UN Sustainable Development Goals Classifier

BUILDING A MACHINE LEARNING MODEL TO DETECT UN SDGs TOPIC

Van-Thien Nguyen

Python for Data Science class 20

Table of content

PART 1	Introduction
PART 2	Input and Output
PART 3	Dataset
PART 4	Data Pre-processing
PART 5	Model Comparison
PART 6	Model Optimization
PART 7	Conclusion

Introduction

The Sustainable Development Goals (SDGs) or Global Goals are a collection of 17 interlinked global goals designed to be a "blueprint to achieve a better and more sustainable future for all". The SDGs were set up in 2015 by the United Nations General Assembly (UN-GA) and are intended to be achieved by 2030.

We need a tool to track stakeholders' alignment with SDGs

INPUT

Text Docs

OUTPUT

SDGs Category

32,327 text excerpts

206,546 assigned labels

Dataset from OSDG Community

The OSDG Community Dataset (OSDG-CD) is a public dataset of thousands of text excerpts, which were validated by approximately 1,000 OSDG Community Platform (OSDG-CP) citizen scientists from over 110 countries, with respect to the Sustainable Development Goals (SDGs).

- The volunteer is shown a text + an SDG label and asked to either accept or reject the suggested label.

- Each text is validated by up to 9 different volunteers.

Dataset

```
#Loading UN sustainable development goals dataset
df = pd.read_csv("UN_dataset_osdg_community.csv", delimiter="\t")
df.head()
```

	doi	text_id	text	sdg	labels_negative	labels_positive	agreement
0	10.6027/9789289342698-7-en	00021941702cd84171ff33962197ca1f	From a gender perspective, Paulgaard points ou...	5	1	8	0.777778
1	10.18356/eca72908-en	00028349a7f9b2485ff344ae44ccfd6b	Labour legislation regulates maximum working h...	11	2	1	0.333333
2	10.1787/9789264289062-4-en	0004eb64f96e1620cd852603d9cbe4d4	The average figure also masks large difference...	3	1	8	0.777778
3	10.1787/5k9b7bn5qzvd-en	0006a887475ccfa5a7f5f51d4ac83d02	The extent to which they are akin to corruptio...	3	1	2	0.333333
4	10.1787/9789264258211-6-en	0006d6e7593776abbdf4a6f985ea6d95	A region reporting a higher rate will not earn...	3	2	2	0.000000

- text - text excerpt from the document
- sdg - the SDG the text is validated against
- labels_negative - the number of volunteers who rejected the suggested SDG label
- labels_positive - the number of volunteers who accepted the suggested SDG label
- agreement - agreement score based on the formula agreement=(labels positive-labels negative)/(labels positive+labels negative)

Dataset

```
[26] df.dtypes
     df.info()
     df.describe()
```

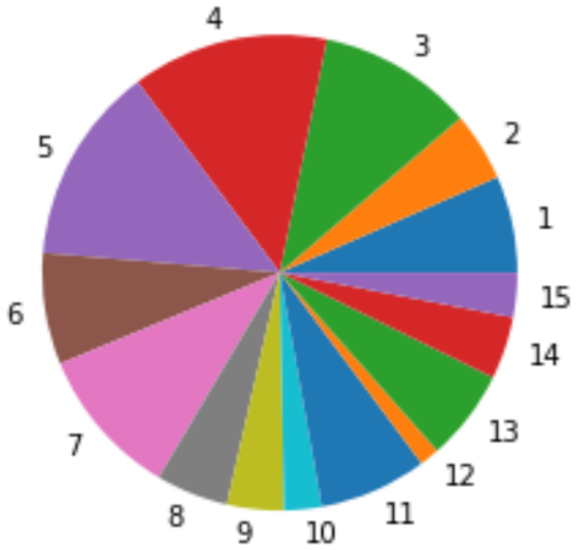
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32327 entries, 0 to 32326
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   doi              32327 non-null  object
 1   text_id          32327 non-null  object
 2   text             32327 non-null  object
 3   sdg              32327 non-null  int64
 4   labels_negative  32327 non-null  int64
 5   labels_positive  32327 non-null  int64
 6   agreement        32327 non-null  float64
dtypes: float64(1), int64(3), object(3)
memory usage: 1.7+ MB
```

	sdg	labels_negative	labels_positive	agreement
count	32327.000000	32327.000000	32327.000000	32327.000000
mean	6.513967	1.501872	4.893216	0.692060
std	3.950474	7.418859	13.021928	0.318663
min	1.000000	0.000000	0.000000	0.000000
25%	4.000000	0.000000	3.000000	0.333333
50%	6.000000	1.000000	4.000000	0.777778
75%	9.000000	2.000000	7.000000	1.000000
max	15.000000	837.000000	925.000000	1.000000

- labels_negative - the number of volunteers who rejected the suggested SDG label
- labels_positive - the number of volunteers who accepted the suggested SDG label
- agreement - agreement score based on the formula agreement=(labels positive-labels negative)/(labels positive+labels negative)

Dataset

```
#EDA
sdg_counts = df_filtered.groupby("sdg").count()
plt.pie(sdg_counts, labels=sdg_counts.index)
plt.show()
```



Data Pre-processing

Keeping only the texts whose suggested sdg labels is accepted and the agreement score is at least 0.6
Taking 2 columns sdg + text

```
#Keeping only the texts whose suggested sdg labels is accepted and the agreement score is at least 0.6
df_filtered = df[(df["agreement"] >= 0.6) & (df["labels_positive"] > df["labels_negative"])]
print(df_filtered.head())
df_filtered.info()
```

```
#Chỉ cần lấy 2 cột sdg + text để modelling
df_filtered = df_filtered[['sdg', 'text']]
df_filtered['text'] = df_filtered['text'].astype(str)
df_filtered.head()
```

Data Pre-processing

```
import nltk
nltk.download('stopwords')
stop = stopwords.words('english')
porter_stemmer = PorterStemmer()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#Hàm loại bỏ dấu phảy trong câu
def remove_punctuation(description):
    """The function to remove punctuation"""
    table = str.maketrans('', '', string.punctuation)
    return description.translate(table)

#Hàm loại bỏ các từ thừa, không có ý nghĩa trong câu
def remove_stopwords(text):
    """The function to removing stopwords"""
    text = [word.lower() for word in text.split() if word.lower() not in stop]
    return " ".join(text)


#Hàm để tìm dạng gốc của từ
#VD loves thì dạng gốc của từ sẽ là love
def stemmer(stem_text):
    """The function to apply stemming"""
    stem_text = [porter_stemmer.stem(word) for word in stem_text.split()]
    return " ".join(stem_text)
```

- Remove punctuation
- Remove stopwords
(ex: "s" in "loves")
- Stem the word to root word
(ex: "lovely" => "love"

Model Comparison

```
#So sánh độ chính xác của các model
lr_acc = accuracy_score(ypred_lg, y_test)
sgd_acc = accuracy_score(ypred_sgd, y_test)
nb_acc = accuracy_score(ypred_nb, y_test)
rf_acc = accuracy_score(ypred_rf, y_test)

#Creating a new dataframe containing all the model names and their corresponding accuracies.
models = pd.DataFrame({
                       'Model': ["Logistic Regression", "SGD Classifier", "Naive Bayes", "Random Forest"],
                       'Score': [lr_acc, sgd_acc, nb_acc, rf_acc]})
models.sort_values(by='Score', ascending=False)
```

	Model	Score
1	SGD Classifier	0.884604
0	Logistic Regression	0.877233
3	Random Forest	0.836688
2	Naive Bayes	0.714772

Stochastic Gradient Descent Classifier model has the highest accuracy

SDG Model Optimization

```
#Stochastic Gradient Descent Classifier model
pipeline = Pipeline([
  ('vect', CountVectorizer()),
  ('tfidf', TfidfTransformer()),
  ('clf', SGDClassifier(
      loss='modified_huber',
      penalty='l2',
      alpha=1e-3,
      random_state=42,
      max_iter=100,
      tol=None,
      )),
])
classifier = pipeline.fit(X_train, y_train)
ytest = np.array(y_test)
ypred_sgd = classifier.predict(X_test)
print(f"accuracy: {accuracy_score(ypred_sgd, y_test)}")
print(classification_report(ytest, ypred_sgd))
```

```
pipeline = Pipeline([
  ('vect', CountVectorizer()),
  ('tfidf', TfidfTransformer()),
  ('clf', SGDClassifier(
      loss='squared_hinge',
      penalty='l2',
      alpha=1e-3,
      random_state=42,
      max_iter=100,
      tol=None,
      )),
])
classifier = pipeline.fit(X_train, y_train)
ytest = np.array(y_test)
ypred_sgd = classifier.predict(X_test)
print(f"accuracy: {accuracy_score(ypred_sgd, y_test)}")
print(classification_report(ytest, ypred_sgd))
```

SDG Model Optimization

accuracy: 0.884604479727814

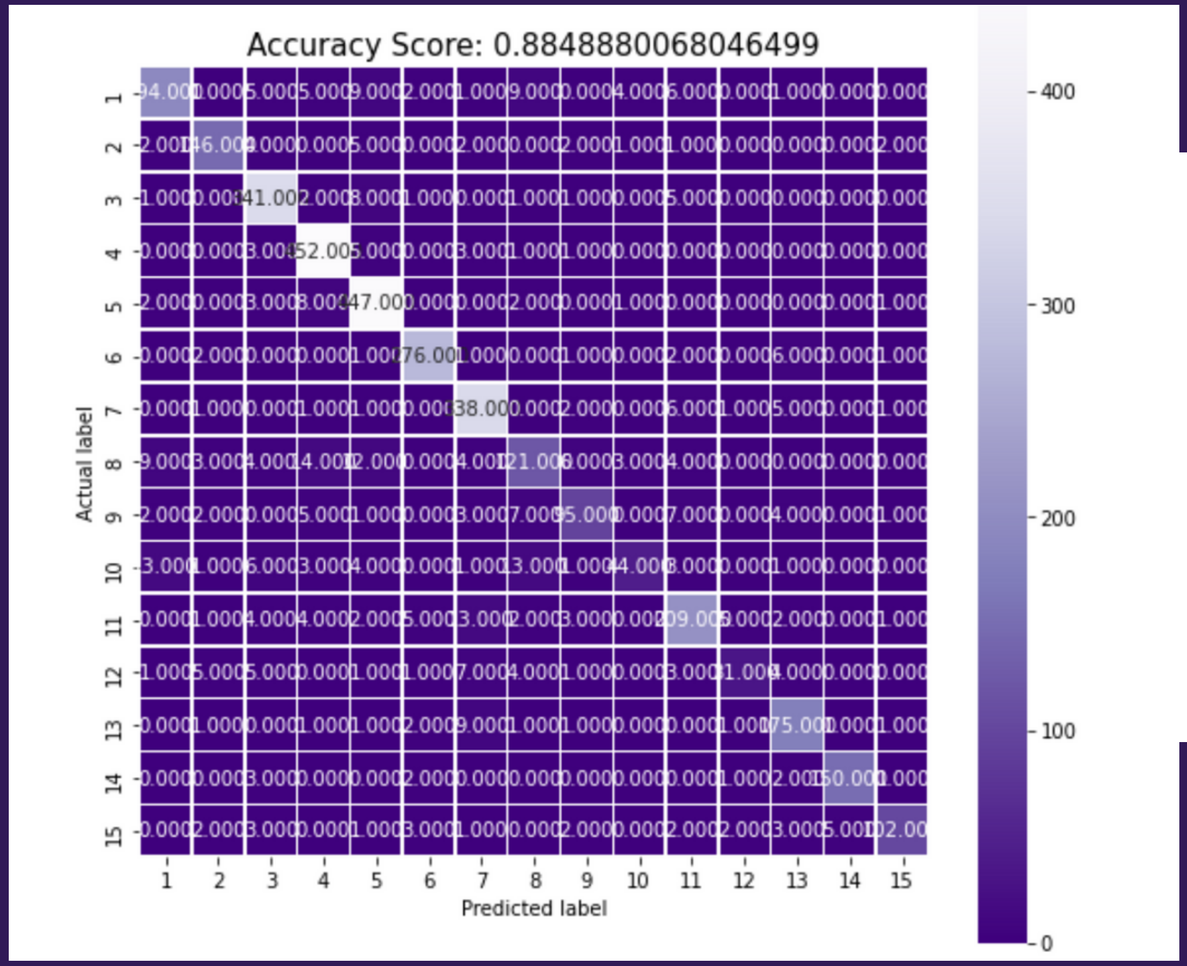
	precision	recall	f1-score	support
1	0.87	0.82	0.84	237
2	0.88	0.88	0.88	165
3	0.90	0.95	0.92	360
4	0.91	0.97	0.94	465
5	0.90	0.96	0.93	464
6	0.95	0.95	0.95	290
7	0.88	0.95	0.91	356
8	0.75	0.67	0.71	180
9	0.82	0.75	0.78	127
10	0.81	0.49	0.61	90
11	0.84	0.83	0.84	251
12	0.76	0.49	0.60	63
13	0.86	0.90	0.88	194
14	0.96	0.94	0.95	159
15	0.92	0.81	0.86	126
accuracy			0.88	3527
macro avg	0.87	0.82	0.84	3527
weighted avg	0.88	0.88	0.88	3527

Accuracy from 0.8846 to 0.8848

accuracy: 0.8848880068046499

	precision	recall	f1-score	support
1	0.87	0.82	0.84	237
2	0.88	0.88	0.88	165
3	0.90	0.95	0.92	360
4	0.91	0.97	0.94	465
5	0.90	0.96	0.93	464
6	0.95	0.95	0.95	290
7	0.88	0.95	0.91	356
8	0.75	0.67	0.71	180
9	0.82	0.75	0.78	127
10	0.83	0.49	0.62	90
11	0.84	0.83	0.84	251
12	0.76	0.49	0.60	63
13	0.86	0.90	0.88	194
14	0.96	0.94	0.95	159
15	0.92	0.81	0.86	126
accuracy			0.88	3527
macro avg	0.87	0.83	0.84	3527
weighted avg	0.88	0.88	0.88	3527

SDG Model Optimization




```
[[194   1   5   5   9   2   1   9   0   4   6   0   1   0   0]
 [  2 146   4   0   5   0   2   0   2   1   1   0   0   0   2]
 [  1   0 341   2   8   1   0   1   1   0   5   0   0   0   0]
 [  0   0   3 452   5   0   3   1   1   0   0   0   0   0   0]
 [  2   0   3   8 447   0   0   2   0   1   0   0   0   0   1]
 [  0   2   0   0   1 276   1   0   1   0   2   0   6   0   1]
 [  0   1   0   1   1   0 338   0   2   0   6   1   5   0   1]
 [  9   3   4  14  12   0   4 121   6   3   4   0   0   0   0]
 [  2   2   0   5   1   0   3   7  95   0   7   0   4   0   1]
 [ 13   1   6   3   4   0   1  13   1  44   3   0   1   0   0]
 [  0   1   4   4   2   5  13   2   3   0 209   5   2   0   1]
 [  1   5   5   0   1   1   7   4   1   0   3  31   4   0   0]
 [  0   1   0   1   1   2   9   1   1   0   0   1 175   1   1]
 [  0   0   3   0   0   2   0   0   0   0   0   1   2 150   1]
 [  0   2   3   0   1   3   1   0   2   0   2   2   3   5 102]]
```

Conclusion

Overall, the model performance is good.

- Precision ranges from 0.75 to 0.95
- Recall ranges from 0.49 (SDG10+SDG12) to 0.97
- F1 score ranges from 0.62 (SDG10) to 0.95

Precision indicates how many positive predictions are true.
Precision = TP/(TP+FP)
Recall indicates how many of the positive classes the model is able to predict correctly.
Recall = TP/(TP+FN)
F1 score is the weighted average of precision and recall.
F1_score = 2*(Precision * Recall)/(Precision + Recall)

accuracy: 0.8848880068046499

	precision	recall	f1-score	support
1	0.87	0.82	0.84	237
2	0.88	0.88	0.88	165
3	0.90	0.95	0.92	360
4	0.91	0.97	0.94	465
5	0.90	0.96	0.93	464
6	0.95	0.95	0.95	290
7	0.88	0.95	0.91	356
8	0.75	0.67	0.71	180
9	0.82	0.75	0.78	127
10	0.83	0.49	0.62	90
11	0.84	0.83	0.84	251
12	0.76	0.49	0.60	63
13	0.86	0.90	0.88	194
14	0.96	0.94	0.95	159
15	0.92	0.81	0.86	126
accuracy			0.88	3527
macro avg	0.87	0.83	0.84	3527
weighted avg	0.88	0.88	0.88	3527

Thank you for listening!

Data set link: https://zenodo.org/record/6393942

THE GLOBAL GOALS

Van-Thien Nguyen

Python for Data Science class 20