```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/int3405-sentiment-analysis-problem/test.csv
/kaggle/input/int3405-sentiment-analysis-problem/full_train.csv
```

```
 !pip install tensorflow==2.0
```

```
Requirement already satisfied: tensorflow==2.0 in
/opt/conda/lib/python3.7/site-packages (2.0.0)
Requirement already satisfied: astor>=0.6.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (0.8.1)
Requirement already satisfied: wrapt>=1.11.1 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.12.1)
Requirement already satisfied: google-pasta>=0.1.6 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (0.2.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.19.5)
Requirement already satisfied: termcolor>=1.1.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.1.0)
Requirement already satisfied: absl-py>=0.7.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (0.15.0)
Requirement already satisfied: wheel>=0.26 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (0.37.1)
Requirement already satisfied: tensorboard<2.1.0,>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (2.0.2)
Requirement already satisfied: tensorflow-estimator<2.1.0,>=2.0.0
in /opt/conda/lib/python3.7/site-packages (from tensorflow==2.0)
```

```
(2.0.1)
Requirement already satisfied: grpcio>=1.8.6 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.32.0)
Requirement already satisfied: protobuf>=3.6.1 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (3.19.4)
Requirement already satisfied: keras-applications>=1.0.8 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.0.8)
Requirement already satisfied: six>=1.10.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.15.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (3.3.0)
Requirement already satisfied: gast==0.2.2 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (0.2.2)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/opt/conda/lib/python3.7/site-packages (from tensorflow==2.0) (1.1.2)
Requirement already satisfied: h5py in /opt/conda/lib/python3.7/site-
packages (from keras-applications>=1.0.8->tensorflow==2.0) (2.10.0)
Requirement already satisfied: setuptools>=41.0.0 in
/opt/conda/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (59.8.0)
Requirement already satisfied: werkzeug>=0.11.15 in
/opt/conda/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (2.2.2)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/opt/conda/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (0.4.6)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/opt/conda/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (1.35.0)
Requirement already satisfied: requests<3,>=2.21.0 in
/opt/conda/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (2.28.1)
Requirement already satisfied: markdown>=2.6.8 in
/opt/conda/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (3.3.7)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (4.2.4)
Requirement already satisfied: rsa<5,>=3.1.4 in
/opt/conda/lib/python3.7/site-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (4.8)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/opt/conda/lib/python3.7/site-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (0.2.7)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/opt/conda/lib/python3.7/site-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow==2.0)
(1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/opt/conda/lib/python3.7/site-packages (from markdown>=2.6.8-
```

```
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (4.13.0)
Requirement already satisfied: charset-normalizer<3,>=2 in
/opt/conda/lib/python3.7/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.7/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/lib/python3.7/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.7/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (1.26.12)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/opt/conda/lib/python3.7/site-packages (from werkzeug>=0.11.15-
>tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (2.1.1)
Requirement already satisfied: typing-extensions>=3.6.4 in
/opt/conda/lib/python3.7/site-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.1.0,>=2.0.0->tensorflow==2.0)
(3.7.4.3)
Requirement already satisfied: zipp>=0.5 in
/opt/conda/lib/python3.7/site-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.1.0,>=2.0.0->tensorflow==2.0) (3.8.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/opt/conda/lib/python3.7/site-packages (from pyasn1-modules>=0.2.1-
>google-auth<2,>=1.6.3->tensorboard<2.1.0,>=2.0.0->tensorflow==2.0)
(0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/opt/conda/lib/python3.7/site-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0-
>tensorflow==2.0) (3.2.0)
^C
ERROR: Operation cancelled by user
```

```
import os
!pip install pyvi
import numpy as np
with open('/kaggle/input/int3405-sentiment-analysis-problem/test.csv',
encoding="utf8") as f:
    lines = f.readlines()
    lines = "".join(lines)
```

```
Requirement already satisfied: pyvi in /opt/conda/lib/python3.7/site-
packages (0.1.1)
Requirement already satisfied: scikit-learn in
/opt/conda/lib/python3.7/site-packages (from pyvi) (1.0.2)
Requirement already satisfied: sklearn-crfsuite in
/opt/conda/lib/python3.7/site-packages (from pyvi) (0.3.6)
Requirement already satisfied: numpy>=1.14.6 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn->pyvi)
(1.19.5)
```

```
Requirement already satisfied: scipy>=1.1.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn->pyvi)
(1.7.3)
Requirement already satisfied: joblib>=0.11 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn->pyvi)
(1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn->pyvi)
(3.1.0)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-
packages (from sklearn-crfsuite->pyvi) (1.15.0)
Requirement already satisfied: tabulate in
/opt/conda/lib/python3.7/site-packages (from sklearn-crfsuite->pyvi)
(0.9.0)
Requirement already satisfied: python-crfsuite>=0.8.3 in
/opt/conda/lib/python3.7/site-packages (from sklearn-crfsuite->pyvi)
(0.9.8)
Requirement already satisfied: tqdm>=2.0 in
/opt/conda/lib/python3.7/site-packages (from sklearn-crfsuite->pyvi)
(4.64.0)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
```

```
#Ignoring the warnings
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns

#Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re, string, unicodedata
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalMaxPooling1D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import *
from tensorflow.keras import backend
from tensorflow.keras import layers
from sklearn.metrics import f1_score, confusion_matrix
import tensorflow as tf
from pyvi import ViTokenizer
from pyvi import ViUtils
```

Tập dữ liệu gồm 2 trường:

- Key: “comment”: các văn bản với nội dung đánh giá về quán ăn
- Key: “rating”: là Label cho đoạn văn bản đó có nội dung tích cực hay tiêu cực.

```
df1 =
pd.read_csv('/kaggle/input/int3405-sentiment-analysis-problem/full_train.csv')
df1 = df1.dropna()

df1 = df1.drop(['Unnamed: 0','RevId','UserId','image_urls'], axis=1)
X_train1 = list(df1['Comment'].values)
y_train1 = list(df1['Rating'].values)
print(df1.shape)
df1.head()
```

```
(9070, 2)
```

	Comment	Rating
0	Xôi dẻo, đô`ăn đậm vị. Hộp xôi được lót lá trô...	1.0
1	Gọi ship 1 xuất cari gà bánh naan và 3 miếng g...	0.0
2	Thời tiết lạnh như này, cả nhà rủ nhau đến leg...	1.0
3	Em có đọc review thấy mng bảo trà sữa nướng đề`...	0.0
4	Đô`ăn rất ngon, nhà hàng cũng rất đẹp, tất cả ...	1.0

Check DF

```
# check for length train
X_train = X_train1 #+ X_train2 + X_train3
print(len(X_train))
y_train = y_train1 #+ y_train2 + y_train3
print(len(y_train))
```

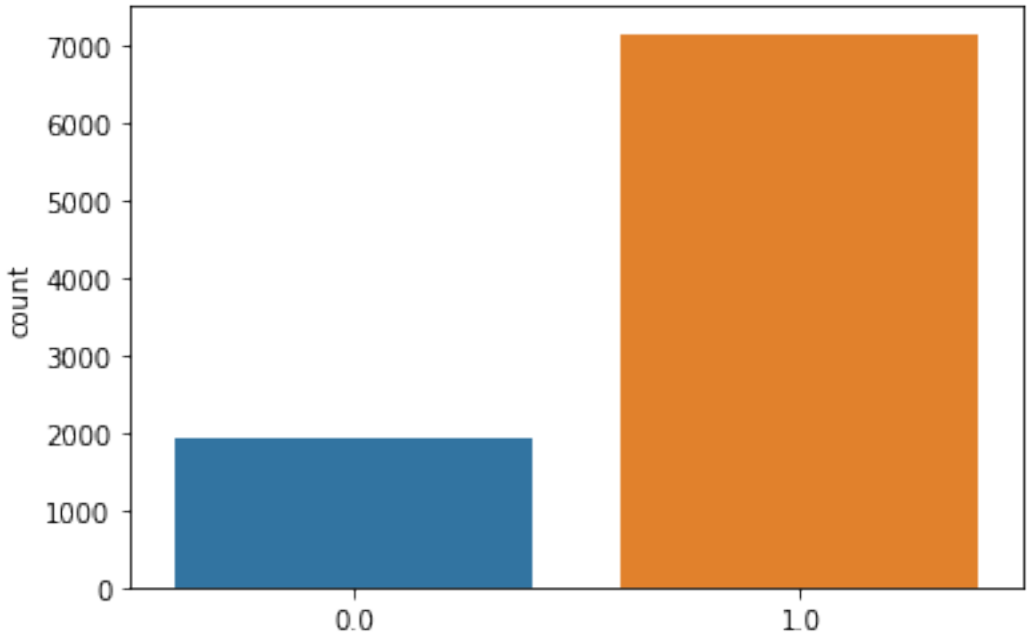
```
9070
9070
```

```
sum(y_train)
print(sum(y_train) / len(y_train))
```

```
0.7878721058434399
```

```
# Check chart output data
sns.countplot(y_train)
```

```
<AxesSubplot:ylabel='count'>
```



Text Preprocessing

```
def clean_text_support(text):
    RE_EMOJI = re.compile('[\U00010000-\U0010ffff]', flags=re.UNICODE)
    text = re.sub(r"<.*?>", " ", text)
    text = re.sub(r"\n", " ", text)
    text = re.sub(r"\s{2,}", " ", text)
    text = RE_EMOJI.sub(r'', text)
    return text.strip().lower()

def clean_text(X):
    processed = []
    for text in X:
        text = clean_text_support(text)
        text = ViTokenizer.tokenize(text)
        processed.append(text)
    return processed

text_test = X_train[100]
print(text_test)
print('===')
print(X_train[333])

X_train_final = X_train

len(X_train_final)

9070

print(X_train_final[:2])
print("===")
print(y_train[:2])
```

```
['Xôi dẻo, đô`ăn đậm vị. Hộp xôi được lót lá trông rất thích', 'Gọi
ship 1 xuất cari gà bánh naan và 3 miếng gà nướng(được tặng 1 coca).
Đô`ăn khá ngon, tổng 210k được giảm 50k còn 160k. Tuy nhiên gọi 3
miếng gà thì thiếu 1 miếng, mà kể cả đó đủ ba miếng thì khẩu phần vẫn
là quá ít so với giá 120k 1 suất.']
===
[1.0, 0.0]
```

```
# Attention Layer
from tensorflow.keras import initializers,regularizers,constraints
from tensorflow.keras import backend as K
class AttentionWithContext(tf.keras.layers.Layer):
    """
    Attention operation, with a context/query vector, for temporal
data.
    Supports Masking.
    Follows the work of Yang et al.
[https://www.cs.cmu.edu/~diyiy/docs/naacl16.pdf]
    "Hierarchical Attention Networks for Document Classification"
    by using a context vector to assist the attention
    # Input shape
        3D tensor with shape: `(samples, steps, features)`.
    # Output shape
        2D tensor with shape: `(samples, features)`.
    How to use:
    Just put it on top of an RNN Layer (GRU/LSTM/SimpleRNN) with
return_sequences=True.
    The dimensions are inferred based on the output shape of the RNN.
    Note: The layer has been tested with Keras 2.0.6
    Example:
        model.add(LSTM(64, return_sequences=True))
        model.add(AttentionWithContext())
        # next add a Dense layer (for classification/regression) or
whatever...
    """

    def __init__(self, W_regularizer=None, u_regularizer=None,
b_regularizer=None,
                 W_constraint=None, u_constraint=None,
b_constraint=None,
                 bias=True, **kwargs):

        self.supports_masking = True
        self.init = initializers.get('glorot_uniform')

        self.W_regularizer = regularizers.get(W_regularizer)
        self.u_regularizer = regularizers.get(u_regularizer)
        self.b_regularizer = regularizers.get(b_regularizer)

        self.W_constraint = constraints.get(W_constraint)
```

```
        self.u_constraint = constraints.get(u_constraint)
        self.b_constraint = constraints.get(b_constraint)

        self.bias = bias
        super(AttentionWithContext, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 3

        self.W = self.add_weight(shape=(input_shape[-1], input_shape[-
1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        if self.bias:
            self.b = self.add_weight(shape=(input_shape[-1],),
                                     initializer='zero',
                                     name='{}_b'.format(self.name),
                                     regularizer=self.b_regularizer,
                                     constraint=self.b_constraint)

        self.u = self.add_weight(shape=(input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_u'.format(self.name),
                                 regularizer=self.u_regularizer,
                                 constraint=self.u_constraint)

        super(AttentionWithContext, self).build(input_shape)

    def compute_mask(self, input, input_mask=None):
        # do not pass the mask to the next layers
        return None
    def get_config(self):
        config = super().get_config().copy()
        config.update({
            'W_regularizer': self.W_regularizer,
            'u_regularizer': self.u_regularizer,
            'b_regularizer': self.b_regularizer,
            'W_constraint': self.W_constraint,
            'u_constraint': self.u_constraint,
            'b_constraint': self.b_constraint,
            'bias': self.bias,
            })
        return config
    def call(self, x, mask=None):
        uit = dot_product(x, self.W)

        if self.bias:
```

```
            uit += self.b

        uit = K.tanh(uit)
        ait = dot_product(uit, self.u)

        a = K.exp(ait)

        # apply mask after the exp. will be re-normalized next
        if mask is not None:
            # Cast the mask to floatX to avoid float64 upcasting in
theano
            a *= K.cast(mask, K.floatx())

        # in some cases especially in the early stages of training the
sum may be almost zero
        # and this results in NaN's. A workaround is to add a very
small positive number ε to the sum.
        # a /= K.cast(K.sum(a, axis=1, keepdims=True), K.floatx())
        a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(),
K.floatx())

        a = K.expand_dims(a)
        weighted_input = x * a
        return K.sum(weighted_input, axis=1)

    def compute_output_shape(self, input_shape):
        return input_shape[0], input_shape[-1]

def dot_product(x, kernel):
    """
    Wrapper for dot product operation, in order to be compatible with
both
    Theano and Tensorflow
    Args:
        x (): input
        kernel (): weights
    Returns:
    """
    if K.backend() == 'tensorflow':
        return K.squeeze(K.dot(x, K.expand_dims(kernel)), axis=-1)
    else:
        return K.dot(x, kernel)

# some properties
vocab_size = 60000
maxlen = 250
encode_dim = 20
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train_final)
tokenized_word_list = tokenizer.texts_to_sequences(X_train_final)
```

```
X_train_padded = pad_sequences(tokenized_word_list, maxlen = maxlen,
padding='post')

#EarlyStopping and ModelCheckpoint
es = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1,
patience = 5)
mc = ModelCheckpoint('model_best.h5', monitor = 'val_loss', mode =
'min', verbose = 1, save_best_only = True)
```

Building model train

```
import tensorflow_addons as tfa

# Build model
def create_model():
    model = Sequential()
    embed = Embedding(input_dim = vocab_size, output_dim = 20,
input_length = X_train_padded.shape[1])
    model.add(embed)
    model.add(Dropout(0.4))
    model.add(Bidirectional(LSTM(200, return_sequences = True)))
    model.add(Dropout(0.3))
    model.add(AttentionWithContext())
    model.add(Dropout(0.3))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation = 'sigmoid'))
    model.summary()
    return model
```

Traning model

```
from sklearn.model_selection import train_test_split
X_train_padded = np.asarray(X_train_padded)
y_train = np.asarray(y_train)
X_train_final2 = X_train_padded
y_train_final2 = y_train
weight = sum(y_train_final2) / y_train_final2.shape[0]

#class weight
weight_for_0 = (1 / (1-(weight))) * 0.5
weight_for_1 = (1 / (weight))* 0.5
class_weight = {0: weight_for_0, 1: weight_for_1}

es = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1,
patience = 5)
mc = ModelCheckpoint('model_best.h5', monitor = 'f1_score', mode =
'min', verbose = 1, save_best_only = True)
batch_size= 300
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='f1_score',
```

```
factor=0.2,
                                patience=3, min_lr=1e-5,verbose = 1)

tpu_strategy = tf.distribute.MirroredStrategy()

# instantiating the model in the strategy scope creates the model on
the TPU
with tpu_strategy.scope():
    model = create_model() # define your model normally
    optim =  tf.keras.optimizers.Adam(learning_rate=5e-4)

    model.compile(optimizer='adam',
                      loss='binary_crossentropy',
                      metrics=['accuracy',
                               tf.keras.metrics.Recall(),
                               tfa.metrics.F1Score(num_classes=1,
                                                   average='micro')
                              ]
                )

len(X_train_final2)

train_x, train_y = np.array(X_train_final2), np.array(y_train_final2)
train_data = tf.data.Dataset.from_tensor_slices((train_x, train_y))

batch_size = 512 * tpu_strategy.num_replicas_in_sync

train_data = train_data.batch(batch_size)
# val_data = val_data.batch(batch_size)
# Disable AutoShard.
options = tf.data.Options()
options.experimental_distribute.auto_shard_policy =
tf.data.experimental.AutoShardPolicy.OFF
train_data = train_data.with_options(options)
# val_data = val_data.with_options(options)

#Fit model
model.fit(train_data ,
          epochs = 15, batch_size = batch_size, verbose = 1,
          callbacks = [reduce_lr, es,mc],class_weight=class_weight)

df =
pd.read_csv('/kaggle/input/int3405-sentiment-analysis-problem/test.csv
')
data_test = pd.DataFrame({'input':df['Comment'],'id':df["RevId"]})
X_test = list(data_test['input'].values)

def clean_text_test(X):
    idx = 0
    y_train = []
    processed = []
```

```
    for text in X:
        text = str(text)
        text = clean_text_support(text)
        input_text_pre_accent = ViTokenizer.tokenize(text)
        processed.append(input_text_pre_accent)
    return processed
# X_test = list()
X_test_final = clean_text_test(X_test)

my_submission['Rating'].sum() / len(my_submission['Rating'])

my_submission = pd.DataFrame({'RevId':
np.array(df["RevId"]).reshape(5103), 'Rating':
np.array(y_pred).reshape(5103)})
my_submission.to_csv('submission.csv', index=False)
```