SGD + CRF Method to Extract Brand Name in Product Titles

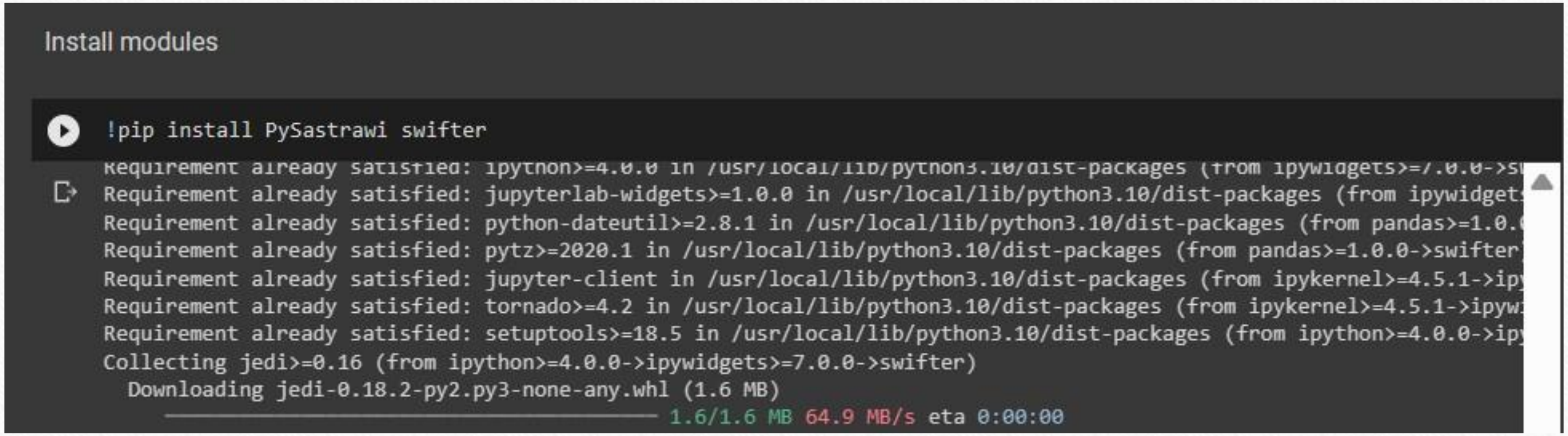
By: Can Şentay

Main Objective

- Our main objective is to train the model with the product titles that have labeled with brands. After that we are going to predict with some listing title to predict the brand inside of it listing title. The method that I will use is SGD and CRF.

Installing modules

- First of all we have to install libraries needed such as Pysastrawi for Indonesian words and swifter to track the running code.

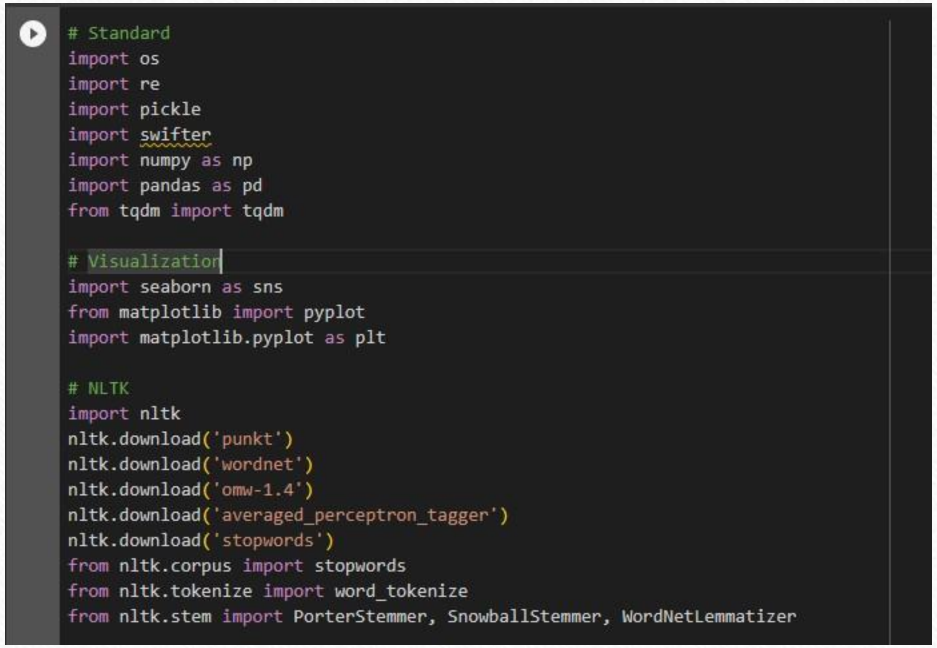


Install modules

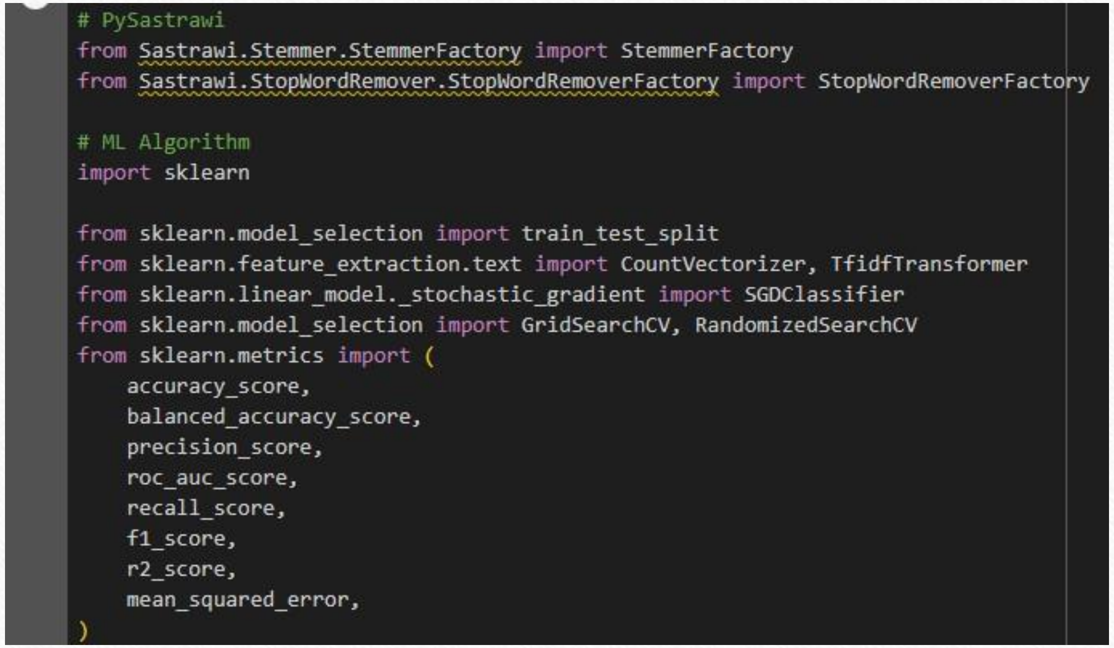
```
!pip install PySastrawi swifter
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.0.0->sw
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->swifter)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.10/dist-packages (from ipykernel>=4.5.1->ipy
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/dist-packages (from ipykernel>=4.5.1->ipywi
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist-packages (from ipython>=4.0.0->ipy
Collecting jedi>=0.16 (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter)
  Downloading jedi-0.18.2-py2.py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 64.9 MB/s eta 0:00:00
```

Importing Libraries

- Here is the libraries I’ve used to create the model



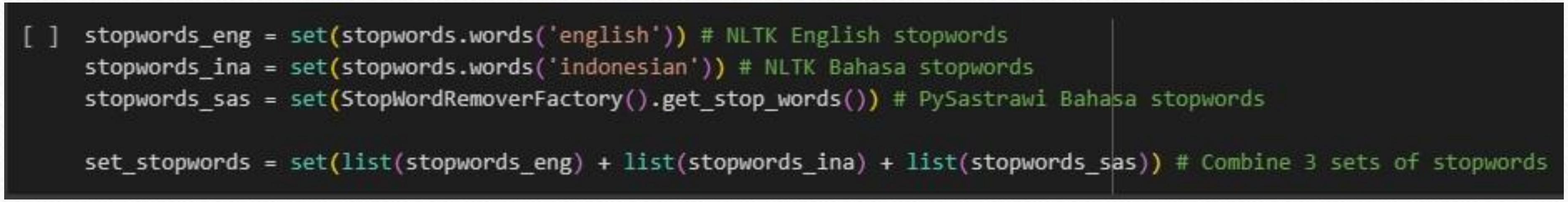
```
# Standard
import os
import re
import pickle
import swifter
import numpy as np
import pandas as pd
from tqdm import tqdm

# Visualization
import seaborn as sns
from matplotlib import pyplot
import matplotlib.pyplot as plt

# NLTK
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, SnowballStemmer, WordNetLemmatizer
```



```
# PySastrawi
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory

# ML Algorithm
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.linear_model._stochastic_gradient import SGDClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import (
    accuracy_score,
    balanced_accuracy_score,
    precision_score,
    roc_auc_score,
    recall_score,
    f1_score,
    r2_score,
    mean_squared_error,
)
```

Setting Stopwords

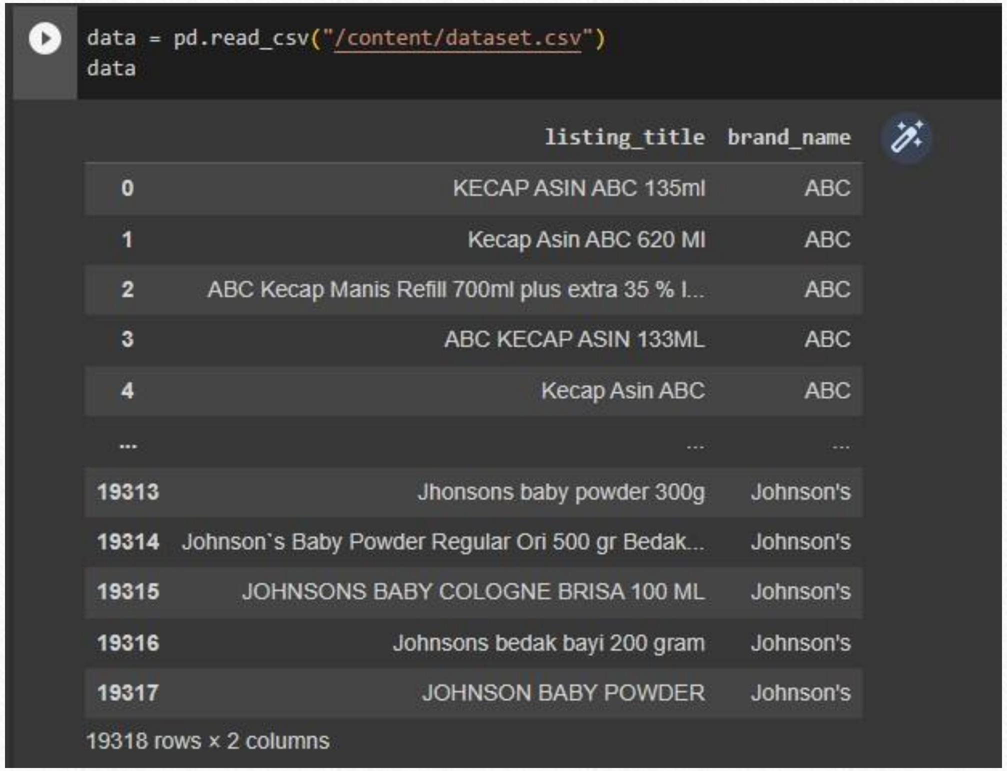
- Setting stopwords is very important to remove some general words such as “I”, “You”, “And”, “Then”, etc



```
stopwords_eng = set(stopwords.words('english')) # NLTK English stopwords
stopwords_ina = set(stopwords.words('indonesian')) # NLTK Bahasa stopwords
stopwords_sas = set(StopWordRemoverFactory().get_stop_words()) # PySastrawi Bahasa stopwords

set_stopwords = set(list(stopwords_eng) + list(stopwords_ina) + list(stopwords_sas)) # Combine 3 sets of stopwords
```

Loading Data

- Here is the data that I use, The listing_title is the product title and also we are going to tell our model about the brands of each of our listing_title.



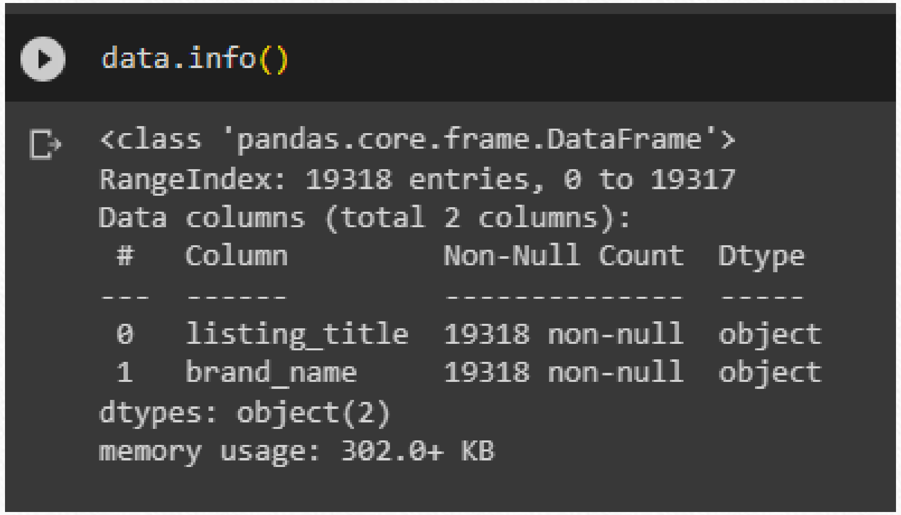
```
data = pd.read_csv("/content/dataset.csv")
data
```

	listing_title	brand_name
0	KECAP ASIN ABC 135ml	ABC
1	Kecap Asin ABC 620 Ml	ABC
2	ABC Kecap Manis Refill 700ml plus extra 35 % l...	ABC
3	ABC KECAP ASIN 133ML	ABC
4	Kecap Asin ABC	ABC
...	...	...
19313	Jhonsons baby powder 300g	Johnson's
19314	Johnson`s Baby Powder Regular Ori 500 gr Bedak...	Johnson's
19315	JOHNSONS BABY COLOGNE BRISA 100 ML	Johnson's
19316	Johnsons bedak bayi 200 gram	Johnson's
19317	JOHNSON BABY POWDER	Johnson's

19318 rows × 2 columns

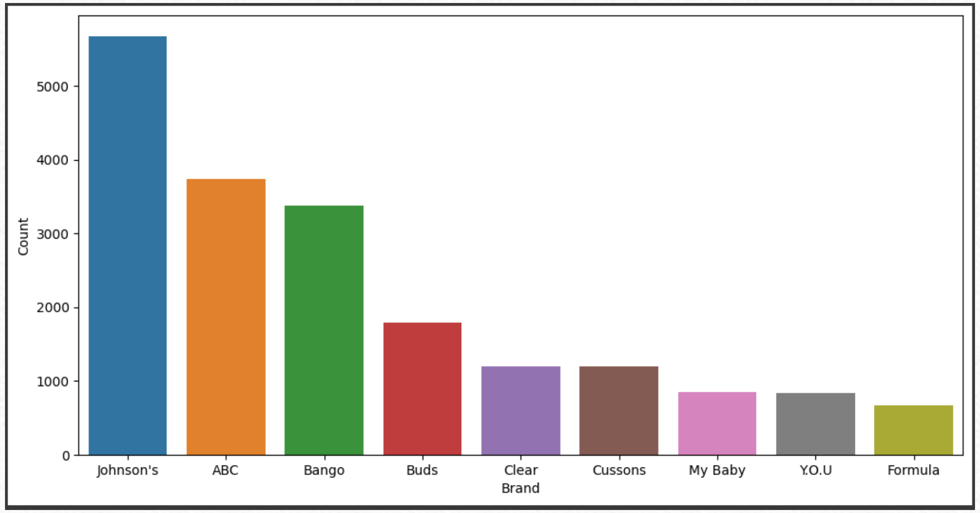
Inspecting Data

- By inspecting our data we could know how clean our data is. Here we have the imbalance dataset, so to evade the overfitting we could increase the volume of the dataset or we can use undersampling or oversampling.



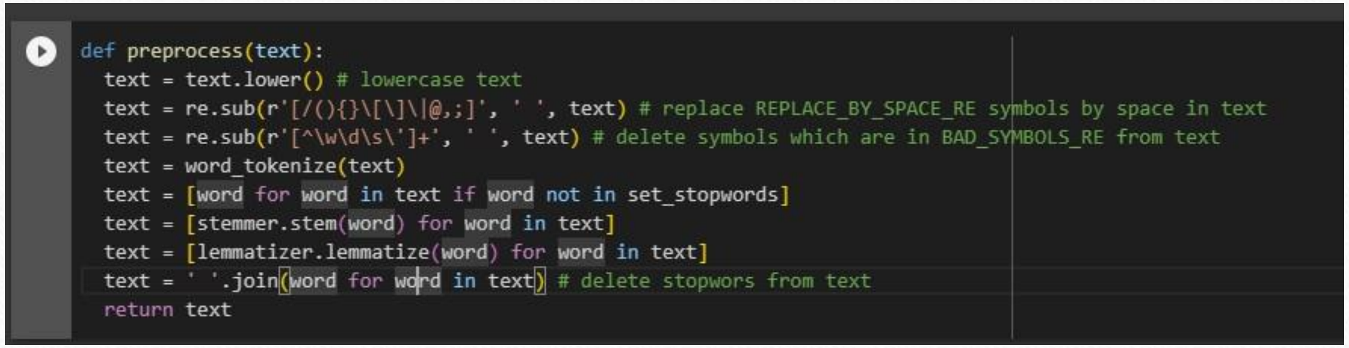
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19318 entries, 0 to 19317
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   listing_title  19318 non-null  object
 1   brand_name     19318 non-null  object
dtypes: object(2)
memory usage: 302.0+ KB
```

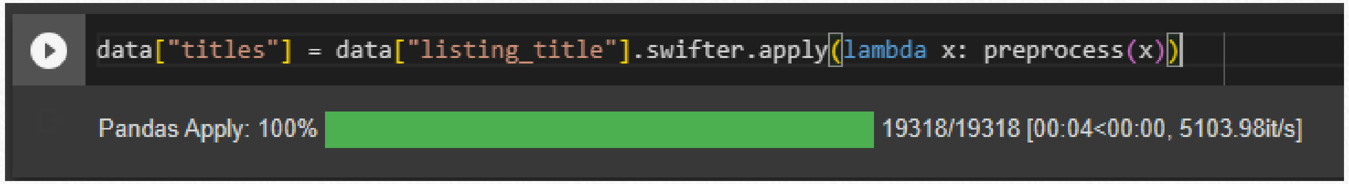


Cleansing

- We have to clean some symbol, excess whitespace, general words, etc with this function:



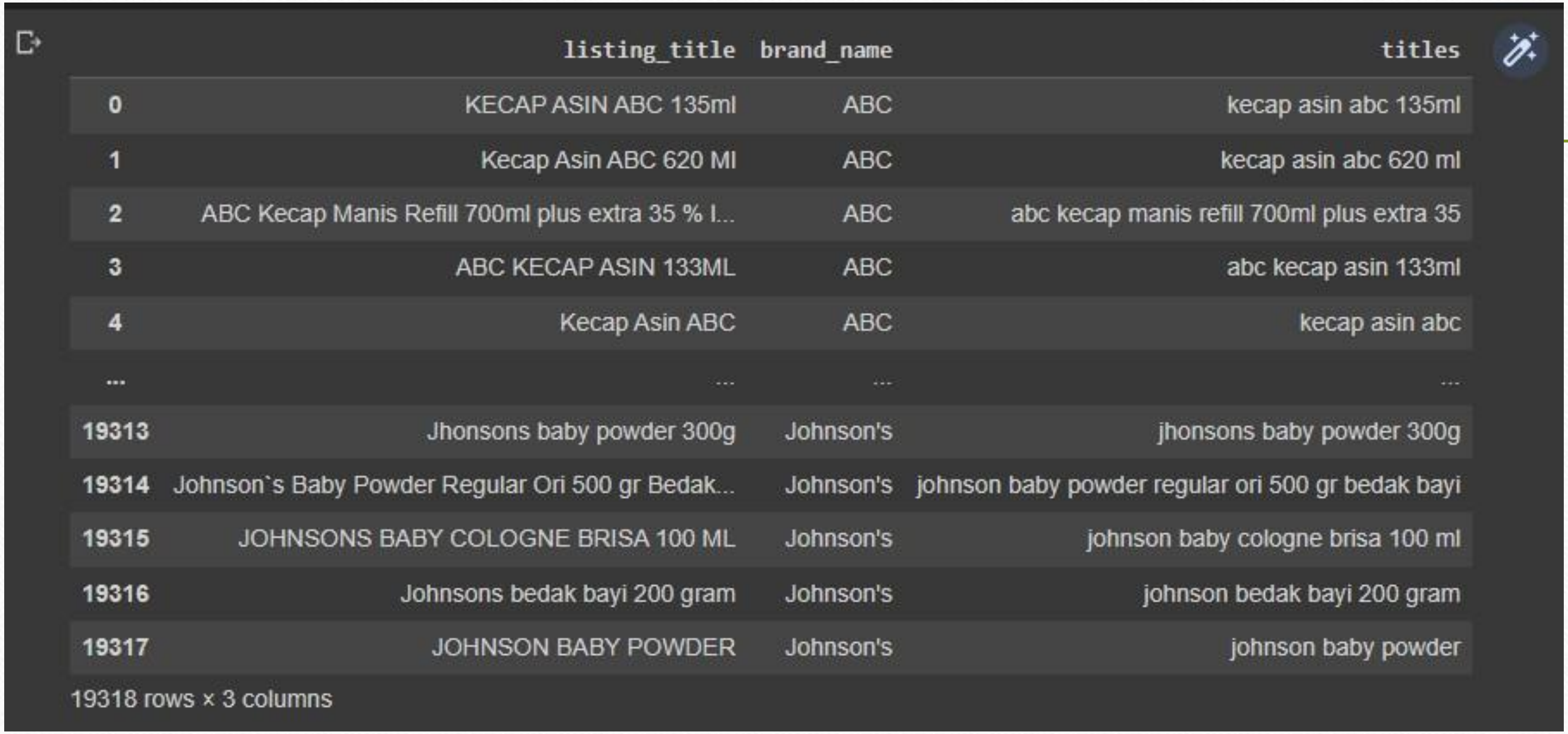
```
def preprocess(text):
  text = text.lower() # lowercase text
  text = re.sub(r'[/(){}\[\]\|@,;]', ' ', text) # replace REPLACE_BY_SPACE_RE symbols by space in text
  text = re.sub(r'[^\w\d\s\']+', ' ', text) # delete symbols which are in BAD_SYMBOLS_RE from text
  text = word_tokenize(text)
  text = [word for word in text if word not in set_stopwords]
  text = [stemmer.stem(word) for word in text]
  text = [lemmatizer.lemmatize(word) for word in text]
  text = ' '.join(word for word in text) # delete stopwors from text
  return text
```



```
data["titles"] = data["listing_title"].swifter.apply(lambda x: preprocess(x))
```

```
Pandas Apply: 100% 19318/19318 [00:04<00:00, 5103.98it/s]
```

- The data set will become like this:

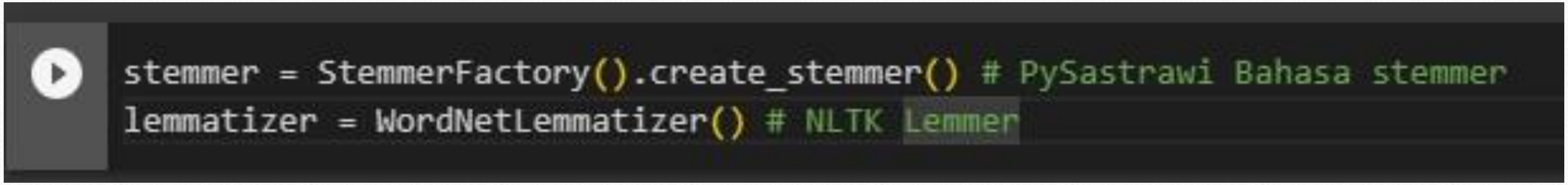


	listing_title	brand_name	titles
0	KECAP ASIN ABC 135ml	ABC	kecap asin abc 135ml
1	Kecap Asin ABC 620 Ml	ABC	kecap asin abc 620 ml
2	ABC Kecap Manis Refill 700ml plus extra 35 % l...	ABC	abc kecap manis refill 700ml plus extra 35
3	ABC KECAP ASIN 133ML	ABC	abc kecap asin 133ml
4	Kecap Asin ABC	ABC	kecap asin abc
...	...	...	...
19313	Jhonsons baby powder 300g	Johnson's	jhonsons baby powder 300g
19314	Johnson`s Baby Powder Regular Ori 500 gr Bedak...	Johnson's	johnson baby powder regular ori 500 gr bedak bayi
19315	JOHNSONS BABY COLOGNE BRISA 100 ML	Johnson's	johnson baby cologne brisa 100 ml
19316	Johnsons bedak bayi 200 gram	Johnson's	johnson bedak bayi 200 gram
19317	JOHNSON BABY POWDER	Johnson's	johnson baby powder

19318 rows × 3 columns

Stemming and Lemmatizing the words

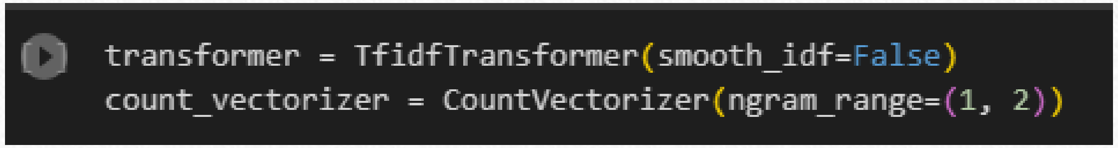
- We could use stemming and lemmatizing to reduce words in our dataset. Stemming is taking the root of the words such as eating after stemmed would become eat, and lemmatize is the method of returning words based on dictionary such as Studies would become Study.



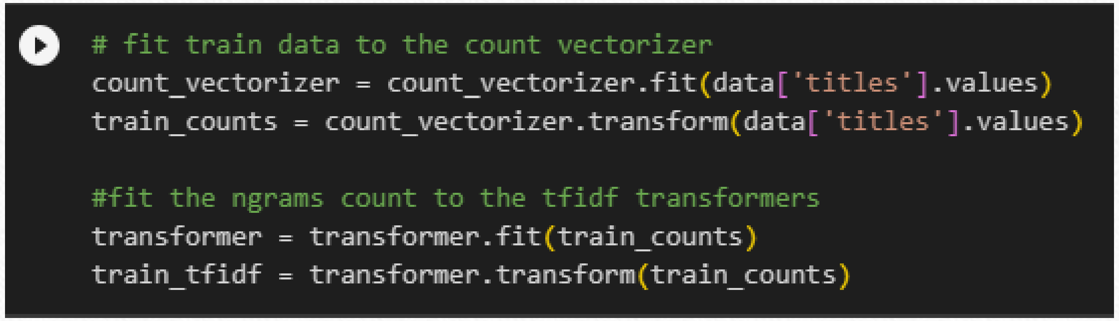
```
stemmer = StemmerFactory().create_stemmer() # PySastrawi Bahasa stemmer
lemmatizer = WordNetLemmatizer() # NLTK Lemmer
```

TF-IDF and Vectorizer

- Here we are going to use some nlp methods such as TF-IDF and Vectorizer, to summarize we are going to chunk our dataset column “titles” into 1-2 words (called ngrams).



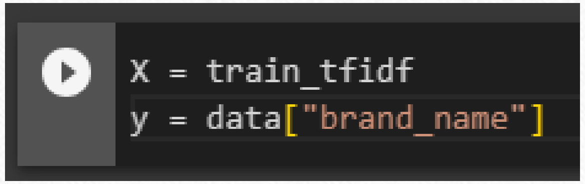
```
transformer = TfidfTransformer(smooth_idf=False)
count_vectorizer = CountVectorizer(ngram_range=(1, 2))
```



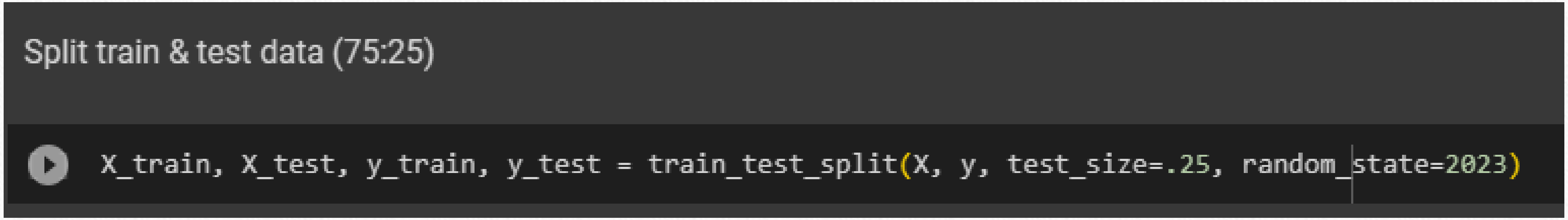
```
# fit train data to the count vectorizer
count_vectorizer = count_vectorizer.fit(data['titles'].values)
train_counts = count_vectorizer.transform(data['titles'].values)

#fit the ngrams count to the tfidf transformers
transformer = transformer.fit(train_counts)
train_tfidf = transformer.transform(train_counts)
```

Defining Train and Test Data

- Here we are going to define the train and test data



```
X = train_tfidf
y = data["brand_name"]
```

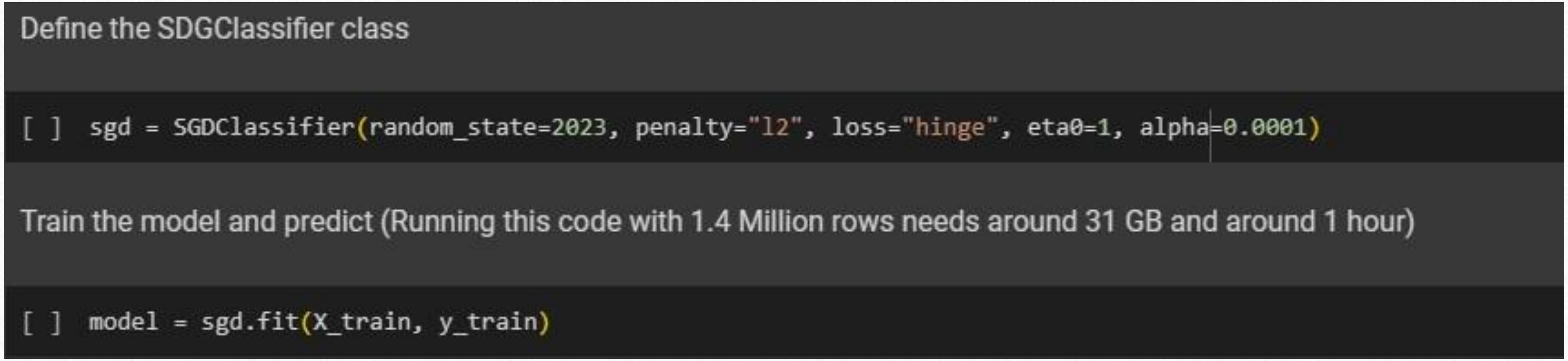


Split train & test data (75:25)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25, random_state=2023)
```

Defining the model

- Here we are going to use SGD to make our classification model.



Define the SDGClassifier class

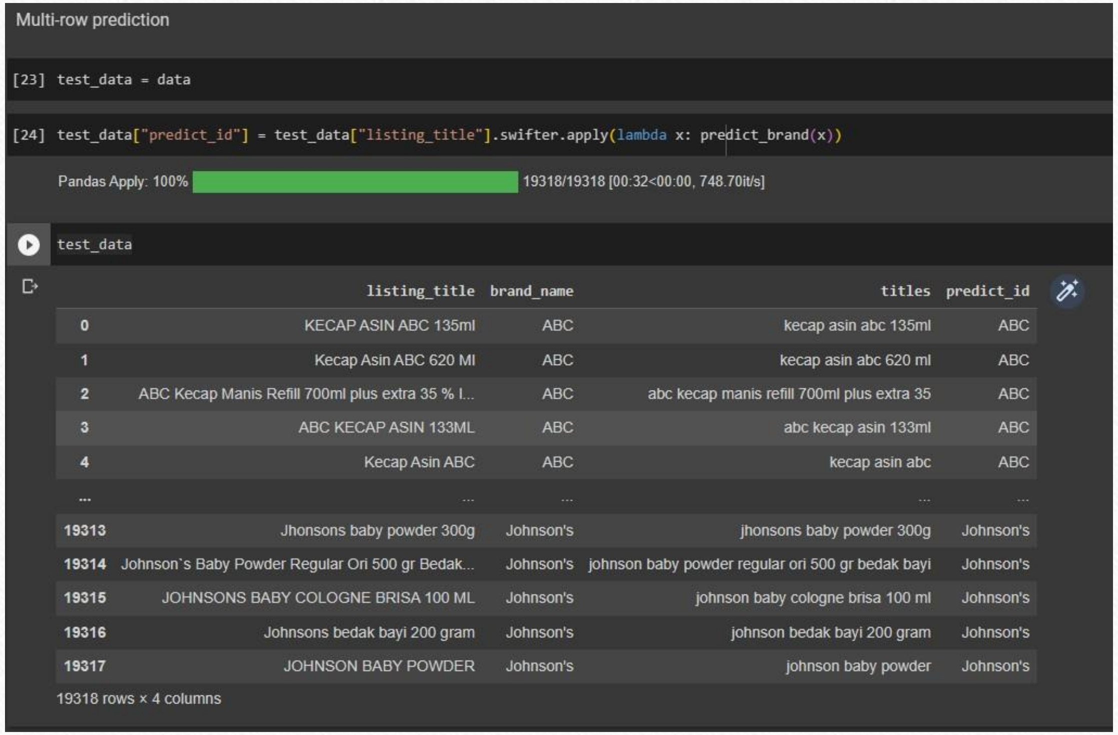
```
sgd = SGDClassifier(random_state=2023, penalty="l2", loss="hinge", eta0=1, alpha=0.0001)
```

Train the model and predict (Running this code with 1.4 Million rows needs around 31 GB and around 1 hour)

```
model = sgd.fit(X_train, y_train)
```

Predicting the model

- We can insert dataset test into our code, but you have to remember by only inserting the product titles with brands that only contained in our train data because this is classification. The model will not perform good if you add random brands that we are not even training it before.



Multi-row prediction

```
[23] test_data = data
```

```
[24] test_data["predict_id"] = test_data["listing_title"].swifter.apply(lambda x: predict_brand(x))
```

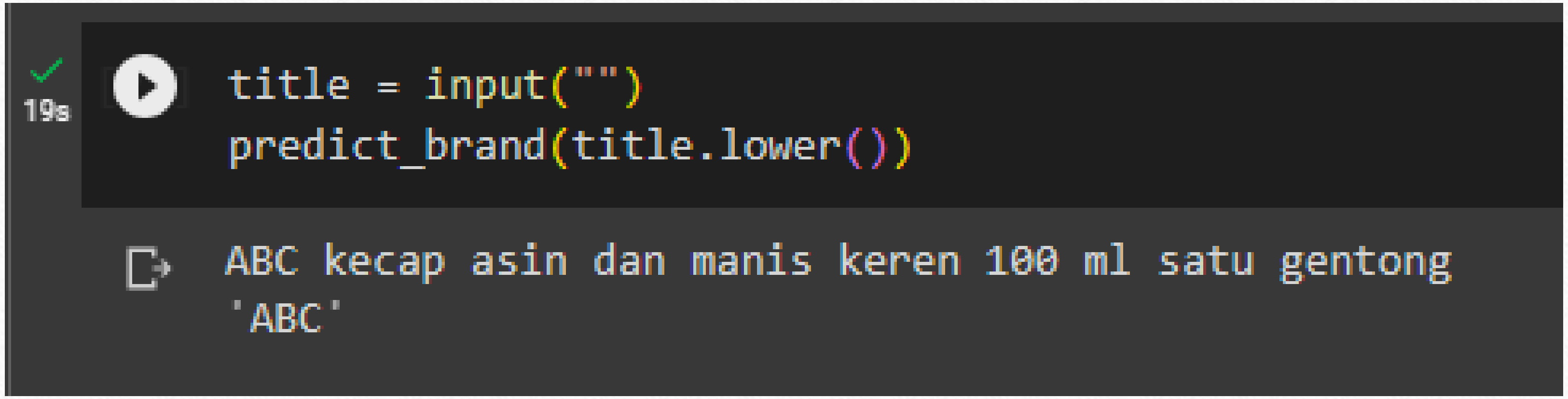
```
Pandas Apply: 100% 19318/19318 [00:32<00:00, 748.70it/s]
```

```
test_data
```

	listing_title	brand_name	titles	predict_id
0	KECAP ASIN ABC 135ml	ABC	kecap asin abc 135ml	ABC
1	Kecap Asin ABC 620 Ml	ABC	kecap asin abc 620 ml	ABC
2	ABC Kecap Manis Refill 700ml plus extra 35 % I...	ABC	abc kecap manis refill 700ml plus extra 35	ABC
3	ABC KECAP ASIN 133ML	ABC	abc kecap asin 133ml	ABC
4	Kecap Asin ABC	ABC	kecap asin abc	ABC
...	...	...	...	...
19313	Jhonsons baby powder 300g	Johnson's	jhonsons baby powder 300g	Johnson's
19314	Johnson`s Baby Powder Regular Ori 500 gr Bedak...	Johnson's	johnson baby powder regular ori 500 gr bedak bayi	Johnson's
19315	JOHNSONS BABY COLOGNE BRISA 100 ML	Johnson's	johnson baby cologne brisa 100 ml	Johnson's
19316	Johnsons bedak bayi 200 gram	Johnson's	johnson bedak bayi 200 gram	Johnson's
19317	JOHNSON BABY POWDER	Johnson's	johnson baby powder	Johnson's

19318 rows × 4 columns

- You can also predict with 1 listing title only by typing the listing title.

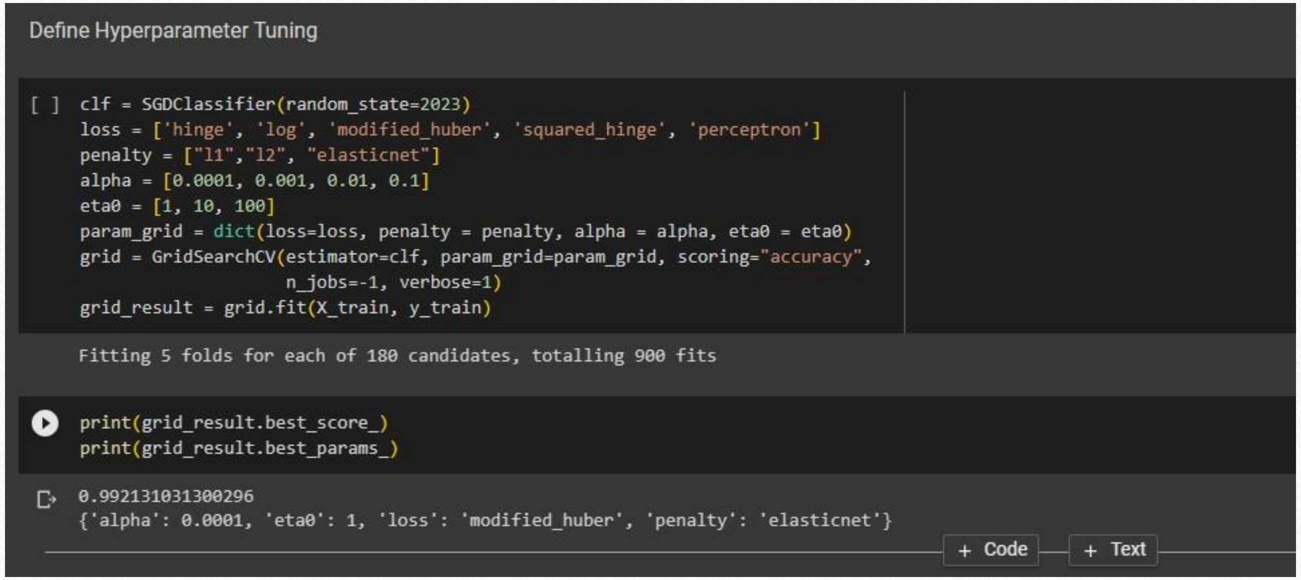


```
title = input("")
predict_brand(title.lower())
```

```
ABC kecap asin dan manis keren 100 ml satu gentong
'ABC'
```

Hyperparameter Tunning

- We can do some tunning in our model to make a better model by selecting the correct parameter.



Define Hyperparameter Tuning

```
clf = SGDClassifier(random_state=2023)
loss = ['hinge', 'log', 'modified_huber', 'squared_hinge', 'perceptron']
penalty = ["l1","l2", "elasticnet"]
alpha = [0.0001, 0.001, 0.01, 0.1]
eta0 = [1, 10, 100]
param_grid = dict(loss=loss, penalty = penalty, alpha = alpha, eta0 = eta0)
grid = GridSearchCV(estimator=clf, param_grid=param_grid, scoring="accuracy",
                    n_jobs=-1, verbose=1)
grid_result = grid.fit(X_train, y_train)
```

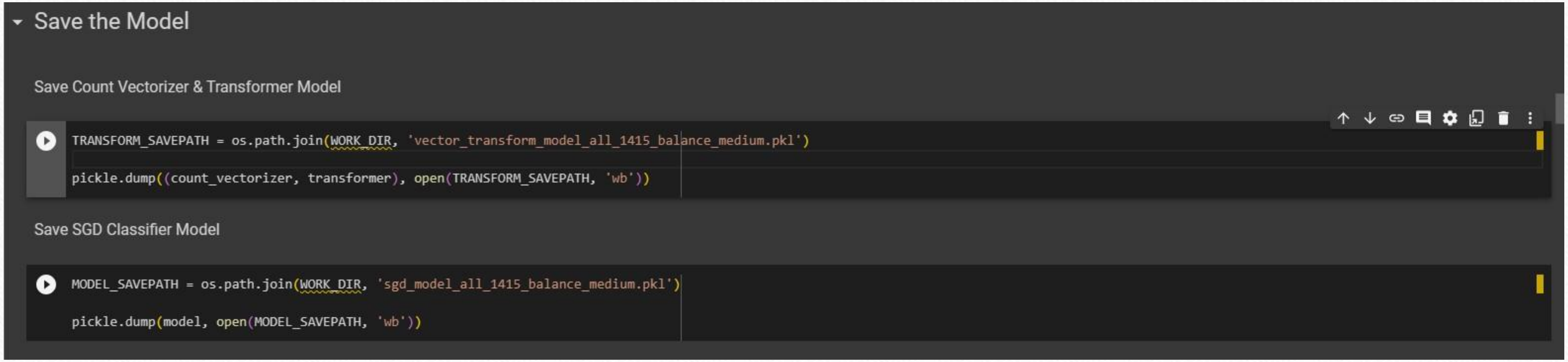
```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
```

```
print(grid_result.best_score_)
print(grid_result.best_params_)
```

```
0.992131031300296
{'alpha': 0.0001, 'eta0': 1, 'loss': 'modified_huber', 'penalty': 'elasticnet'}
```

Saving model

- You can save the model with pickle.dump function



Save the Model

Save Count Vectorizer & Transformer Model

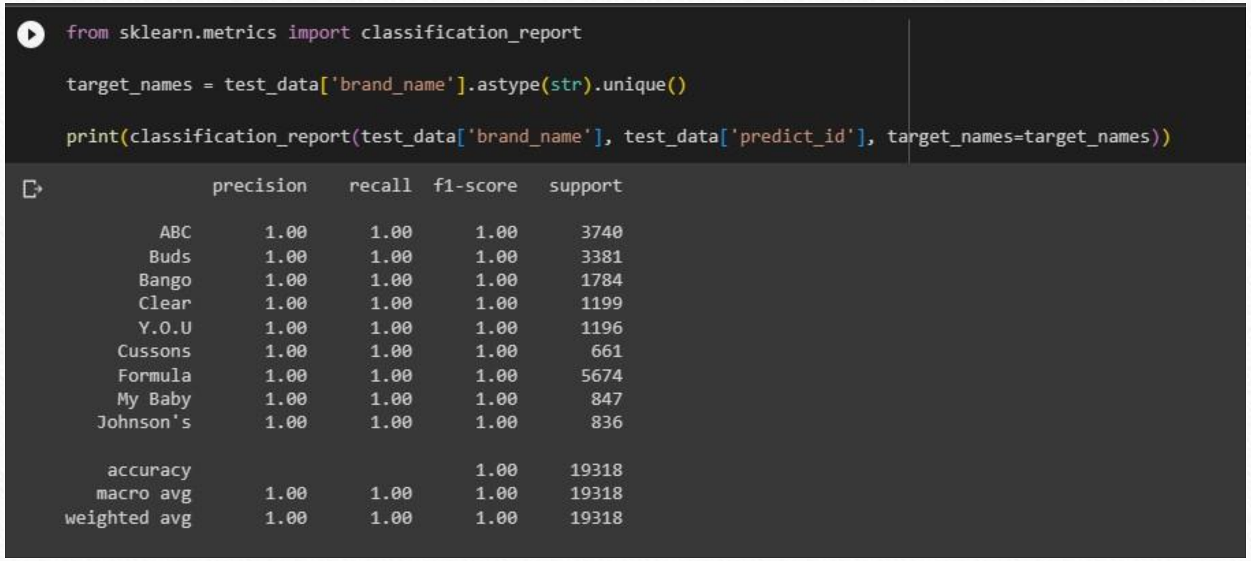
```
TRANSFORM_SAVEPATH = os.path.join(WORK_DIR, 'vector_transform_model_all_1415_balance_medium.pkl')
pickle.dump((count_vectorizer, transformer), open(TRANSFORM_SAVEPATH, 'wb'))
```

Save SGD Classifier Model

```
MODEL_SAVEPATH = os.path.join(WORK_DIR, 'sgd_model_all_1415_balance_medium.pkl')
pickle.dump(model, open(MODEL_SAVEPATH, 'wb'))
```

The Metrics Score

- Here is the detail of our metrics score. It has good precission, recall, f1 score and accuracy in which indicates our model works perfectly



```
from sklearn.metrics import classification_report

target_names = test_data['brand_name'].astype(str).unique()

print(classification_report(test_data['brand_name'], test_data['predict_id'], target_names=target_names))
```

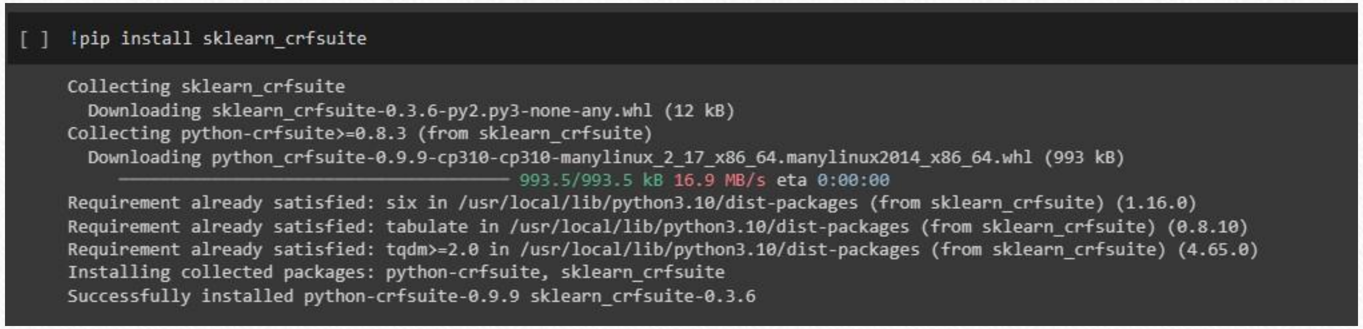
```
              precision    recall  f1-score   support

         ABC       1.00      1.00      1.00      3740
        Buds       1.00      1.00      1.00      3381
       Bango       1.00      1.00      1.00      1784
       Clear       1.00      1.00      1.00      1199
       Y.O.U       1.00      1.00      1.00      1196
     Cussons       1.00      1.00      1.00       661
     Formula       1.00      1.00      1.00      5674
     My Baby       1.00      1.00      1.00       847
   Johnson's       1.00      1.00      1.00       836

    accuracy                           1.00     19318
   macro avg       1.00      1.00      1.00     19318
weighted avg       1.00      1.00      1.00     19318
```

Additional Methods (Conditional Random Fields)

- We can use the CRF method to extract our brand, this method usually called feature extraction.
- The model is working with pattern and context from the input, so we have to label our dataset too.
- This method has little RAM consumption than the SGD Classification.
- Reference: https://github.com/maciej-cecot/brand-detection

Installing modules

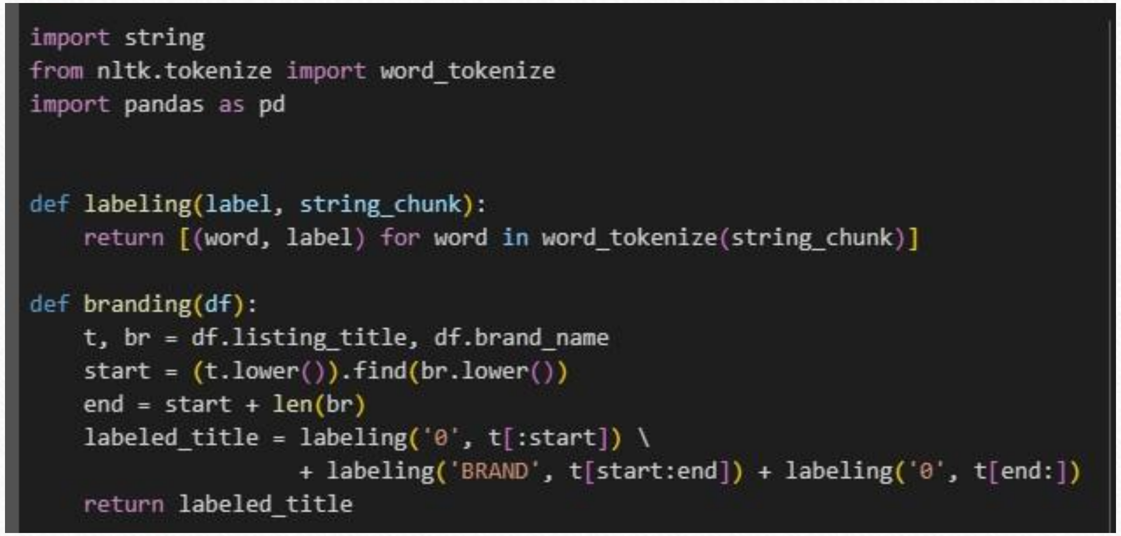
- We have to install the sklearn_crfsuite to train our model later



```
[ ]  !pip install sklearn_crfsuite

     Collecting sklearn_crfsuite
       Downloading sklearn_crfsuite-0.3.6-py2.py3-none-any.whl (12 kB)
     Collecting python-crfsuite>=0.8.3 (from sklearn_crfsuite)
       Downloading python_crfsuite-0.9.9-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (993 kB)
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 993.5/993.5 kB 16.9 MB/s eta 0:00:00
     Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from sklearn_crfsuite) (1.16.0)
     Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (from sklearn_crfsuite) (0.8.10)
     Requirement already satisfied: tqdm>=2.0 in /usr/local/lib/python3.10/dist-packages (from sklearn_crfsuite) (4.65.0)
     Installing collected packages: python-crfsuite, sklearn_crfsuite
     Successfully installed python-crfsuite-0.9.9 sklearn_crfsuite-0.3.6
```

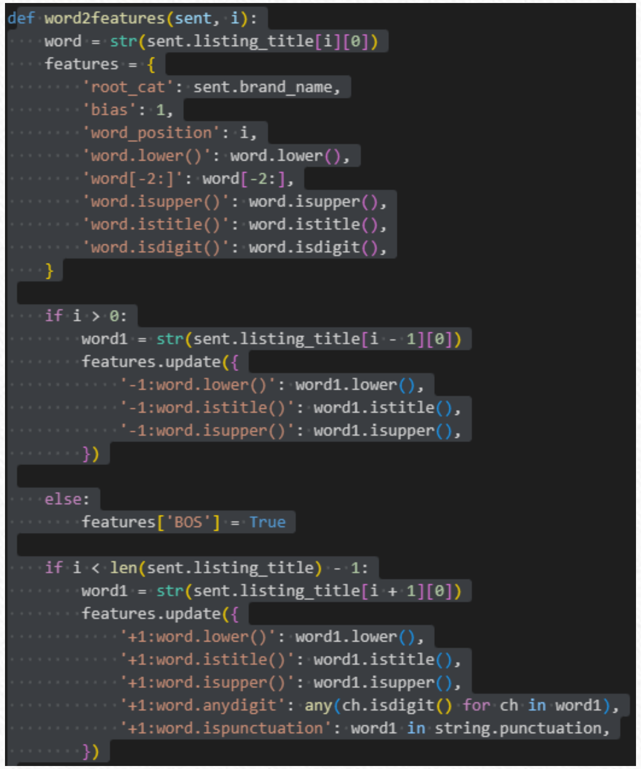
Defining function and class

- We have to chunk the listing_title into words to label the brands for our dataset training.

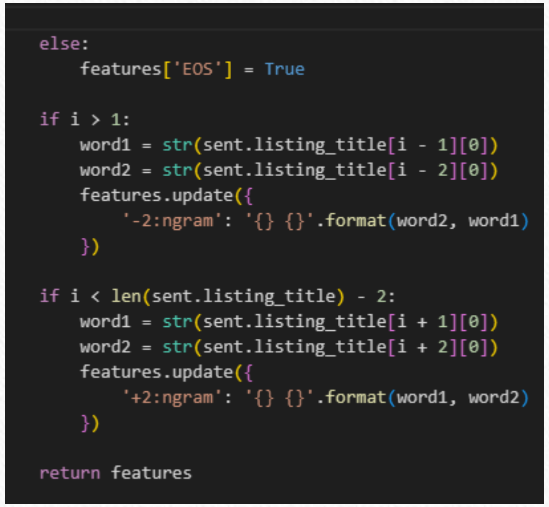


```python
import string
from nltk.tokenize import word_tokenize
import pandas as pd


def labeling(label, string_chunk):
    return [(word, label) for word in word_tokenize(string_chunk)]

def branding(df):
    t, br = df.listing_title, df.brand_name
    start = (t.lower()).find(br.lower())
    end = start + len(br)
    labeled_title = labeling('0', t[:start]) \
                    + labeling('BRAND', t[start:end]) + labeling('0', t[end:])
    return labeled_title
```

- Here we are going to give some context for our input such as word before and after.

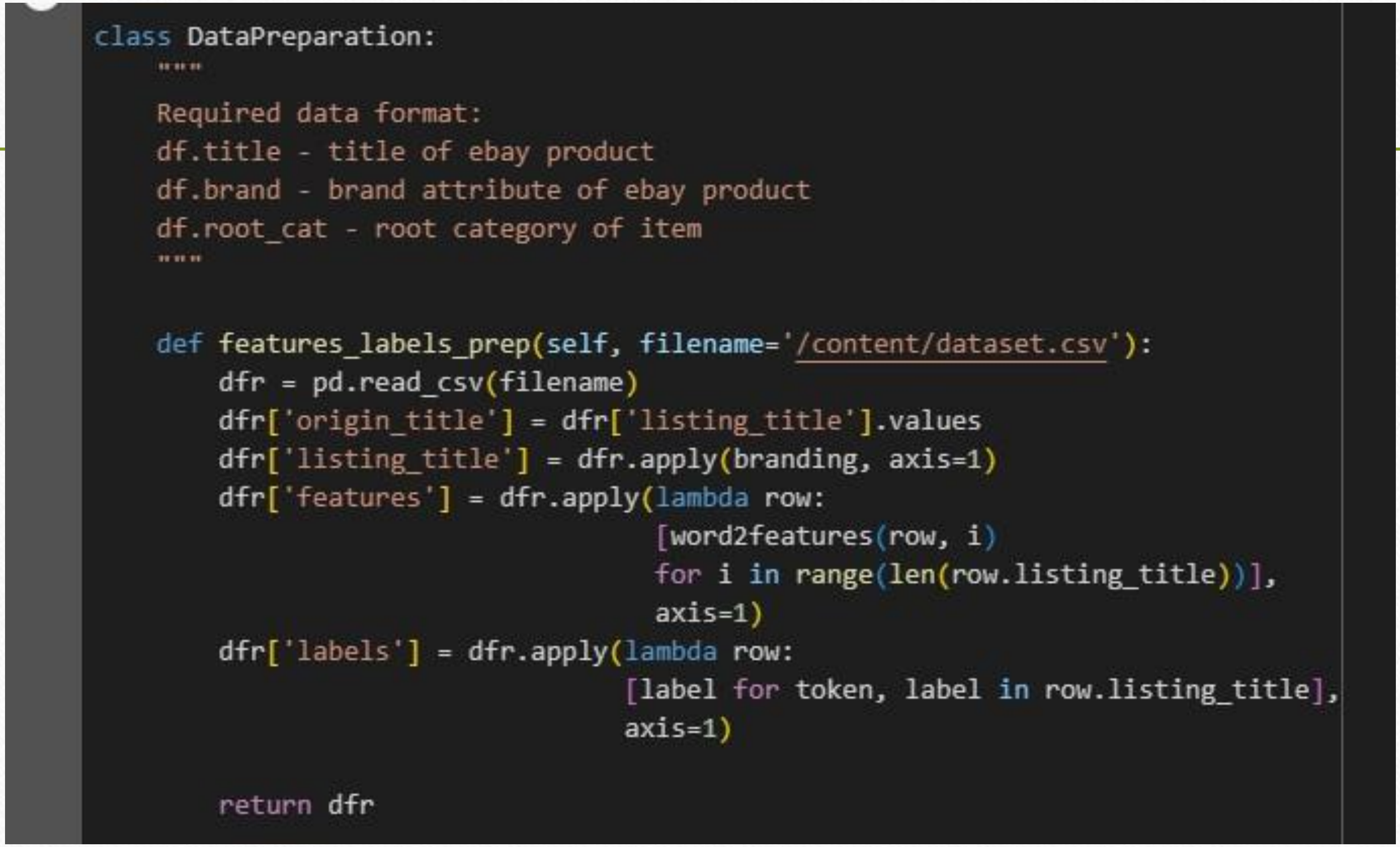


```python
def word2features(sent, i):
    word = str(sent.listing_title[i][0])
    features = {
        'root_cat': sent.brand_name,
        'bias': 1,
        'word_position': i,
        'word.lower()': word.lower(),
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
    }

    if i > 0:
        word1 = str(sent.listing_title[i - 1][0])
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
        })

    else:
        features['BOS'] = True

    if i < len(sent.listing_title) - 1:
        word1 = str(sent.listing_title[i + 1][0])
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
            '+1:word.anydigit': any(ch.isdigit() for ch in word1),
            '+1:word.ispunctuation': word1 in string.punctuation,
        })
```



```python
else:
    features['EOS'] = True

if i > 1:
    word1 = str(sent.listing_title[i - 1][0])
    word2 = str(sent.listing_title[i - 2][0])
    features.update({
        '-2:ngram': '{} {}'.format(word2, word1)
    })

if i < len(sent.listing_title) - 2:
    word1 = str(sent.listing_title[i + 1][0])
    word2 = str(sent.listing_title[i + 2][0])
    features.update({
        '+2:ngram': '{} {}'.format(word1, word2)
    })

return features
```

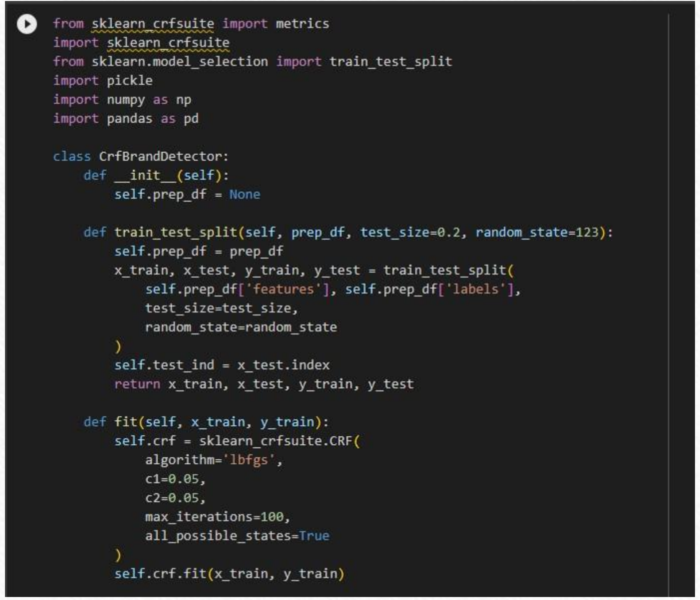
- Next we are going to define the class of the code to execute the function that we define before



```
class DataPreparation:
    """
    Required data format:
    df.title - title of ebay product
    df.brand - brand attribute of ebay product
    df.root_cat - root category of item
    """

    def features_labels_prep(self, filename='/content/dataset.csv'):
        dfr = pd.read_csv(filename)
        dfr['origin_title'] = dfr['listing_title'].values
        dfr['listing_title'] = dfr.apply(branding, axis=1)
        dfr['features'] = dfr.apply(lambda row:
                                    [word2features(row, i)
                                    for i in range(len(row.listing_title))],
                                    axis=1)
        dfr['labels'] = dfr.apply(lambda row:
                                  [label for token, label in row.listing_title],
                                  axis=1)

        return dfr
```

Defining the model

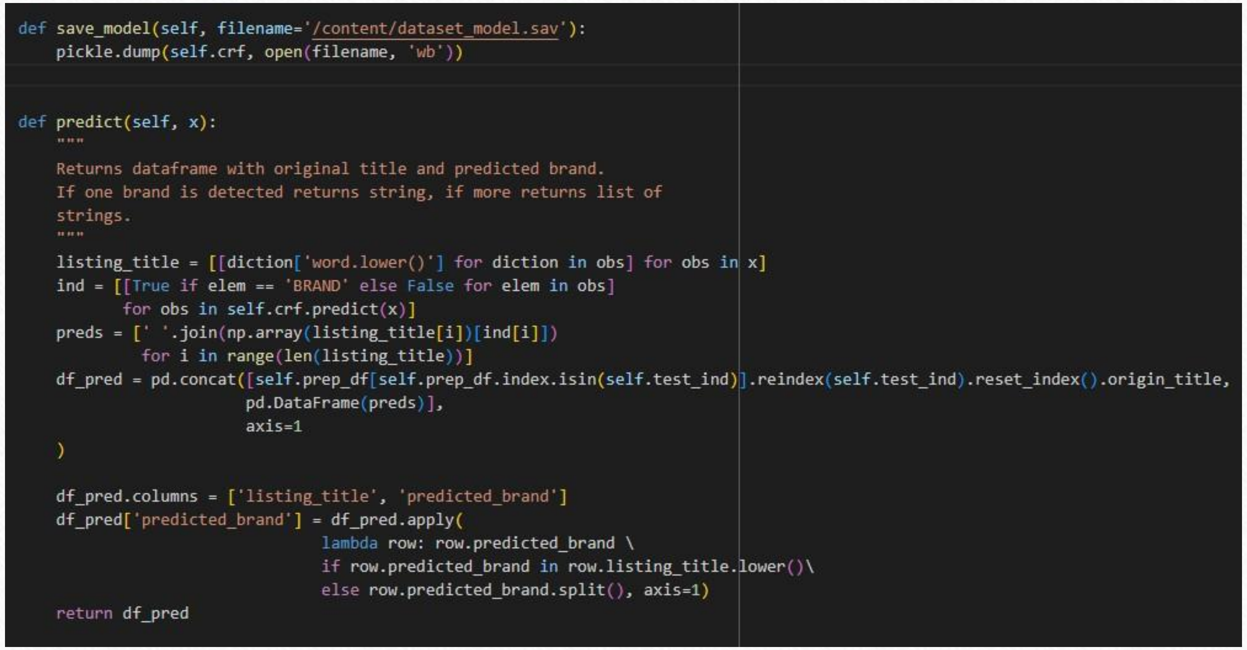
- Here we are going to use the sklearn_crfsuite to train our model, we also splitting our train and test data and also using lbfgs optimization to our model.



```python
from sklearn_crfsuite import metrics
import sklearn_crfsuite
from sklearn.model_selection import train_test_split
import pickle
import numpy as np
import pandas as pd

class CrfBrandDetector:
    def __init__(self):
        self.prep_df = None

    def train_test_split(self, prep_df, test_size=0.2, random_state=123):
        self.prep_df = prep_df
        x_train, x_test, y_train, y_test = train_test_split(
            self.prep_df['features'], self.prep_df['labels'],
            test_size=test_size,
            random_state=random_state
        )
        self.test_ind = x_test.index
        return x_train, x_test, y_train, y_test

    def fit(self, x_train, y_train):
        self.crf = sklearn_crfsuite.CRF(
            algorithm='lbfgs',
            c1=0.05,
            c2=0.05,
            max_iterations=100,
            all_possible_states=True
        )
        self.crf.fit(x_train, y_train)
```

Function for predicting the data

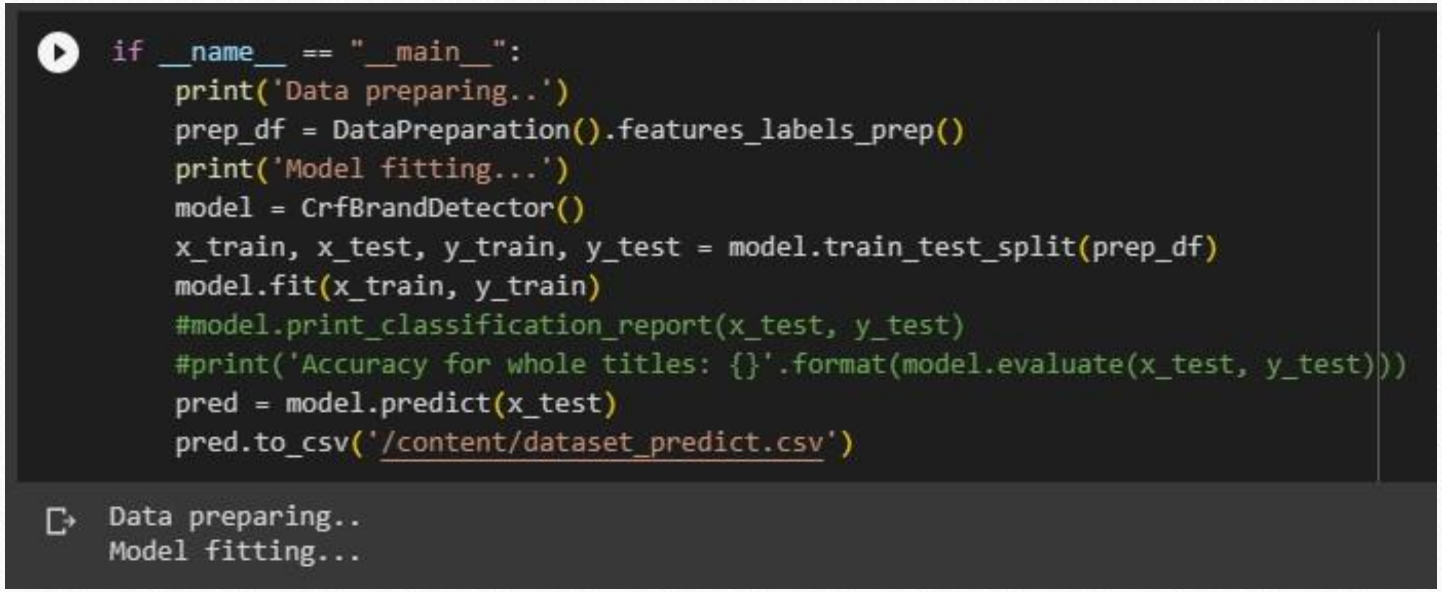
- The next step is to save and predict the model



```python
def save_model(self, filename='/content/dataset_model.sav'):
    pickle.dump(self.crf, open(filename, 'wb'))


def predict(self, x):
    """
    Returns dataframe with original title and predicted brand.
    If one brand is detected returns string, if more returns list of
    strings.
    """
    listing_title = [[diction['word.lower()'] for diction in obs] for obs in x]
    ind = [[True if elem == 'BRAND' else False for elem in obs]
           for obs in self.crf.predict(x)]
    preds = [' '.join(np.array(listing_title[i])[ind[i]])
             for i in range(len(listing_title))]
    df_pred = pd.concat([self.prep_df[self.prep_df.index.isin(self.test_ind)].reindex(self.test_ind).reset_index().origin_title,
                         pd.DataFrame(preds)],
                         axis=1
    )

    df_pred.columns = ['listing_title', 'predicted_brand']
    df_pred['predicted_brand'] = df_pred.apply(
                                 lambda row: row.predicted_brand \
                                 if row.predicted_brand in row.listing_title.lower()\
                                 else row.predicted_brand.split(), axis=1)
    return df_pred
```

Lets run the code

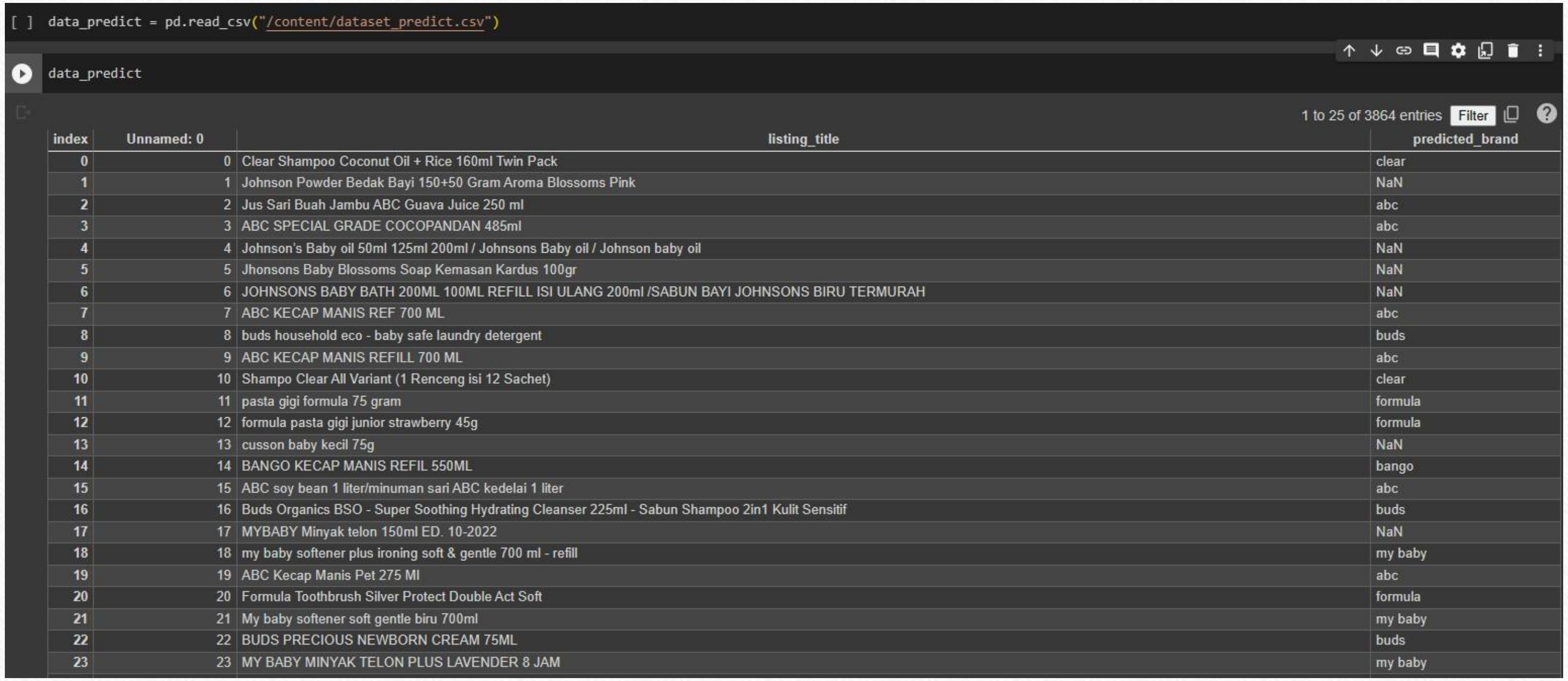
- After we define class and function the dataset preparation, training model, saving model and predict dataset. We can execute all of them just like this



```
if __name__ == "__main__":
    print('Data preparing..')
    prep_df = DataPreparation().features_labels_prep()
    print('Model fitting...')
    model = CrfBrandDetector()
    x_train, x_test, y_train, y_test = model.train_test_split(prep_df)
    model.fit(x_train, y_train)
    #model.print_classification_report(x_test, y_test)
    #print('Accuracy for whole titles: {}'.format(model.evaluate(x_test, y_test)))
    pred = model.predict(x_test)
    pred.to_csv('/content/dataset_predict.csv')
```

```
Data preparing..
Model fitting...
```

Here is the result



```
[ ] data_predict = pd.read_csv("/content/dataset_predict.csv")
```

```
data_predict
```

1 to 25 of 3864 entries Filter

index	Unnamed: 0	listing_title	predicted_brand
0	0	Clear Shampoo Coconut Oil + Rice 160ml Twin Pack	clear
1	1	Johnson Powder Bedak Bayi 150+50 Gram Aroma Blossoms Pink	NaN
2	2	Jus Sari Buah Jambu ABC Guava Juice 250 ml	abc
3	3	ABC SPECIAL GRADE COCOPANDAN 485ml	abc
4	4	Johnson's Baby oil 50ml 125ml 200ml / Johnsons Baby oil / Johnson baby oil	NaN
5	5	Jhonsons Baby Blossoms Soap Kemasan Kardus 100gr	NaN
6	6	JOHNSONS BABY BATH 200ML 100ML REFILL ISI ULANG 200ml /SABUN BAYI JOHNSONS BIRU TERMURAH	NaN
7	7	ABC KECAP MANIS REF 700 ML	abc
8	8	buds household eco - baby safe laundry detergent	buds
9	9	ABC KECAP MANIS REFILL 700 ML	abc
10	10	Shampo Clear All Variant (1 Renceng isi 12 Sachet)	clear
11	11	pasta gigi formula 75 gram	formula
12	12	formula pasta gigi junior strawberry 45g	formula
13	13	cusson baby kecil 75g	NaN
14	14	BANGO KECAP MANIS REFIL 550ML	bango
15	15	ABC soy bean 1 liter/minuman sari ABC kedelai 1 liter	abc
16	16	Buds Organics BSO - Super Soothing Hydrating Cleanser 225ml - Sabun Shampoo 2in1 Kulit Sensitif	buds
17	17	MYBABY Minyak telon 150ml ED. 10-2022	NaN
18	18	my baby softener plus ironing soft & gentle 700 ml - refill	my baby
19	19	ABC Kecap Manis Pet 275 Ml	abc
20	20	Formula Toothbrush Silver Protect Double Act Soft	formula
21	21	My baby softener soft gentle biru 700ml	my baby
22	22	BUDS PRECIOUS NEWBORN CREAM 75ML	buds
23	23	MY BABY MINYAK TELON PLUS LAVENDER 8 JAM	my baby

Summary

- The SGD Classifier has a good metrics score but also needed high ram.
- The CRF has a little RAM consume but the metric score still has lower score than SGD Classifier.
- The CRF is case sensitive because it is involving context of input.