction:** It is implemented via _sub_ and _isub_ methods. It usually works by using addition and scalar multiplication (a-b = a+(-1*b)).
   - **Scalar Product:** The _mul_ and _rmul method from my understanding in which they allow multiplication of vectors and numbers, which is the weight of every feature in the vector.
   - **Dot Product:** A custom dot() method is used which efficiently calculates the dot product. This is I think very important as it is used for predictions, as it computes the sum of weighted features.
   - **Negation:** The _neg_ method returns anew svector with all the features and its weights having their signs flipped.

2) **Take a look at train.py, and briefly explain train () and test () functions.**
   Ans:
   Train(trainfile, devfile, epochs=5): In this function we are implementing the perceptron learning algorithm. It initializes an empty weight vector which is **w**. It iterates throughout the entire dataset for a specified number of epochs. So, for each training example, it makes a prediction.
   Going into code:
   - First it initializes the empty svector called model which will serve as the weight vector.
   - It uses outer loop to iterate through the entire training data multiple times for range. Each full pass is one epoch.
   - Inside each epoch it processes each sentence one by one.

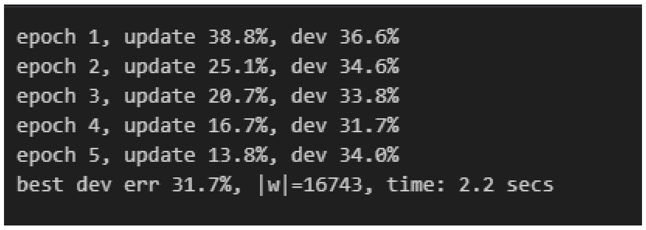
- For each sentence, it creates a feature vector sent using make_vector(words) method.
- The condition if label * (model.dot(sent)) <= 0: is used to check if the model makes an incorrect prediction on the sentence sent.
- If the prediction is incorrect, two things happen:
  a) **updates += 1**: The counter for the number of updates in the epoch is incremented.
  b) **model += label * sent**: This is the perceptron update rule. The model's weights are adjusted by adding the feature vector sent which is scaled by the true label. This nudges closer to the model's decision boundary so that it is more likely to classify these specific sentences correctly in the future.

Test(devfile, model): This function is used for evaluating the trained model's performance on the dataset. It iterates through each sentence, converts it into a feature vector, and then calculates the dot product with the model's weight vector(w). Based on the result whatever sign is present for this dot product it determines the predicted class. The function checks for a misclassification using the condition **label*(model.dot(vector)) <= 0**. If this condition is true then error is counted. Finally, it returns the overall error rate based on total errors / total sentences.

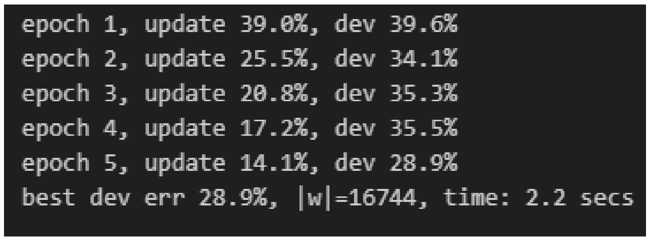
3) **There is one thing missing in my train.py: the bias dimension! Try add it. How did you do it? (Hint:by adding bias or <bias>?) Did it improve error on dev?**
Ans:
I did it using the **v['<bias>'] = 1** in my code. Basically, adding bias to each vector. It definitely improved the error rate earlier without the bias the best dev error was **31.7%.** But after adding bias the best dev error changed to **28.9%**, which means the model accuracy has improved slightly.



```
#adding bias in this section
def make_vector1(words):
    v = svector()
    for word in words:
        v[word] += 1
    # This line adds the bias term to every vector.
    v['<bias>'] = 1
    return v
```
✓ 0.0s



```
epoch 1, update 38.8%, dev 36.6%
epoch 2, update 25.1%, dev 34.6%
epoch 3, update 20.7%, dev 33.8%
epoch 4, update 16.7%, dev 31.7%
epoch 5, update 13.8%, dev 34.0%
best dev err 31.7%, |w|=16743, time: 2.2 secs
```

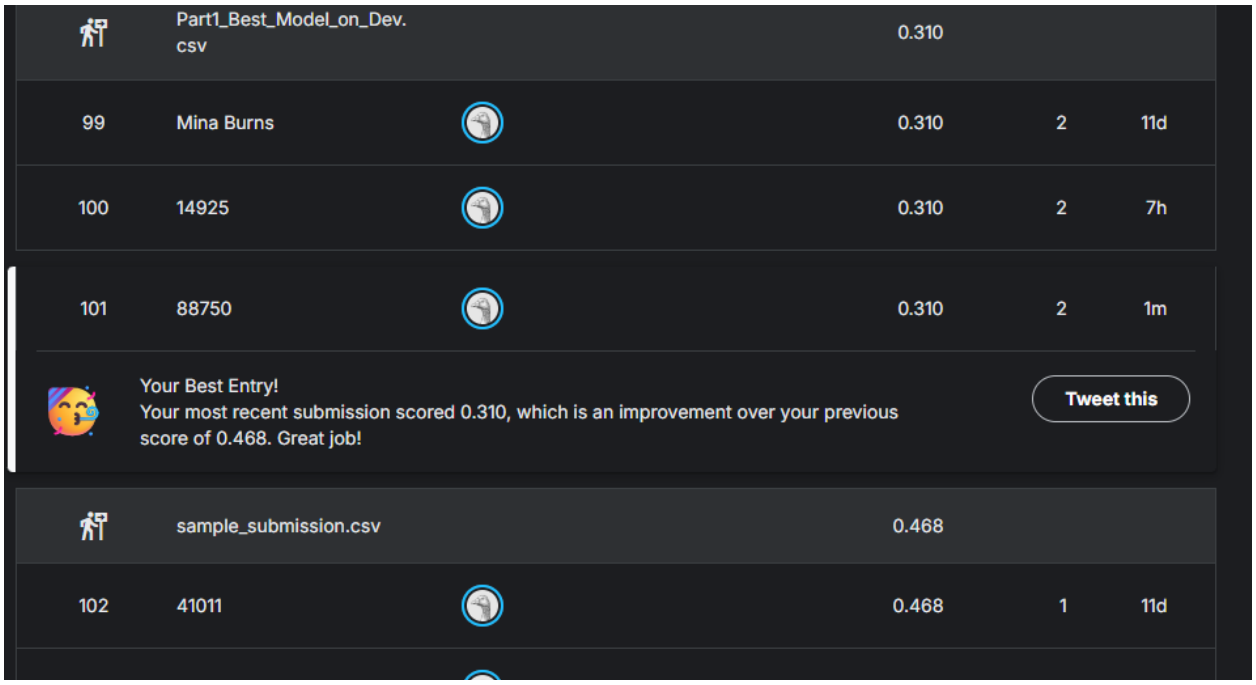


```
epoch 1, update 39.0%, dev 39.6%
epoch 2, update 25.5%, dev 34.1%
epoch 3, update 20.8%, dev 35.3%
epoch 4, update 17.2%, dev 35.5%
epoch 5, update 14.1%, dev 28.9%
best dev err 28.9%, |w|=16744, time: 2.2 secs
```

4) **Using your best model (in terms of dev error rate), predict the semi-blind test data, and submit it to Kaggle (follow instructions from Part 5). What is your error rate and ranking on the public leaderboard? Take a screenshot. Hint: your public error rate should be ~ 31%.**

Ans:

My best model's error rate with "bias added" for the function make_vector1(name that I have given) is **31%.** My ranking is 101 in the public leaderboard.



	Part1_Best_Model_on_Dev.csv	0.310		
99	Mina Burns	0.310	2	11d
100	14925	0.310	2	7h
101	88750	0.310	2	1m

Your Best Entry!
Your most recent submission scored 0.310, which is an improvement over your previous score of 0.468. Great job!

	sample_submission.csv	0.468		
102	41011	0.468	1	11d

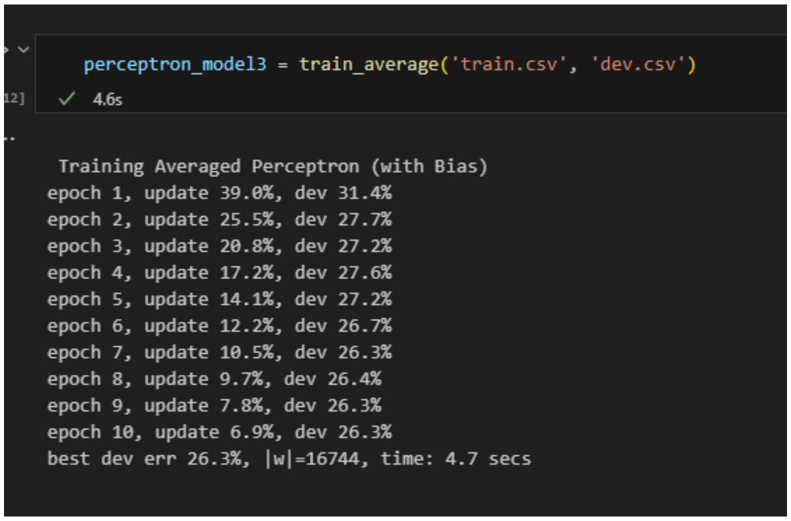
5) **Wait a second, I thought the data set is already balanced (50% positive, 50% negative). I remember the bias being important in highly unbalanced data sets. Why do I still need to add the bias dimension here??**

Ans:

A bias term is still very much necessary even with a balanced dataset which is 50% positive and 50% negative for a very fundamental geometric reason. Without a bias, the separating hyperplane on which the perceptron learns is forced to pass through the origin of the feature space. However, the optimal line or plane that best separates the positive and negative data points may not pass through the origin. The bias term adds a constant feature to every single vector, which acts as an intercept, allowing the decision boundary to be shifted. This gives the model to find the most optimal boundary for the data, leading to better accuracy, regardless of whether the dataset is balanced or not. So, bias is extremely important.

PART 2 (Average Perceptron):

1) **Train for 10 epochs and report the results. Did averaging improve the dev error rate? (Hint: should be around 26%). Did it also make dev error rates more stable?**
   Ans:
   After the implementation of smart averaging in the perceptron model the performance of the model improved. After 10 epochs, it made the best dev error to **26.3%** from **28.9%.** The results are as follows:



```
perceptron_model3 = train_average('train.csv', 'dev.csv')
✓ 4.6s

 Training Averaged Perceptron (with Bias)
epoch 1, update 39.0%, dev 31.4%
epoch 2, update 25.5%, dev 27.7%
epoch 3, update 20.8%, dev 27.2%
epoch 4, update 17.2%, dev 27.6%
epoch 5, update 14.1%, dev 27.2%
epoch 6, update 12.2%, dev 26.7%
epoch 7, update 10.5%, dev 26.3%
epoch 8, update 9.7%, dev 26.4%
epoch 9, update 7.8%, dev 26.3%
epoch 10, update 6.9%, dev 26.3%
best dev err 26.3%, |w|=16744, time: 4.7 secs
```

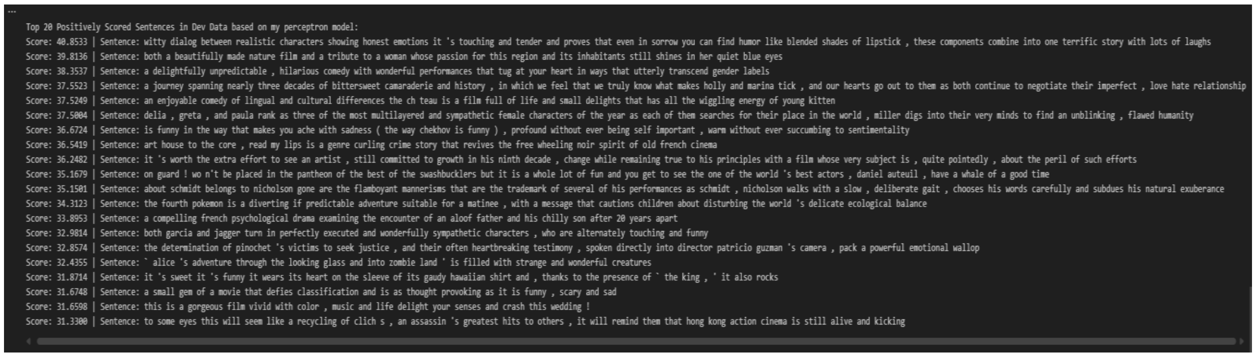
   The dev error has also stabilized quite a bit at **~26%**. The epoch-by-epoch error rates for the averaged perceptron showed a consistent downward trend with less fluctuations when compared to the standard perceptron with bias.

2) **Did smart averaging slow down training?**
   Ans:
   The smart averaging did slow the training process from **2.3 seconds** to **4.7 seconds.** I feel the slight increase is expected because the smart averaging algorithm requires one additional vector update (in my code: **cached_model += c * label * sent**) for every single misclassification, adding it additional computation. Anyways, it is helping in accuracy so it is fine.

3) **What are the top 20 most positive and top 20 most negative features? Do they make sense?**

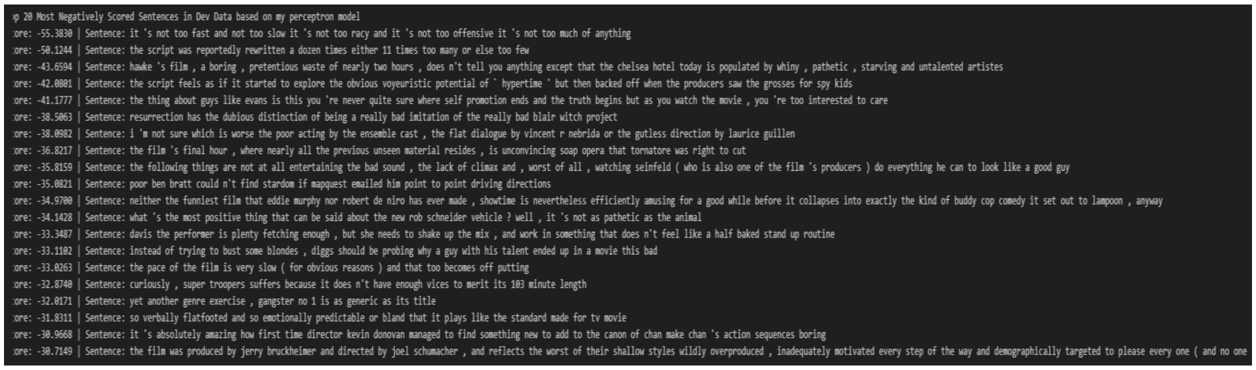
Ans:

The top 20 positive features according to my smart averaged perceptron model is:



```
...
   Top 20 Positively Scored Sentences in Dev Data based on my perceptron model:
   Score: 40.8533 | Sentence: witty dialog between realistic characters showing honest emotions it 's touching and tender and proves that even in sorrow you can find humor like blended shades of lipstick , these components combine into one terrific story with lots of laughs
   Score: 39.8136 | Sentence: both a beautifully made nature film and a tribute to a woman whose passion for this region and its inhabitants still shines in her quiet blue eyes
   Score: 38.3537 | Sentence: a delightfully unpredictable , hilarious comedy with wonderful performances that tug at your heart in ways that utterly transcend gender labels
   Score: 37.5523 | Sentence: a journey spanning nearly three decades of bittersweet camaraderie and history , in which we feel that we truly know what makes holly and marina tick , and our hearts go out to them as both continue to negotiate their imperfect , love hate relationship
   Score: 37.5249 | Sentence: an enjoyable comedy of lingual and cultural differences the ch teau is a film full of life and small delights that has all the wiggling energy of young kitten
   Score: 37.5004 | Sentence: delia , greta , and paula rank as three of the most multilayered and sympathetic female characters of the year as each of them searches for their place in the world , miller digs into their very minds to find an unblinking , flawed humanity
   Score: 36.6724 | Sentence: is funny in the way that makes you ache with sadness ( the way chekhov is funny ) , profound without ever being self important , warm without ever succumbing to sentimentality
   Score: 36.5419 | Sentence: art house to the core , read my lips is a genre curling crime story that revives the free wheeling noir spirit of old french cinema
   Score: 36.2482 | Sentence: it 's worth the extra effort to see an artist , still committed to growth in his ninth decade , change while remaining true to his principles with a film whose very subject is , quite pointedly , about the peril of such efforts
   Score: 35.1679 | Sentence: on guard ! wo n't be placed in the pantheon of the best of the swashbucklers but it is a whole lot of fun and you get to see the one of the world 's best actors , daniel auteuil , have a whale of a good time
   Score: 35.1501 | Sentence: about schmidt belongs to nicholson gone are the flamboyant mannerisms that are the trademark of several of his performances as schmidt , nicholson walks with a slow , deliberate gait , chooses his words carefully and subdues his natural exuberance
   Score: 34.3123 | Sentence: the fourth pokemon is a diverting if predictable adventure suitable for a matinee , with a message that cautions children about disturbing the world 's delicate ecological balance
   Score: 33.8953 | Sentence: a compelling french psychological drama examining the encounter of an aloof father and his chilly son after 20 years apart
   Score: 32.9814 | Sentence: both garcia and jagger turn in perfectly executed and wonderfully sympathetic characters , who are alternately touching and funny
   Score: 32.8574 | Sentence: the determination of pinochet 's victims to seek justice , and their often heartbreaking testimony , spoken directly into director patricio guzman 's camera , pack a powerful emotional wallop
   Score: 32.4355 | Sentence: ` alice 's adventure through the looking glass and into zombie land ' is filled with strange and wonderful creatures
   Score: 31.8714 | Sentence: it 's sweet it 's funny it wears its heart on the sleeve of its gaudy hawaiian shirt and , thanks to the presence of ` the king , ' it also rocks
   Score: 31.6748 | Sentence: a small gem of a movie that defies classification and is as thought provoking as it is funny , scary and sad
   Score: 31.6598 | Sentence: this is a gorgeous film vivid with color , music and life delight your senses and crash this wedding !
   Score: 31.3300 | Sentence: to some eyes this will seem like a recycling of clich s , an assassin 's greatest hits to others , it will remind them that hong kong action cinema is still alive and kicking
```

The top 20 negative features are:



```
p 20 Most Negatively Scored Sentences in Dev Data based on my perceptron model
ore: -55.3830 | Sentence: it 's not too fast and not too slow it 's not too racy and it 's not too offensive it 's not too much of anything
ore: -50.1244 | Sentence: the script was reportedly rewritten a dozen times either 11 times too many or else too few
ore: -43.6594 | Sentence: hawke 's film , a boring , pretentious waste of nearly two hours , does n't tell you anything except that the chelsea hotel today is populated by whiny , pathetic , starving and untalented artistes
ore: -42.0801 | Sentence: the script feels as if it started to explore the obvious voyeuristic potential of ` hypertime ' but then backed off when the producers saw the grosses for spy kids
ore: -41.1777 | Sentence: the thing about guys like evans is this you 're never quite sure where self promotion ends and the truth begins but as you watch the movie , you 're too interested to care
ore: -38.5063 | Sentence: resurrection has the dubious distinction of being a really bad imitation of the really bad blair witch project
ore: -38.0982 | Sentence: i 'm not sure which is worse the poor acting by the ensemble cast , the flat dialogue by vincent r nebrida or the gutless direction by laurice guillen
ore: -36.8217 | Sentence: the film 's final hour , where nearly all the previous unseen material resides , is unconvincing soap opera that tornatore was right to cut
ore: -35.8159 | Sentence: the following things are not at all entertaining the bad sound , the lack of climax and , worst of all , watching seinfeld ( who is also one of the film 's producers ) do everything he can to look like a good guy
ore: -35.0821 | Sentence: poor ben bratt could n't find stardom if mapquest emailed him point to point driving directions
ore: -34.9700 | Sentence: neither the funniest film that eddie murphy nor robert de niro has ever made , showtime is nevertheless efficiently amusing for a good while before it collapses into exactly the kind of buddy cop comedy it set out to lampoon , anyway
ore: -34.1428 | Sentence: what 's the most positive thing that can be said about the new rob schneider vehicle ? well , it 's not as pathetic as the animal
ore: -33.3487 | Sentence: davis the performer is plenty fetching enough , but she needs to shake up the mix , and work in something that does n't feel like a half baked stand up routine
ore: -33.1102 | Sentence: instead of trying to bust some blondes , diggs should be probing why a guy with his talent ended up in a movie this bad
ore: -33.0263 | Sentence: the pace of the film is very slow ( for obvious reasons ) and that too becomes off putting
ore: -32.8740 | Sentence: curiously , super troopers suffers because it does n't have enough vices to merit its 103 minute length
ore: -32.0171 | Sentence: yet another genre exercise , gangster no 1 is as generic as its title
ore: -31.8311 | Sentence: so verbally flatfooted and so emotionally predictable or bland that it plays like the standard made for tv movie
ore: -30.9668 | Sentence: it 's absolutely amazing how first time director kevin donovan managed to find something new to add to the canon of chan make chan 's action sequences boring
ore: -30.7149 | Sentence: the film was produced by jerry bruckheimer and directed by joel schumacher , and reflects the worst of their shallow styles wildly overproduced , inadequately motivated every step of the way and demographically targeted to please every one ( and no one
```

4) **Show 5 negative examples in dev where your model most strongly believes to be positive. Show 5 positive examples in dev where your model most strongly believes to be negative. What observations do you get?**

Ans:

The top 5 negative examples in dev where my model most strongly believes to be positive (I have manually done the analysis based on a csv file I made of top60 most positive sentences and most negative sentences based on score, and I willing be attaching this in the submission file):

- 26.0377,bravo reveals the true intent of her film by carefully selecting interview subjects who will construct a portrait of castro so predominantly charitable it can only be seen as propaganda
- 27.0627,"mr wollter and ms seldhal give strong and convincing performances , but neither reaches into the deepest recesses of the character to unearth the quaking essence of passion , grief and fear"

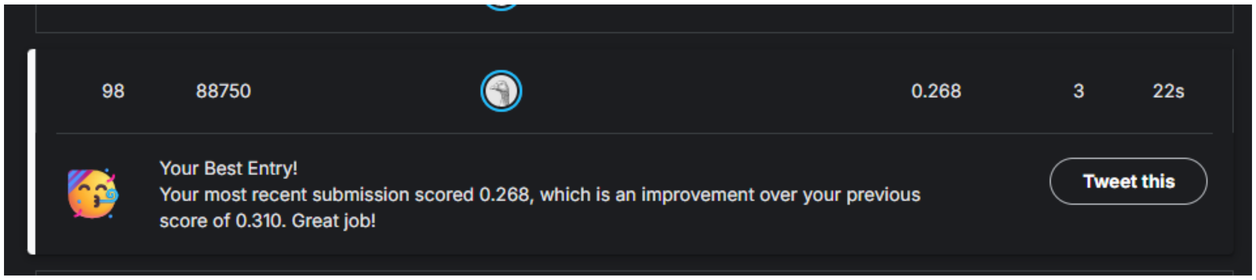
- 27.8475,"when compared to the usual , more somber festival entries , davis ' highly personal brand of romantic comedy is a tart , smart breath of fresh air that stands out from the pack even if the picture itself is somewhat problematic"
- 30.6303,"` in this poor remake of such a well loved classic , parker exposes the limitations of his skill and the basic flaws in his vision '"
- 31.3300,"to some eyes this will seem like a recycling of clich s , an assassin 's greatest hits to others , it will remind them that hong kong action cinema is still alive and kicking"

The top 5 positive examples in dev where your model most strongly believes to be negative:

- -41.1777,"the thing about guys like evans is this you 're never quite sure where self promotion ends and the truth begins but as you watch the movie , you 're too interested to care"
- -34.9700,"neither the funniest film that eddie murphy nor robert de niro has ever made , showtime is nevertheless efficiently amusing for a good while before it collapses into exactly the kind of buddy cop comedy it set out to lampoon , anyway"
-
- -26.9345,"enough is not a bad movie , just mediocre the performances are so overstated , the effect comes off as self parody"
- -24.8673,"even before it builds up to its insanely staged ballroom scene , in which 3000 actors appear in full regalia , it 's waltzed itself into the art film pantheon"

5) Again, using your new best model (in terms of dev error rate), predict the semi-blind test data, and submit it to Kaggle. What is your new error rate and ranking on the public leaderboard? Take a screenshot. Hint: your public error rate should improve to ~ 27%.

Ans:



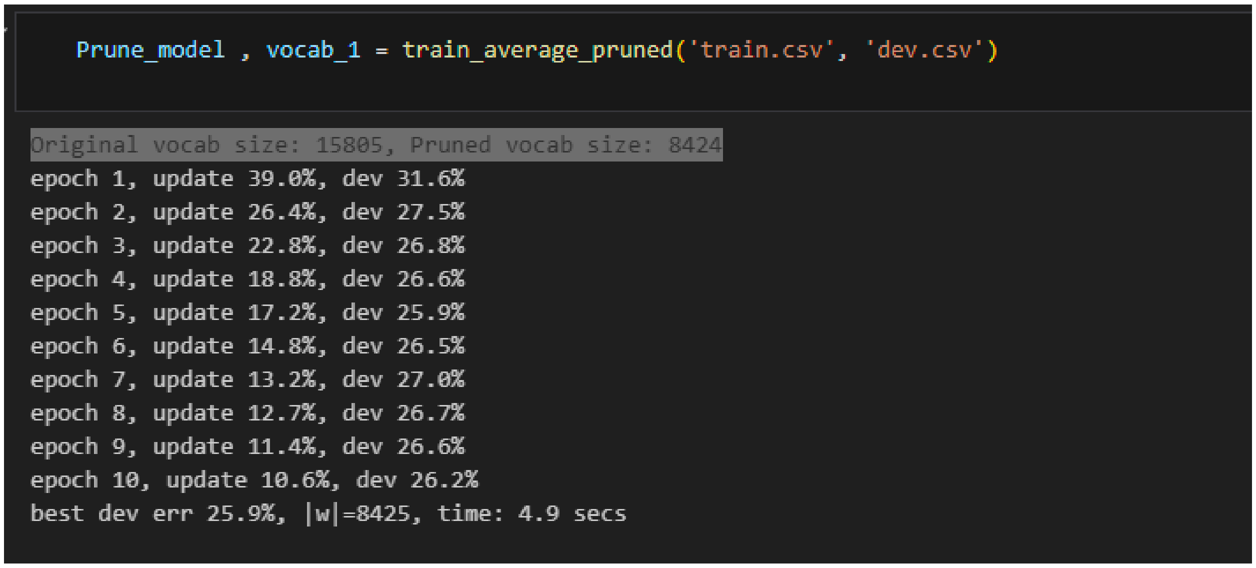
98 88750 0.268 3 22s

Your Best Entry!
Your most recent submission scored 0.268, which is an improvement over your previous score of 0.310. Great job!

The new error rate is **26.8%** on the test file my ranking is 98 which has improved from previous submission.

PART 3 (Pruning the Vocabulary):

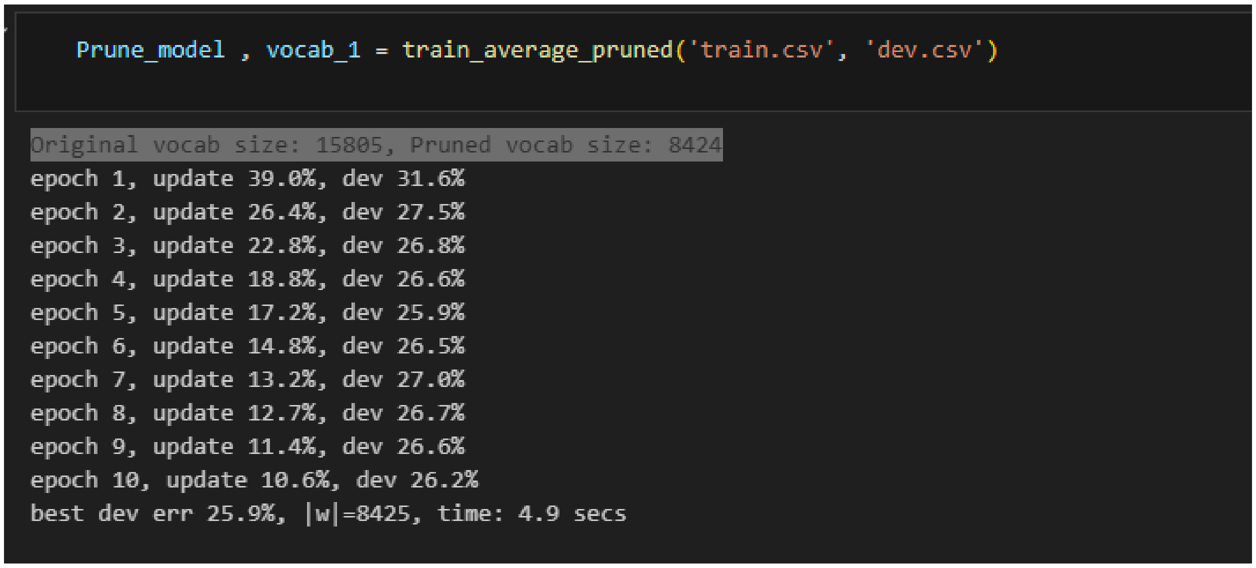
1) **Try neglecting one-count words in the training set during training. Did it improve the dev error rate? (Hint: should help a little bit, error rate lower than 26%).**
   Ans:
   Yes, it improved my model performance. My error_rate is **25.9%** which is slightly better than earlier.



```
Prune_model , vocab_1 = train_average_pruned('train.csv', 'dev.csv')

Original vocab size: 15805, Pruned vocab size: 8424
epoch 1, update 39.0%, dev 31.6%
epoch 2, update 26.4%, dev 27.5%
epoch 3, update 22.8%, dev 26.8%
epoch 4, update 18.8%, dev 26.6%
epoch 5, update 17.2%, dev 25.9%
epoch 6, update 14.8%, dev 26.5%
epoch 7, update 13.2%, dev 27.0%
epoch 8, update 12.7%, dev 26.7%
epoch 9, update 11.4%, dev 26.6%
epoch 10, update 10.6%, dev 26.2%
best dev err 25.9%, |w|=8425, time: 4.9 secs
```

2) **Did your model size shrink, and by how much? (Hint: should almost halve). Does this shrinking help prevent overfitting?**
   Ans:

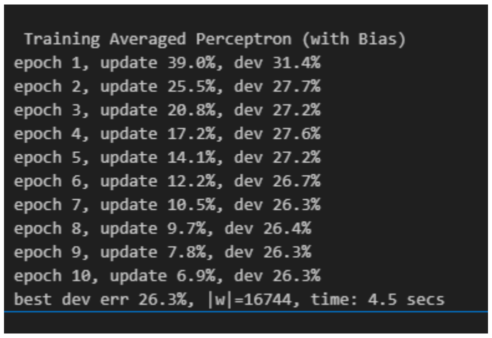


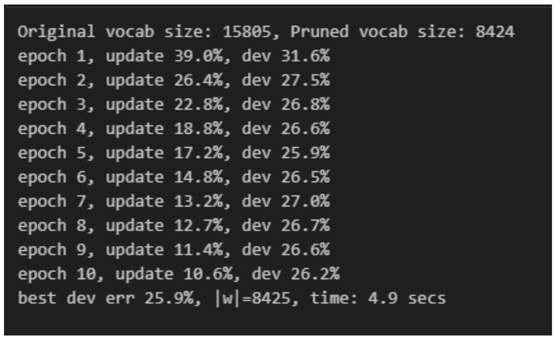
```
Prune_model , vocab_1 = train_average_pruned('train.csv', 'dev.csv')

Original vocab size: 15805, Pruned vocab size: 8424
epoch 1, update 39.0%, dev 31.6%
epoch 2, update 26.4%, dev 27.5%
epoch 3, update 22.8%, dev 26.8%
epoch 4, update 18.8%, dev 26.6%
epoch 5, update 17.2%, dev 25.9%
epoch 6, update 14.8%, dev 26.5%
epoch 7, update 13.2%, dev 27.0%
epoch 8, update 12.7%, dev 26.7%
epoch 9, update 11.4%, dev 26.6%
epoch 10, update 10.6%, dev 26.2%
best dev err 25.9%, |w|=8425, time: 4.9 secs
```

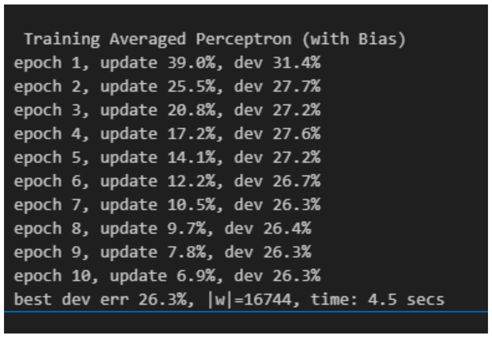
Original vocab dictionary size was **15805** and after pruning the dictionary size shrank to **8424.** This shrinking helps prevent overfitting as overfitting means that the model learns the training(/dev) data too well, including its noise and idiosyncrasies, and as a result, performs poorly on new, unseen data. So, by pruning the data and restricting it to 1 it helped the model to increase its accuracy slightly as there is less data to train on and not training two times for the same word.

3) **Did update % change? Does the change make sense?**
Ans:
Yes, the update percentage did change, and the change makes sense. The "update percentage" from my understanding explains how often the model made a mistake on the training data and had to be corrected. If we compare the best dev errors of PART2 and PART3 the change is huge because the model made more mistakes when it gave the best dev error percentage in the pruned model's update_percentage is 17.2% and for average_perceptron at the best dev error the update_percentage was 6.9%.

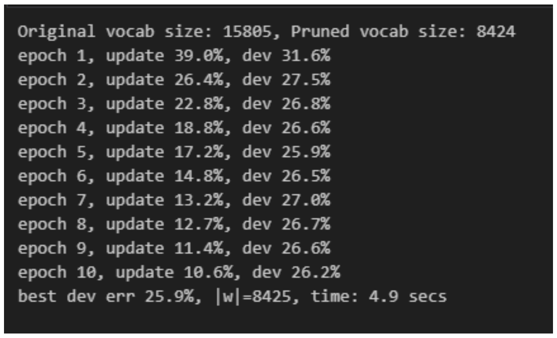

```
 Training Averaged Perceptron (with Bias)
epoch 1, update 39.0%, dev 31.4%
epoch 2, update 25.5%, dev 27.7%
epoch 3, update 20.8%, dev 27.2%
epoch 4, update 17.2%, dev 27.6%
epoch 5, update 14.1%, dev 27.2%
epoch 6, update 12.2%, dev 26.7%
epoch 7, update 10.5%, dev 26.3%
epoch 8, update 9.7%, dev 26.4%
epoch 9, update 7.8%, dev 26.3%
epoch 10, update 6.9%, dev 26.3%
best dev err 26.3%, |w|=16744, time: 4.5 secs
```


```
Original vocab size: 15805, Pruned vocab size: 8424
epoch 1, update 39.0%, dev 31.6%
epoch 2, update 26.4%, dev 27.5%
epoch 3, update 22.8%, dev 26.8%
epoch 4, update 18.8%, dev 26.6%
epoch 5, update 17.2%, dev 25.9%
epoch 6, update 14.8%, dev 26.5%
epoch 7, update 13.2%, dev 27.0%
epoch 8, update 12.7%, dev 26.7%
epoch 9, update 11.4%, dev 26.6%
epoch 10, update 10.6%, dev 26.2%
best dev err 25.9%, |w|=8425, time: 4.9 secs
```

4) **Did the training speed change?**
Ans:
There isn't much change in the training speed if **train_average_pruned is** compared to **train_average(perceptron_model3)** only a difference of **0.4 seconds** but the training speed decreased ever so slightly.



```
 Training Averaged Perceptron (with Bias)
epoch 1, update 39.0%, dev 31.4%
epoch 2, update 25.5%, dev 27.7%
epoch 3, update 20.8%, dev 27.2%
epoch 4, update 17.2%, dev 27.6%
epoch 5, update 14.1%, dev 27.2%
epoch 6, update 12.2%, dev 26.7%
epoch 7, update 10.5%, dev 26.3%
epoch 8, update 9.7%, dev 26.4%
epoch 9, update 7.8%, dev 26.3%
epoch 10, update 6.9%, dev 26.3%
best dev err 26.3%, |w|=16744, time: 4.5 secs
```

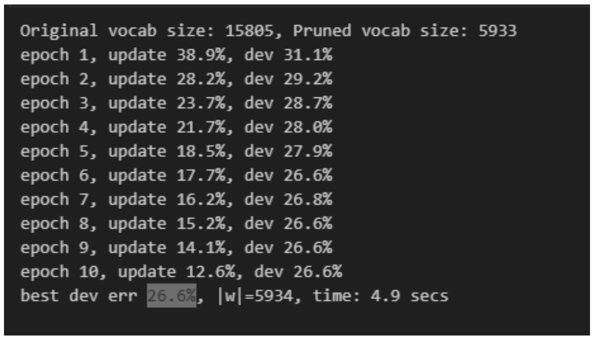


```
Original vocab size: 15805, Pruned vocab size: 8424
epoch 1, update 39.0%, dev 31.6%
epoch 2, update 26.4%, dev 27.5%
epoch 3, update 22.8%, dev 26.8%
epoch 4, update 18.8%, dev 26.6%
epoch 5, update 17.2%, dev 25.9%
epoch 6, update 14.8%, dev 26.5%
epoch 7, update 13.2%, dev 27.0%
epoch 8, update 12.7%, dev 26.7%
epoch 9, update 11.4%, dev 26.6%
epoch 10, update 10.6%, dev 26.2%
best dev err 25.9%, |w|=8425, time: 4.9 secs
```

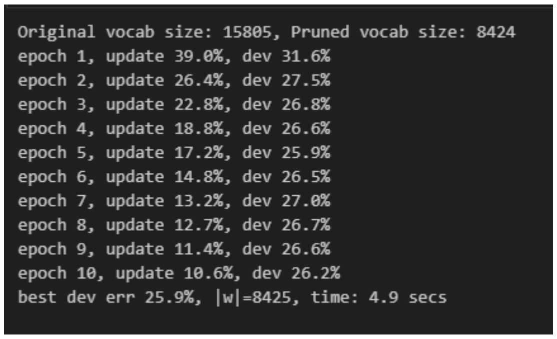
5) **What about further pruning two-count words (words that appear twice in the training set)? Did it further improve dev error rate?**

Ans:

The model's performance further decreased because it was underfitting as the error percentage increased to **26.6%.**



```
Original vocab size: 15805, Pruned vocab size: 5933
epoch 1, update 38.9%, dev 31.1%
epoch 2, update 28.2%, dev 29.2%
epoch 3, update 23.7%, dev 28.7%
epoch 4, update 21.7%, dev 28.0%
epoch 5, update 18.5%, dev 27.9%
epoch 6, update 17.7%, dev 26.6%
epoch 7, update 16.2%, dev 26.8%
epoch 8, update 15.2%, dev 26.6%
epoch 9, update 14.1%, dev 26.6%
epoch 10, update 12.6%, dev 26.6%
best dev err 26.6%, |w|=5934, time: 4.9 secs
```

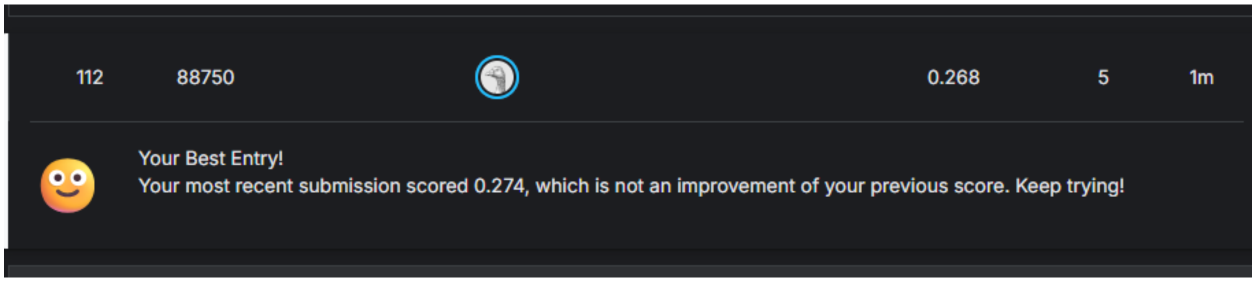


```
Original vocab size: 15805, Pruned vocab size: 8424
epoch 1, update 39.0%, dev 31.6%
epoch 2, update 26.4%, dev 27.5%
epoch 3, update 22.8%, dev 26.8%
epoch 4, update 18.8%, dev 26.6%
epoch 5, update 17.2%, dev 25.9%
epoch 6, update 14.8%, dev 26.5%
epoch 7, update 13.2%, dev 27.0%
epoch 8, update 12.7%, dev 26.7%
epoch 9, update 11.4%, dev 26.6%
epoch 10, update 10.6%, dev 26.2%
best dev err 25.9%, |w|=8425, time: 4.9 secs
```

As you can see in the screenshots earlier it was **25.9%.** So, I feel we should not prune the data further.

6) **Using your current best model (in terms of dev error rate) from this part, predict the semi-blind test data, and submit it to Kaggle. What is your new error rate and ranking on the public leaderboard? Take a screenshot. Hint: your public error rate should still be ~ 27% and it is not necessarily lower than Part 2.**

Ans:

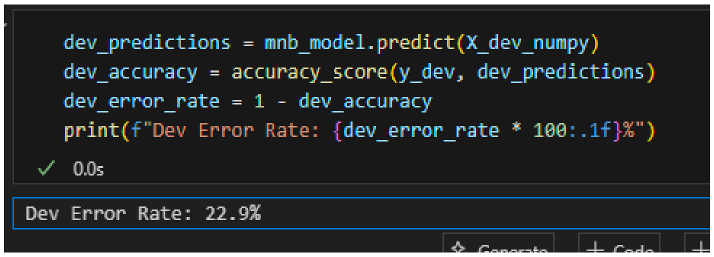


112 88750 0.268 5 1m

Your Best Entry!
Your most recent submission scored 0.274, which is not an improvement of your previous score. Keep trying!

Yes, my error rate is still **27.4%.**

PART 4 (Try some other learning algorithms with sklearn):

1) **Which algorithm did you try? What adaptations did you make to your code to make it work with that algorithm? What specific setting (e.g., vocabulary pruning) did you use?**
   Ans:
   The algorithm I tried was Multinomial Naive Bayes (MNB). It gave me pretty good results as the dev error was **22.9%**.
   

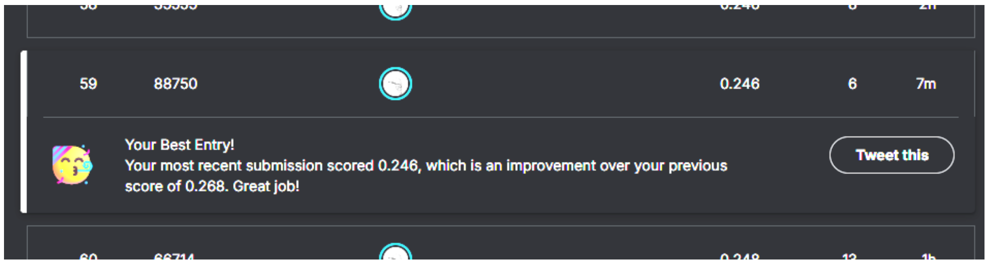
```
dev_predictions = mnb_model.predict(X_dev_numpy)
dev_accuracy = accuracy_score(y_dev, dev_predictions)
dev_error_rate = 1 - dev_accuracy
print(f"Dev Error Rate: {dev_error_rate * 100:.1f}%")
✓ 0.0s
Dev Error Rate: 22.9%
```

   I did some adaptions like I changed the manual iterative training loops which I used earlier from PART 1 to PART 3 and used scikit.learn workflow like **.fit()** which I'm much more familiar with.
   Adaptions:
   - I first used my own make_vectorp function to create a list of svector dictionaries for all the three datasets (train,dev and test datasets).
   - Then I used feature_extraction.DictVectorize method from the sklearn library it was used to convert the standerized dictionaries into spare numerical matrix and then eventually converted to Numpy matrix and these numpy_matrix was used for training the model.

2) **What's the dev error rate(s) and running time?**
   Ans:
   Dev error rate is **22.9%.** The model trained in **0.19** seconds because I used the scikit learn libraries .fit method which made the model train super-fast.

3) **Submit your best prediction to Kaggle. What was your public score and rank? For example, our TA Zetian got ~ 24% (quite a bit better than 27%).**
   Ans:
   I submitted my best predictions model which was the Multinomial Naive Bayes model to Kaggle.
   - **Public Score (Error Rate):** 0.246 (or 24.6%)

- **Rank:** 59



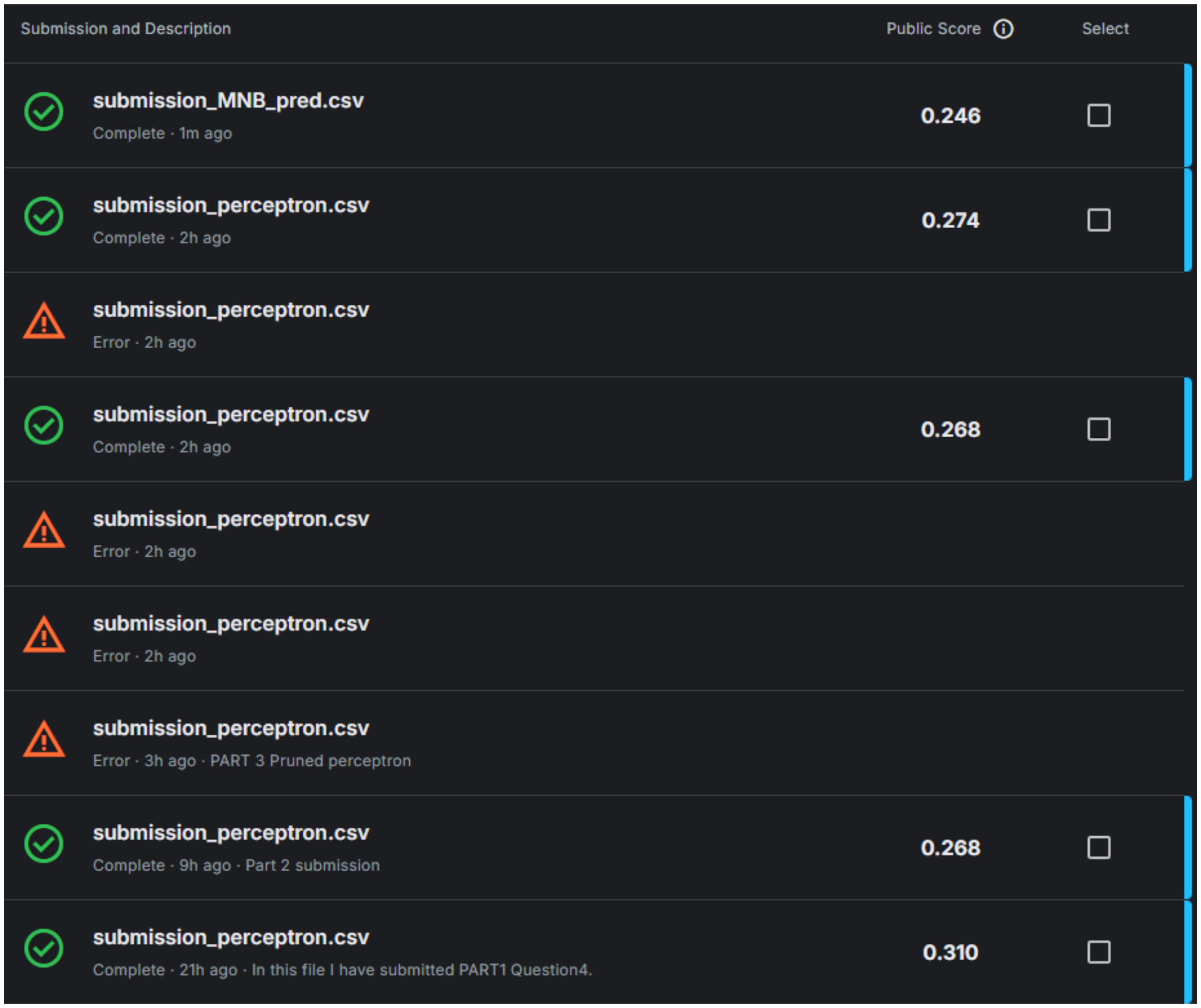
59 88750 0.246 6 7m

Your Best Entry!
Your most recent submission scored 0.246, which is an improvement over your previous score of 0.268. Great job!

4) **What did you learn in terms of the comparison between averaged perceptron and these other (presumably more popular and well-known) learning algorithms?**
   Ans:
   - I feel Perceptron algorithm tries to find one single boundary for each feature and especially Multinomial naves bayes (MNB) tries to find probability of each feature.
   - The MNB algorithm is a lot faster and accurate as well compared to perceptron algorithm I would say. Especially dataset like these like "bag of words" and maybe classification models are better.

PART 5 (Deployment):

1) **What's the dev error rate(s) and how did you achieve it?**
   Ans:



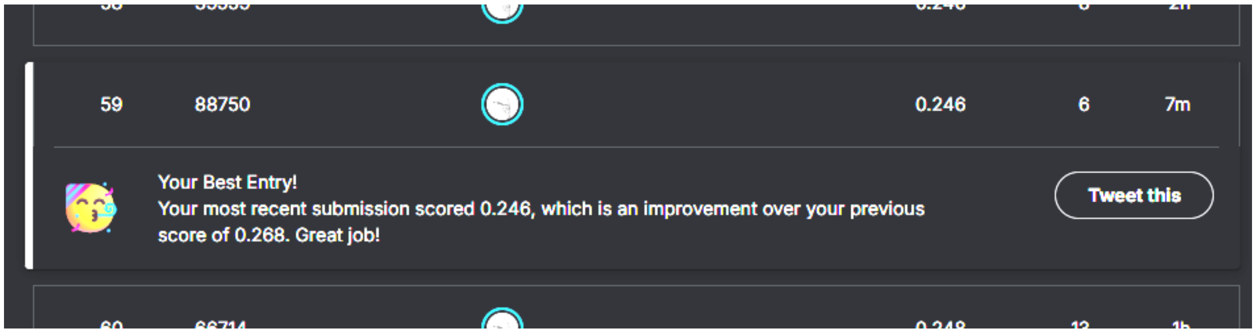
Submission and Description	Public Score	Select
✓ **submission_MNB_pred.csv** Complete · 1m ago	0.246	☐
✓ **submission_perceptron.csv** Complete · 2h ago	0.274	☐
⚠ **submission_perceptron.csv** Error · 2h ago		
✓ **submission_perceptron.csv** Complete · 2h ago	0.268	☐
⚠ **submission_perceptron.csv** Error · 2h ago		
⚠ **submission_perceptron.csv** Error · 2h ago		
⚠ **submission_perceptron.csv** Error · 3h ago · PART 3 Pruned perceptron		
✓ **submission_perceptron.csv** Complete · 9h ago · Part 2 submission	0.268	☐
✓ **submission_perceptron.csv** Complete · 21h ago · In this file I have submitted PART1 Question4.	0.310	☐

I have only tried one model because personally, it took a lot of time and I struggled to understand perceptron Algorithm. I discussed with a lot of students and they suggested me to use Multinomial Naïve bayes because it gave them the best dev error rate. But I will surely try other Machine learning algorithms in the future.

2) **Submit your best prediction to Kaggle. What was your overall best public score and final rank?**

Ans:

My best model got a public test error rate of **24.6%.** My final rank is 59.



59	88750	0.246	6	7m

Your Best Entry!
Your most recent submission scored 0.246, which is an improvement over your previous score of 0.268. Great job!

PART 6 (Debriefing):

1) **Approximately how many hours did you spend on this assignment?**
   Ans:
   It took me around 75 hrs to complete the assignment.
2) **Would you rate it as easy, moderate, or difficult?**
   Ans:
   I would rate this assignment as difficult especially the math behind the perceptron algorithm. I would say I struggled a lot and hours of thinking and implementation I was super exhausted.
3) **Did you work on it mostly alone, or mostly with other people?**
   Ans:
   For the code I would say I worked all alone. But for math I asked couple of my friends who helped me understand the perceptron algorithm
4) **How deeply do you feel you understand the material it covers (0%–100%)?**
   Ans:
   I would still say I understood about 75% of the material because of math.
5) **Any other comments?**
   Ans:
   I would say, this assignment really pushed me and was lot harder than expected. Maybe, I need to work even harder than what I'm currently doing I guess….