Data Science Applications

Classification Assignment

Students: Group 7

1. Kareem Atif Mohamed Bakli.

2. Gehad Hisham Hassan Abdelghany.

3. Kareem Khaled.

4. Mostafa Nofal.

Overview:

The goal of this project is to classify books by its title, compare the different models we used; analyze each one and come out the best model which is most efficient in this problem.

The dataset:

1. We scrapped 5 books with different authors and with the same genre (Philosophy genre):
   - The Art of War,
   - Beyond Good and Evil,
   - The Psychology and Pedagogy of Anger
   - The Republic
   - Meditations
2. Created 200 partitions of each book text, each partition contains 100 words.
3. We come out with a DataFrame contains:

- Partitions columns,
- The title of the book column,
- Author column.

Preprocess the data:

- Converted the text to lower case.
- Removed any special characters.
- Used RegexpTokenizer to tokenize the text.
- Created our stop words list and removed from our text.
- Remove single char, and chars with size 2.

Visualization:

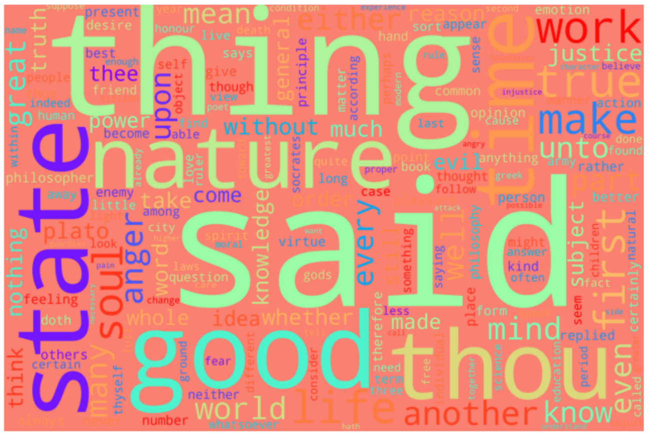
Word Cloud

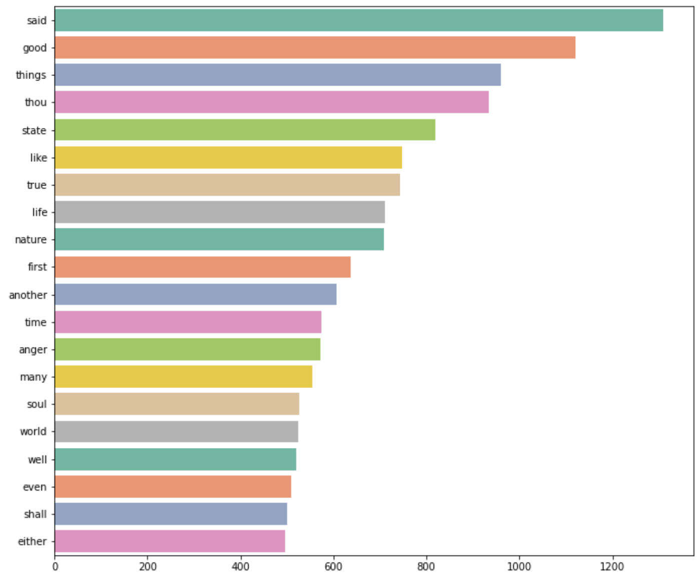
Data visualizations (like charts, graphs, infographics, and more) give a valuable way to visualize, and statistical important information, but what if your raw data is text-based?

Word cloud generators can help simplify this process.

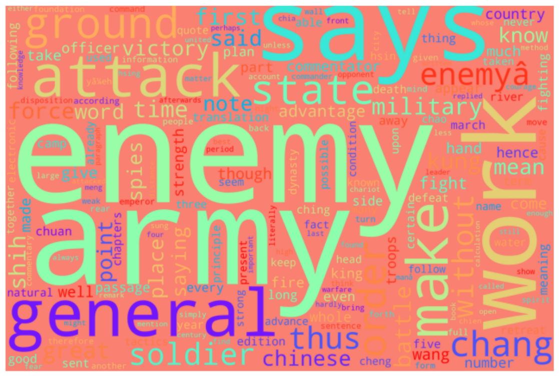
A word cloud is a collection, or cluster, of words depicted in different sizes. The bigger and bolder the word appears, the more often it's mentioned within a given text and the more important it is.

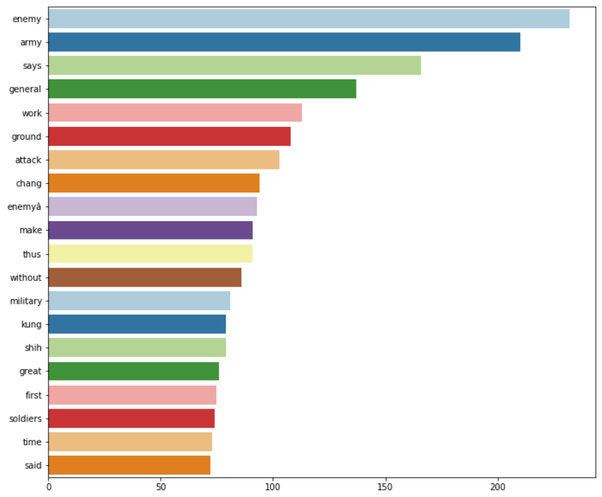
- All Books



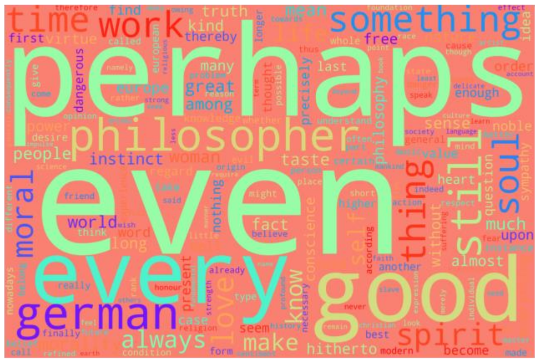


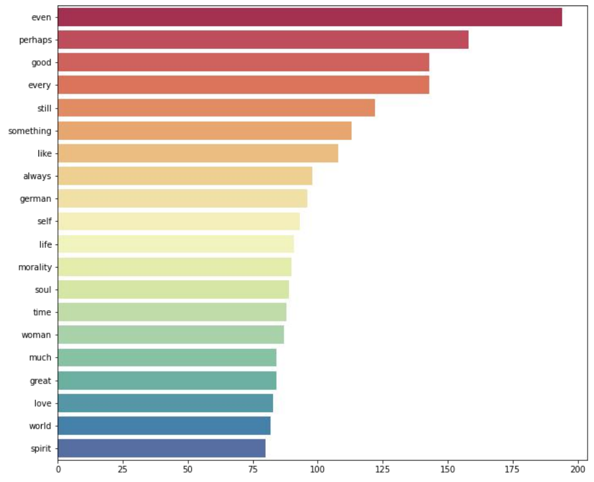
- Book1



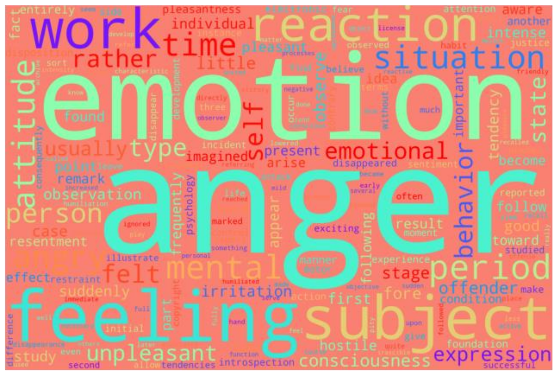


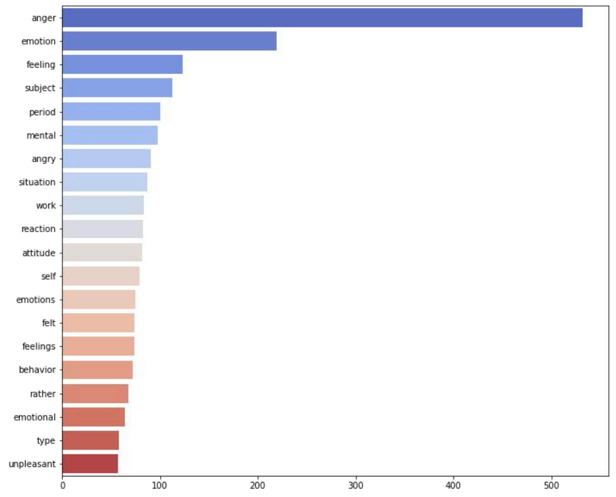
- Book2



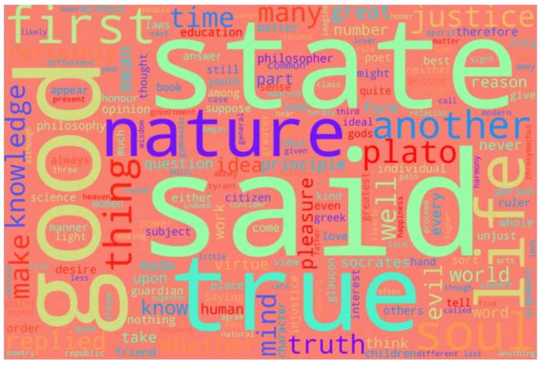


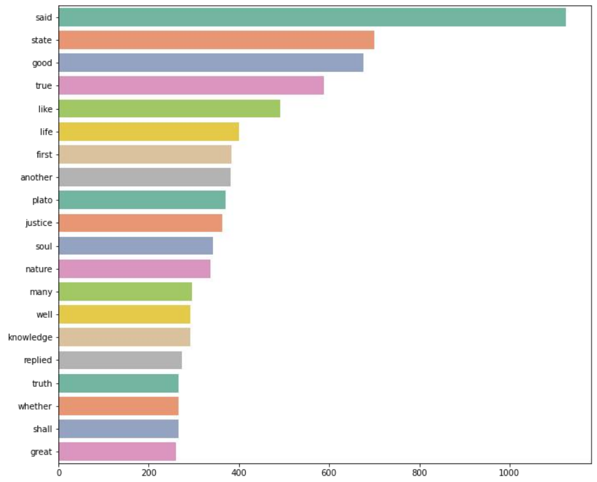
- Book3



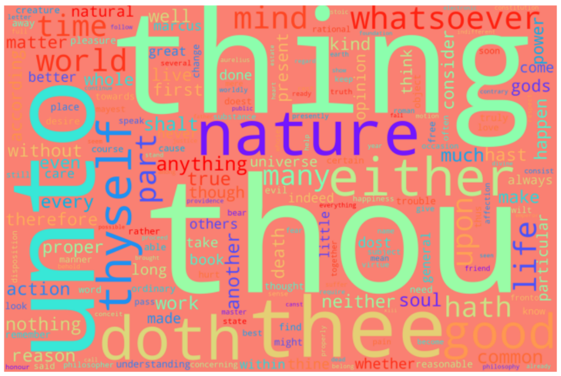


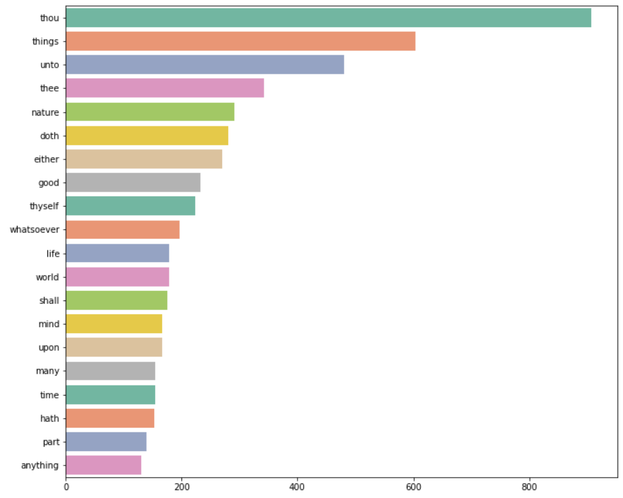
- Book4



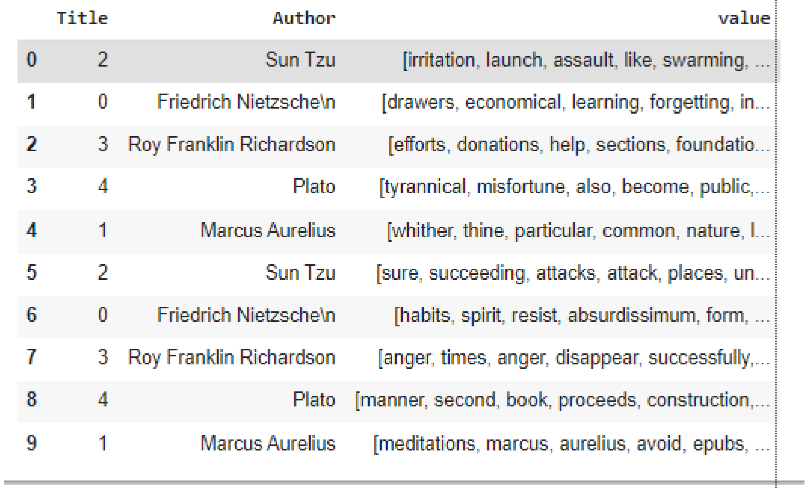


- Book5





- Label Encode "Title" Column.

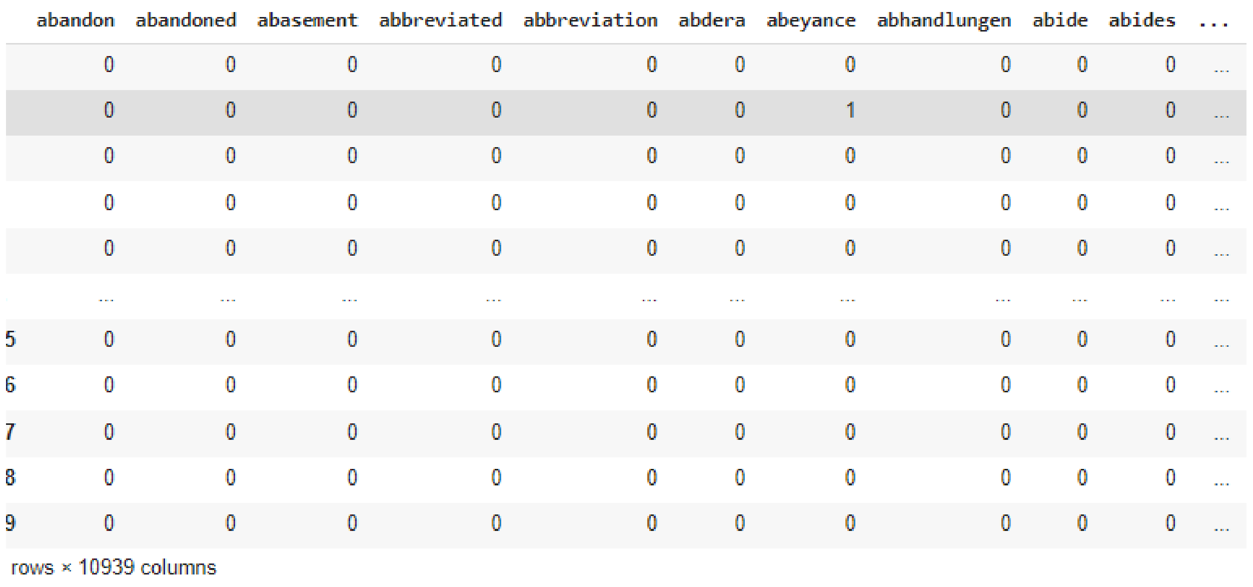


	Title	Author	value
0	2	Sun Tzu	[irritation, launch, assault, like, swarming, ...
1	0	Friedrich Nietzsche\n	[drawers, economical, learning, forgetting, in...
2	3	Roy Franklin Richardson	[efforts, donations, help, sections, foundatio...
3	4	Plato	[tyrannical, misfortune, also, become, public,...
4	1	Marcus Aurelius	[whither, thine, particular, common, nature, l...
5	2	Sun Tzu	[sure, succeeding, attacks, attack, places, un...
6	0	Friedrich Nietzsche\n	[habits, spirit, resist, absurdissimum, form, ...
7	3	Roy Franklin Richardson	[anger, times, anger, disappear, successfully,...
8	4	Plato	[manner, second, book, proceeds, construction,...
9	1	Marcus Aurelius	[meditations, marcus, aurelius, avoid, epubs, ...

Modeling:

1. Split the dataset into train, test dataset, with size: 80% for train and 20 for test.
2. Used BOW, TF-IDF and n-gram with each one of it.

- BOW:
  - Used BOW to transfer the train, test features.



	abandon	abandoned	abasement	abbreviated	abbreviation	abdera	abeyance	abhandlungen	abide	abides	...
	0	0	0	0	0	0	0	0	0	0	...
	0	0	0	0	0	0	1	0	0	0	...
	0	0	0	0	0	0	0	0	0	0	...
	0	0	0	0	0	0	0	0	0	0	...
	0	0	0	0	0	0	0	0	0	0	...
	...	...	...	...	...	...	...	...	...	...	...
5	0	0	0	0	0	0	0	0	0	0	...
6	0	0	0	0	0	0	0	0	0	0	...
7	0	0	0	0	0	0	0	0	0	0	...
8	0	0	0	0	0	0	0	0	0	0	...
9	0	0	0	0	0	0	0	0	0	0	...

rows × 10939 columns

- **Use Different algorithms and evaluate them**

  1. SVM,
  2. Decision Tree,
  3. k-Nearest Neighbor.

- **K-Fold cross validation:**

We used K-Fold cross validation with 10 folds and compared the accuracy of each model, then chose the champion model, and the transformation and the classification algorithms, and chose it also by Mean, Bias, variance, standard deviation.

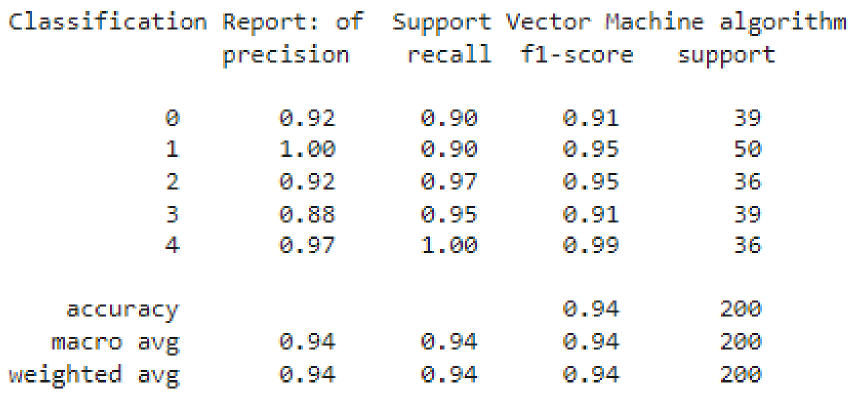
1. SVM

Accuracy

- Support Vector Machine Accuracy for Training: 0.9825
- Support Vector Machine Accuracy for Testing: 0.94

Cross Validation:

Classification Report



```
Classification Report: of  Support Vector Machine algorithm
              precision    recall  f1-score   support

           0       0.92      0.90      0.91        39
           1       1.00      0.90      0.95        50
           2       0.92      0.97      0.95        36
           3       0.88      0.95      0.91        39
           4       0.97      1.00      0.99        36

    accuracy                           0.94       200
   macro avg       0.94      0.94      0.94       200
weighted avg       0.94      0.94      0.94       200
```

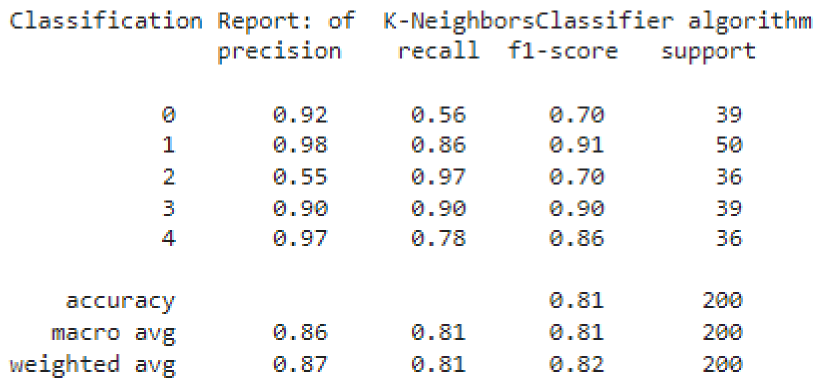
2. k-Nearest Neighbor

Accuracy

- K-NeighborsClassifier Accuracy for Training: 0.90875
- K-NeighborsClassifier Accuracy for Testing: 0.815

Cross Validation:

Classification Report



```
Classification Report: of  K-NeighborsClassifier algorithm
              precision    recall  f1-score   support

           0       0.92      0.56      0.70        39
           1       0.98      0.86      0.91        50
           2       0.55      0.97      0.70        36
           3       0.90      0.90      0.90        39
           4       0.97      0.78      0.86        36

    accuracy                           0.81       200
   macro avg       0.86      0.81      0.81       200
weighted avg       0.87      0.81      0.82       200
```

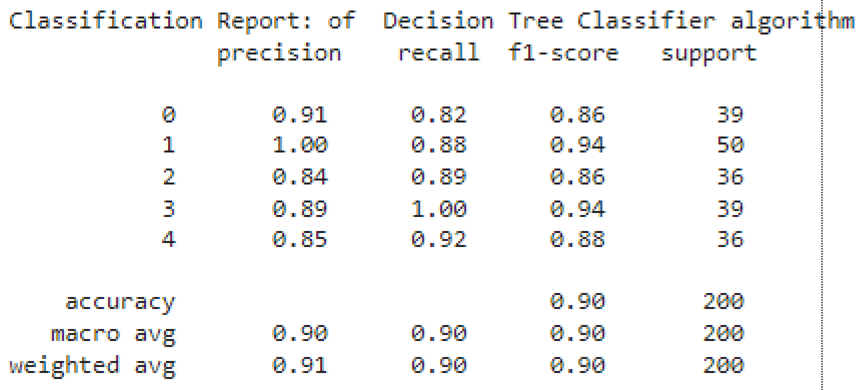
3. Decision Tree

Accuracy

- Decision Tree Classifier Accuracy for Training: 1.0
- Decision Tree Classifier Accuracy for Testing: 0.86

Cross Validation:

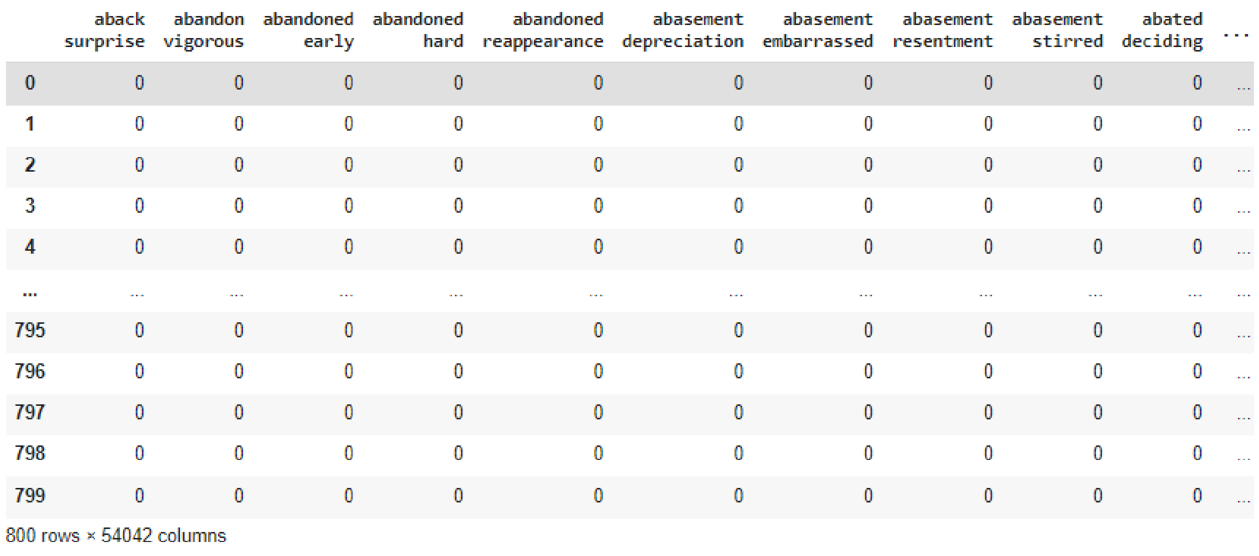
Classification Report



```
Classification Report: of  Decision Tree Classifier algorithm
              precision    recall  f1-score   support

           0       0.91      0.82      0.86        39
           1       1.00      0.88      0.94        50
           2       0.84      0.89      0.86        36
           3       0.89      1.00      0.94        39
           4       0.85      0.92      0.88        36

    accuracy                           0.90       200
   macro avg       0.90      0.90      0.90       200
weighted avg       0.91      0.90      0.90       200
```

- N-gram (n =2)

  Used 2-gram to transfer the train, test features.



	aback surprise	abandon vigorous	abandoned early	abandoned hard	abandoned reappearance	abasement depreciation	abasement embarrassed	abasement resentment	abasement stirred	abated deciding	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
795	0	0	0	0	0	0	0	0	0	0	...
796	0	0	0	0	0	0	0	0	0	0	...
797	0	0	0	0	0	0	0	0	0	0	...
798	0	0	0	0	0	0	0	0	0	0	...
799	0	0	0	0	0	0	0	0	0	0	...

800 rows × 54042 columns

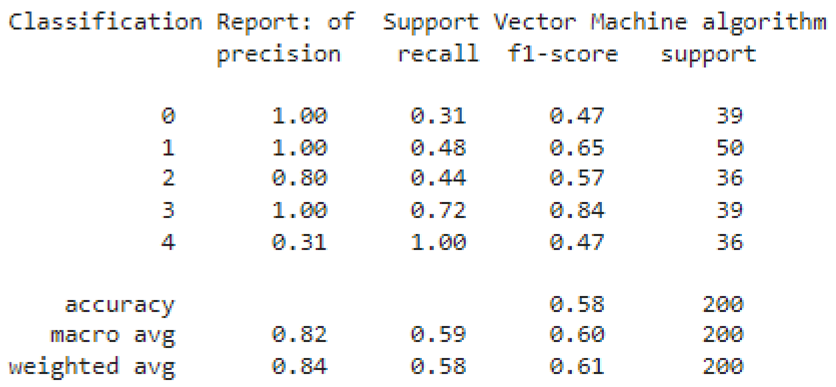
- **Use Different algorithms and evaluate them**

1. SVM,

- `Support Vector Machine Accuracy for Training: 0.98875`
- `Support Vector Machine Accuracy for Testing: 0.58`

Cross Validation:

Classification Report



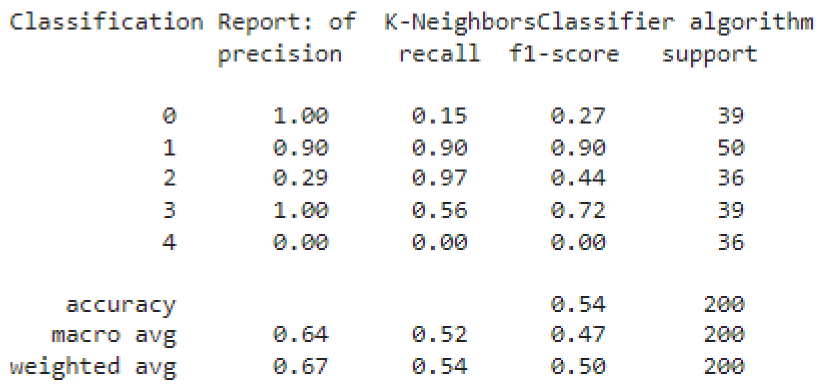
```
Classification Report: of  Support Vector Machine algorithm
              precision    recall  f1-score   support

           0       1.00      0.31      0.47        39
           1       1.00      0.48      0.65        50
           2       0.80      0.44      0.57        36
           3       1.00      0.72      0.84        39
           4       0.31      1.00      0.47        36

    accuracy                           0.58       200
   macro avg       0.82      0.59      0.60       200
weighted avg       0.84      0.58      0.61       200
```

2. k-Nearest Neighbor.

```
- K-NeighborsClassifier Accuracy for Training: 0.6425
- K-NeighborsClassifier Accuracy for Testing: 0.54
```

Cross Validation:

Classification Report



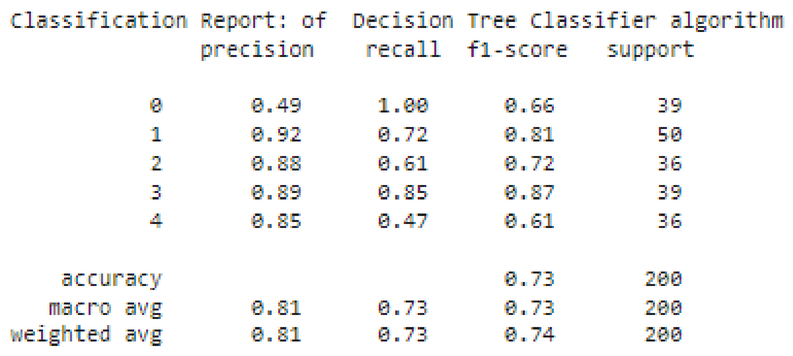
```
Classification Report: of  K-NeighborsClassifier algorithm
              precision    recall  f1-score   support

           0       1.00      0.15      0.27        39
           1       0.90      0.90      0.90        50
           2       0.29      0.97      0.44        36
           3       1.00      0.56      0.72        39
           4       0.00      0.00      0.00        36

    accuracy                           0.54       200
   macro avg       0.64      0.52      0.47       200
weighted avg       0.67      0.54      0.50       200
```

3. **Decision Tree,**

- `Decision Tree Classifier Accuracy for Training: 1.0`
- `Decision Tree Classifier Accuracy for Testing: 0.735`

Cross Validation:

Classification Report



```
Classification Report: of  Decision Tree Classifier algorithm
              precision    recall  f1-score   support

           0       0.49      1.00      0.66        39
           1       0.92      0.72      0.81        50
           2       0.88      0.61      0.72        36
           3       0.89      0.85      0.87        39
           4       0.85      0.47      0.61        36

    accuracy                           0.73       200
   macro avg       0.81      0.73      0.73       200
weighted avg       0.81      0.73      0.74       200
```

- **TF-IDF**

Used TF-IDF to transfer the train, test features.

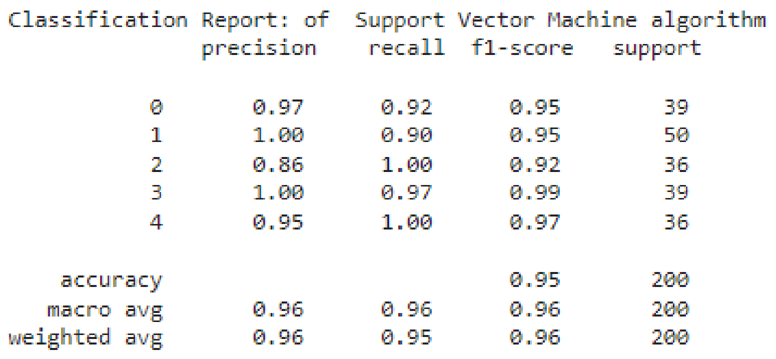
- **Use Different algorithms and evaluate them**

1. **SVM,**

```
- Support Vector Machine Accuracy for Training: 0.9775
- Support Vector Machine Accuracy for Testing: 0.955
```

Cross Validation:

Classification Report



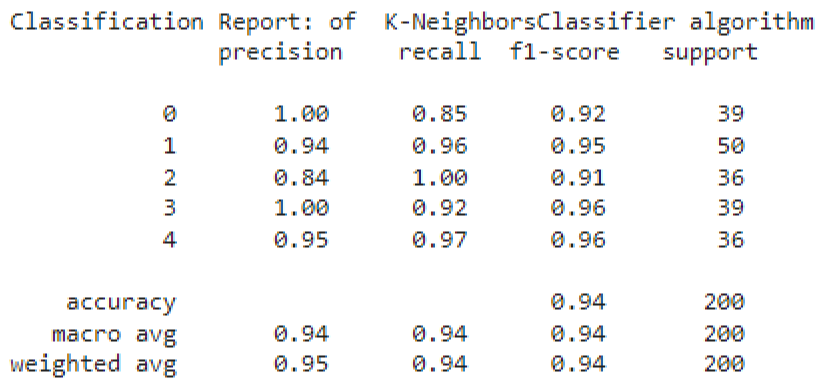
```
Classification Report: of  Support Vector Machine algorithm
              precision    recall  f1-score   support

           0       0.97      0.92      0.95        39
           1       1.00      0.90      0.95        50
           2       0.86      1.00      0.92        36
           3       1.00      0.97      0.99        39
           4       0.95      1.00      0.97        36

    accuracy                           0.95       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.96      0.95      0.96       200
```

2. Decision Tree,

- Decision Tree Classifier Accuracy for Training: 1.0
- Decision Tree Classifier Accuracy for Testing: 0.895

Cross Validation:

Classification Report



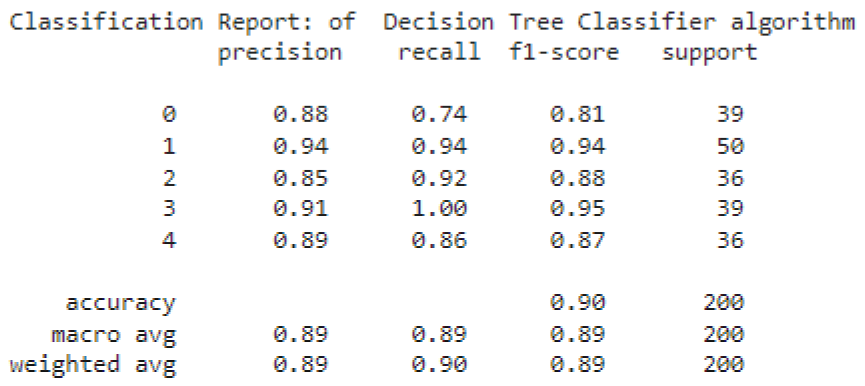
```
Classification Report: of  K-NeighborsClassifier algorithm
              precision    recall  f1-score   support

           0       1.00      0.85      0.92        39
           1       0.94      0.96      0.95        50
           2       0.84      1.00      0.91        36
           3       1.00      0.92      0.96        39
           4       0.95      0.97      0.96        36

    accuracy                           0.94       200
   macro avg       0.94      0.94      0.94       200
weighted avg       0.95      0.94      0.94       200
```

3. K-3 Nearest Neighbor.

- K-NeighborsClassifier Accuracy for Training: 0.93
- K-NeighborsClassifier Accuracy for Testing: 0.94

Cross Validation:

Classification Report



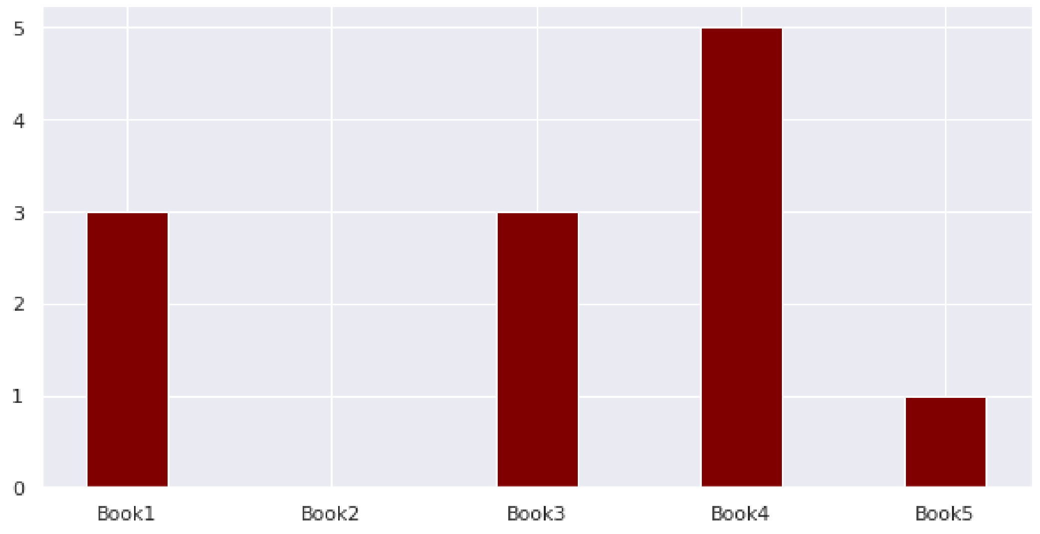
```
Classification Report: of  Decision Tree Classifier algorithm
              precision    recall  f1-score   support

           0       0.88      0.74      0.81        39
           1       0.94      0.94      0.94        50
           2       0.85      0.92      0.88        36
           3       0.91      1.00      0.95        39
           4       0.89      0.86      0.87        36

    accuracy                           0.90       200
   macro avg       0.89      0.89      0.89       200
weighted avg       0.89      0.90      0.89       200
```

Choosing Champion Model

- In Bag of words we choose SVM according to Accuracy results and Cross - validation results also calculate mean and variance and choose Algorithm with high accuracy, mean and low variance of cross-validation results.
- In n-gram we choose Decision Tree Classifier according to Accuracy and Cross-validation results also calculate mean and variance and choose Algorithm with high accuracy, mean and low variance of Cross-validation results.
- In TF-IDF we choose SVM according to Accuracy and Cross-validation results also calculate mean and variance and choose Algorithm with high accuracy, mean and low variance of validation results.
- From above results we choose BOW approach with Support Vector Machine Classifier as Champion model according to results.
- We used K-Fold cross validation with 10 folds and compared the accuracy of each model, then chose the champion model, and the transformation and the classification algorithms, and chose it also by Mean, Bias, variance, standard deviation.

Error Analysis

- We calculated the number of wrong prediction labels by comparing the predicted output with the actual output of each class.
- And plotted the number of wrong predictions of each class.



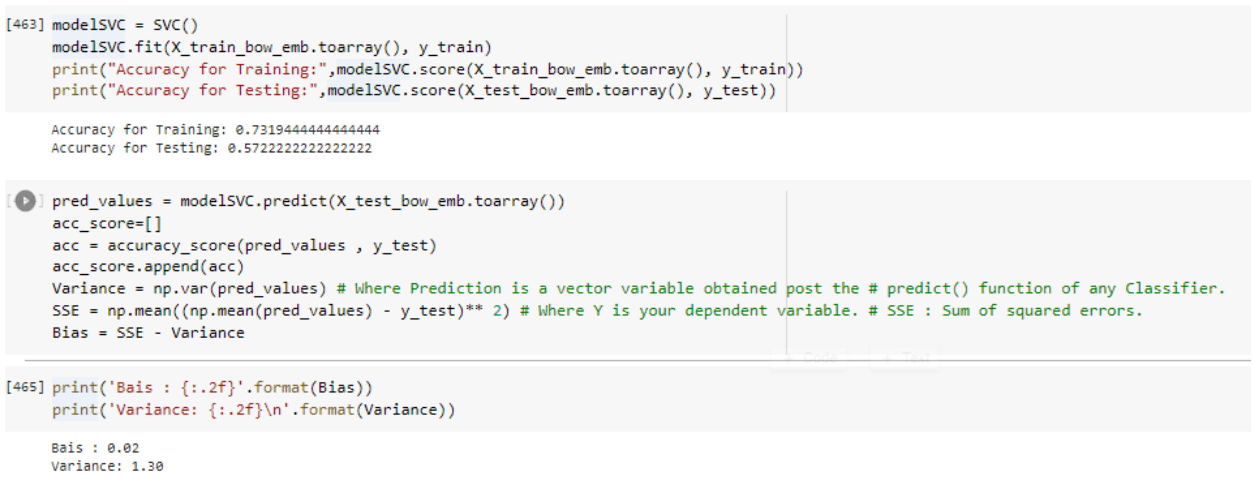
Play with features and decrease accuracy:

After decided the Champion book, we tried to add some books that does not belong to the same category of the original book, without preprocess its text, and also add some special characters to the original dataset.

Accuracy:

```
Accuracy for Training: 0.7319444444444444
Accuracy for Testing: 0.5722222222222222
```

And after that, we calculated the bias and variance



```
[463] modelSVC = SVC()
      modelSVC.fit(X_train_bow_emb.toarray(), y_train)
      print("Accuracy for Training:",modelSVC.score(X_train_bow_emb.toarray(), y_train))
      print("Accuracy for Testing:",modelSVC.score(X_test_bow_emb.toarray(), y_test))

      Accuracy for Training: 0.7319444444444444
      Accuracy for Testing: 0.5722222222222222
```

```
      pred_values = modelSVC.predict(X_test_bow_emb.toarray())
      acc_score=[]
      acc = accuracy_score(pred_values , y_test)
      acc_score.append(acc)
      Variance = np.var(pred_values) # Where Prediction is a vector variable obtained post the # predict() function of any Classifier.
      SSE = np.mean((np.mean(pred_values) - y_test)** 2) # Where Y is your dependent variable. # SSE : Sum of squared errors.
      Bias = SSE - Variance
```

```
[465] print('Bais : {:.2f}'.format(Bias))
      print('Variance: {:.2f}\n'.format(Variance))

      Bais : 0.02
      Variance: 1.30
```

Augmentation by Reinforcement Learning:

- TextRL is Text generation Library with reinforcement learning using hugging face's transformer.

we clone to the files in this github link which include library that we used to generate tests

```
[ ] !git clone https://github.com/voidful/TextRL.git
```

```
[ ] cd '/content/TextRL'
```

```
[ ] pip install -e .
```

Example

Controllable generation via RL to let Elon Musk speak ill of DOGE

- before: i think dogecoin is a great idea.
- after: i think dogecoin is a great idea, but I think it is a little overused.

setup reward function for environment

- predicted(list[str]): will be the list of predicted token
- finish(bool): it met the end of sentence or not

predicted(list[str]): will be the list of predicted token finish(bool): it met the end of sentence or not.

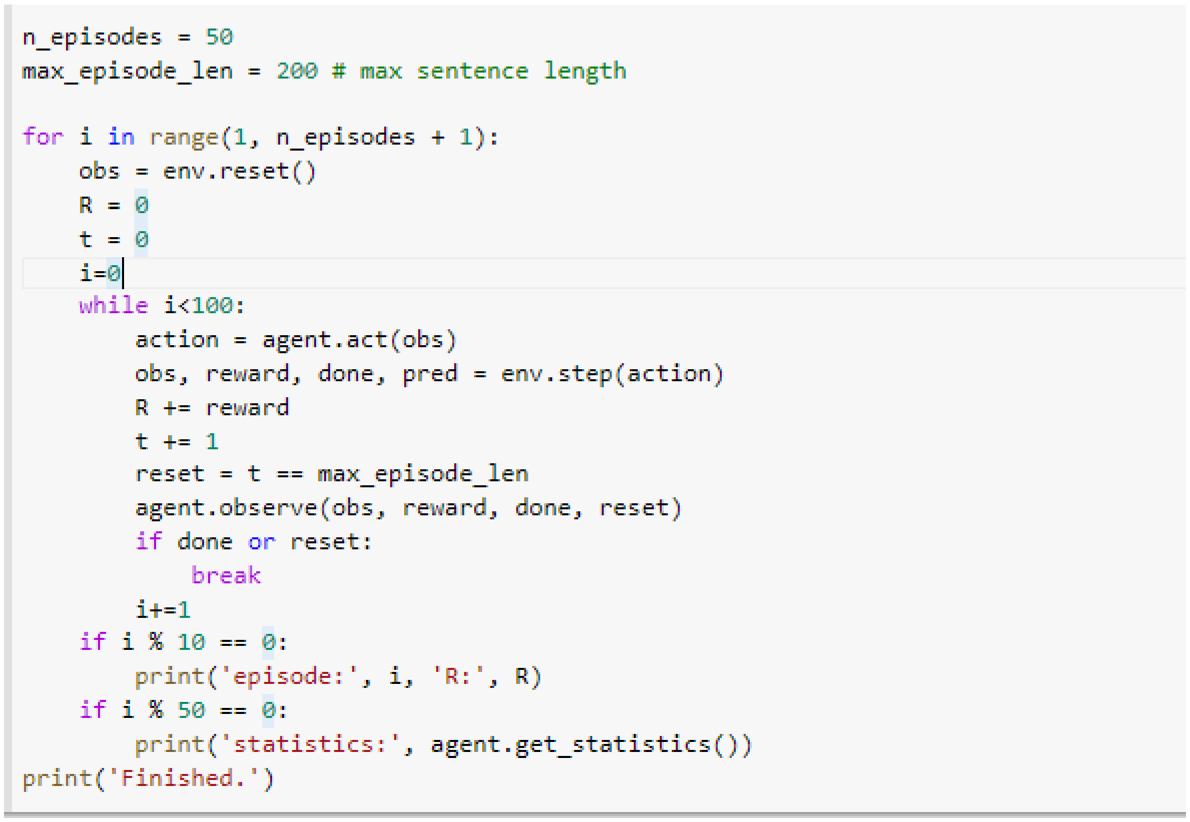
```
class MyRLEnv(TextRLEnv):
    def get_reward(self, input_text, predicted_list, finish):
      print(predicted_list)
      if "dinner" in predicted_list:
        reward = -1
      else:
        reward = 1
      return reward
```

```
env = MyRLEnv(model, tokenizer, observation_input=observaton_list)
actor = TextRLActor(env,model,tokenizer)
agent = actor.agent_ppo(update_interval=10, minibatch_size=2000, epochs=10)
```

Train

prepare for training

- observation_input should be a list of all possible input string for model



```python
n_episodes = 50
max_episode_len = 200 # max sentence length

for i in range(1, n_episodes + 1):
    obs = env.reset()
    R = 0
    t = 0
    i=0
    while i<100:
        action = agent.act(obs)
        obs, reward, done, pred = env.step(action)
        R += reward
        t += 1
        reset = t == max_episode_len
        agent.observe(obs, reward, done, reset)
        if done or reset:
            break
        i+=1
    if i % 10 == 0:
        print('episode:', i, 'R:', R)
    if i % 50 == 0:
        print('statistics:', agent.get_statistics())
print('Finished.')
```

For prediction

```python
#agent.load("somewhere/best") # loading the best model
actor.predict("dinner")
```