# Homework 2

**Yu-Chieh Kuo B07611039**[†]

[†]Department of Information Management, National Taiwan University

## Usage

```
cd b07611039
cp -R /PA2-data .
python -V # Python 3.7.12 in my environment
pip install nltk
pip install sklearn
pip install matplotlib
# Requirements: nltk, sklearn, matplotlib
python3 pa2_NB.py
python3 pa2_svmLinear.py
python3 pa2_svmRbf.py
```

## Precision, Recall, and F1 scores

The precision, recall and F1 scores for these three method are represented as below.

```
Precision, Recall, and F1 scores are as below.
              precision    recall  f1-score   support

           1       0.40      1.00      0.57         2
           2       0.00      0.00      0.00         1
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      0.33      0.50         3
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         1

    accuracy                           0.85        20
   macro avg       0.87      0.86      0.84        20
weighted avg       0.89      0.85      0.83        20

F1 scores = 0.8321428571428571
Precision = 0.89
recall = 0.85
```

Figure 1: Precision, Recall, and F1 scores for Bernoulli Naïve Bayes method

```
Precision, Recall, and F1 scores are as below.
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         3
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         2

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

F1 scores : 1.0
Precision : 1.0
recall : 1.0
```

Figure 2: Precision, Recall, and F1 scores for SVM method with linear kernel

```
Precision, Recall, and F1 scores are as below.
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         3
           4       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         3
          10       1.00      1.00      1.00         4
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         2
          13       1.00      1.00      1.00         3

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

F1 scores : 1.0
Precision : 1.0
recall : 1.0
```

Figure 3: Precision, Recall, and F1 scores for SVM method with RBF kernel

# Precision Recall Curves

The precision recall curves for these three method are represented as below.

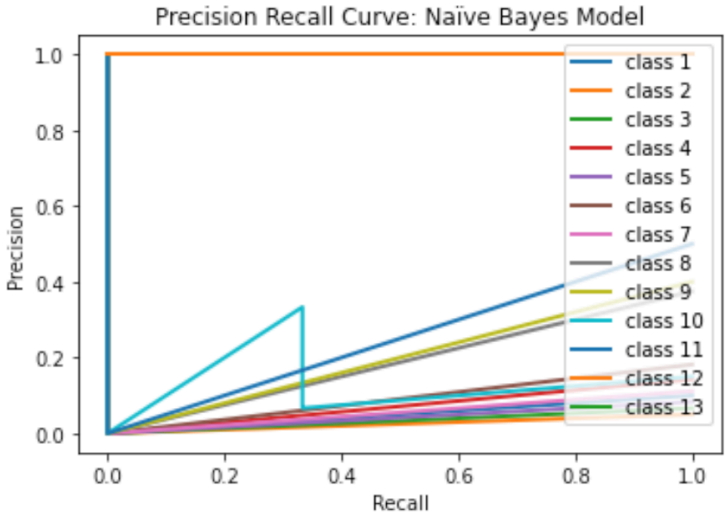


Figure 4: Precision Recall curve for Bernoulli Naïve Bayes method

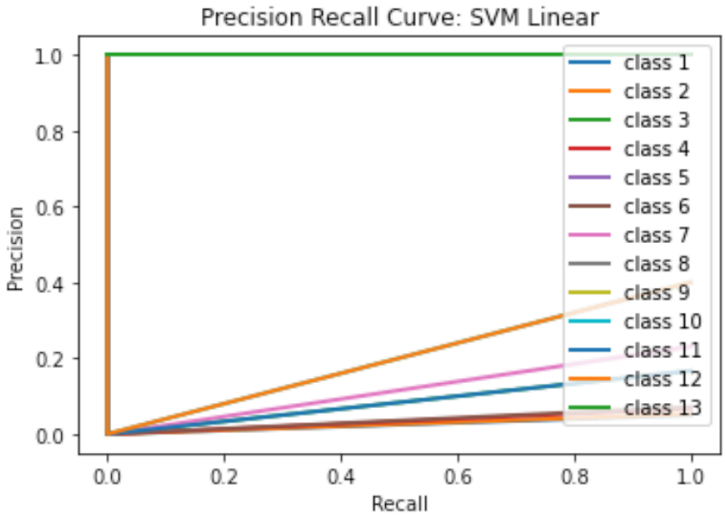


Figure 5: Precision Recall curve for SVM method with linear kernel

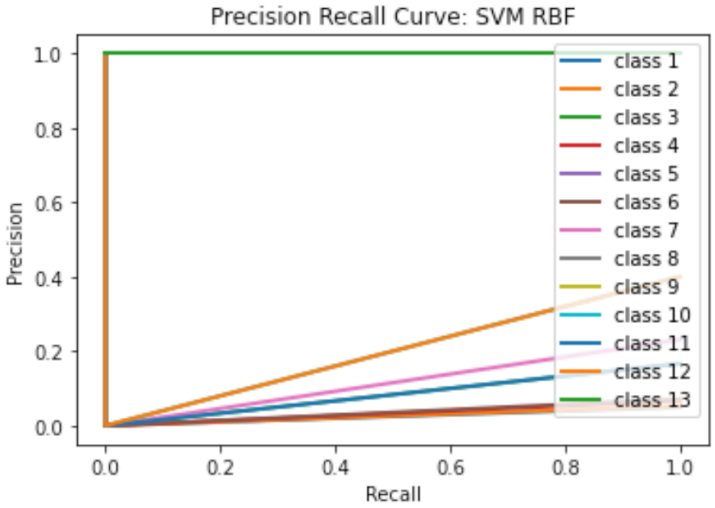


Figure 6: Precision Recall curve for SVM method with RBF kernel

# Scores on Kaggle

The submission score on Kaggle is shown as below. I get 0.98444 on this competition, ranking at first place on Nov. 11th.

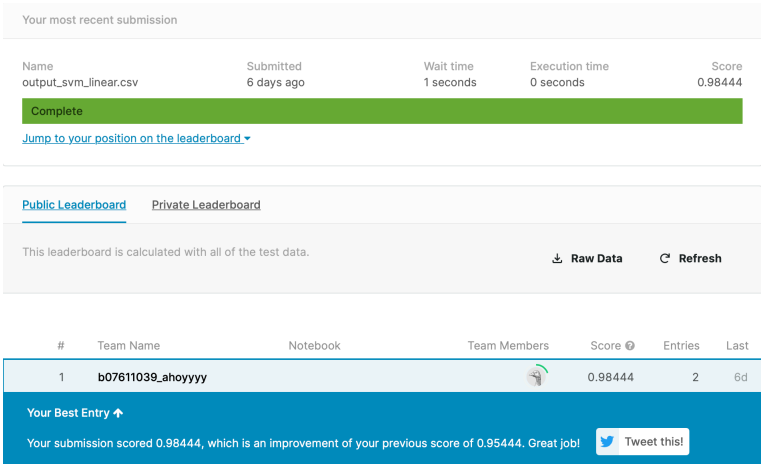


Figure 7: Score on Kaggle

# Implement

## Preprocess

1. Import all necessary libraries, for example, sklearn, numpy, pandas, etc.
2. Lowercase all words in docs, eliminate the stop word in English, replace EOL.
3. Separate the testing data set and the training data set.

```
# classes stores the given testing data set via a 2D array
labels = []
for q in range(0, len(docs)):
    for i in range(0, 13):
        for j in range(0, 15):
            if q + 1 == classes[i][j]:
```

```
                labels.append([classes[i][j], i+1])
    labels = pd.DataFrame(sorted(labels, key = lambda l:l[0]), columns = ['training_id', 'classes'])
    training_docs = docs[docs['id'].isin(labels['training_id'])]
    testing_docs = docs[~docs['id'].isin(labels['training_id'])]
```

## Bernoulli Naïve Bayes

```
binary_vectorizer = CountVectorizer(binary = True)
binary_vectors = binary_vectorizer.fit_transform(training_docs['text'])
binary_vectors_test = binary_vectorizer.transform(testing_docs['text'])
x_train, x_test, y_train, y_test = train_test_split(binary_vectors, labels['classes'], test_size =
    0.1) # amazing method taught by my classmates
model = BernoulliNB()
model.fit(x_train, y_train)
prediction = []
expectation = []
prediction.extend(model.predict(x_test))
expectation.extend(y_test)
```

## Linear Kernel SVM

```
TFIDF_vectorizer = TfidfVectorizer(stop_words = 'english')
TFIDF_vectors_training = TFIDF_vectorizer.fit_transform(training_docs['text'])
TFIDF_vectors_testing = TFIDF_vectorizer.transform(testing_docs['text'])

x_train, x_test, y_train, y_test = train_test_split(TFIDF_vectors_training, labels['classes'],
    test_size = 0.1)
SVC_Linear_model = SVC(kernel='linear', C = 1.0)
SVC_Linear_model.fit(x_train, y_train)

prediction = []
expectation = []

prediction.extend(SVC_Linear_model.predict(x_test))
expectation.extend(y_test)
```

## RBF Kernel SVM

```
TFIDF_vectorizer = TfidfVectorizer(stop_words = 'english')
TFIDF_vectors_training = TFIDF_vectorizer.fit_transform(training_docs['text'])
TFIDF_vectors_testing = TFIDF_vectorizer.transform(testing_docs['text'])

x_train, x_test, y_train, y_test = train_test_split(TFIDF_vectors_training, labels['classes'],
    test_size = 0.1)
SVC_RBF_model = SVC(kernel='RBF', C = 1.0)
SVC_RBF_model.fit(x_train, y_train)

prediction = []
expectation = []

prediction.extend(SVC_RBF_model.predict(x_test))
expectation.extend(y_test)
```

## Representation of scores

```
print("Precision, Recall, and F1 scores are as below.")
print(metrics.classification_report(expectation, prediction))
print("F1 scores :", metrics.f1_score(expectation, prediction, average='weighted'))
print("Precision :", metrics.precision_score(expectation, prediction, average='weighted'))
print("recall :", metrics.recall_score(expectation, prediction, average ='weighted'))
```

## Plotting

```
precision = dict()
recall = dict()
for i in range(13):
    precision[i], recall[i], thresholds = metrics.precision_recall_curve(expectation, prediction,
        pos_label = (i + 1))
    plt.plot(recall[i], precision[i], lw = 2, label = 'class {}'.format(i + 1))

plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend(loc = "upper right")
plt.title("Precision Recall Curve")
plt.show()
```