```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import time
from collections import Counter

from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.feature_selection import chi2
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

from scipy import sparse
data_folder = '/kaggle/input/learn-ai-bbc/'

train_csv_path = data_folder + 'BBC News Train.csv'
test_csv_path = data_folder + 'BBC News Test.csv'
sample_solutions_path = data_folder + 'BBC News Sample Solution.csv'
class BBC:

    def __init__(self, train_csv_path):
        self.train_df,self.classes = self.preprocess_df()
        self.target = self.train_df['label'].values
        #self.feature_matrix,self.tfidf = self.build_feature_matrix()

    def preprocess_df(self):
        train_df = pd.read_csv(train_csv_path)
        columns_mapper = {'ArticleId': 'tid', 'Text': 'text', 'Category':
'label'}
        train_df = train_df.rename(columns=columns_mapper)
        label,classes = train_df['label'].factorize()
        train_df['label'] = label
        return train_df,classes

    def build_feature_matrix(self, encoding_method={'tfidf','bow'}):
        if encoding_method == 'tfidf':
            tfidf = TfidfVectorizer(stop_words='english', ngram_range=(1,2),
min_df=5)
            return tfidf.fit_transform(self.train_df['text'].values),tfidf

        elif encoding_method == 'bow':
            bow = CountVectorizer(stop_words='english', ngram_range=(1,2),
min_df=5)
            return bow.fit_transform(self.train_df['text'].values),bow
# Init object and show basic information
bbc = BBC(train_csv_path=train_csv_path)
print('Number of texts in the datset: {}'.format(bbc.train_df.shape[0]))
gb = bbc.train_df.groupby('label').count()
plt.bar(bbc.classes[gb.index], height=gb['tid'])
plt.title('Number of Texts of Each Class')
```

```
Number of texts in the datset: 1490
Text(0.5, 1.0, 'Number of Texts of Each Class')
```

```
class NMF(BBC):

    def NMF(self, feature_matrix):
        from sklearn.decomposition import NMF
        nmf = NMF(n_components=len(self.classes), random_state=2022)
        W = nmf.fit_transform(feature_matrix)
        H = nmf.components_
        return W,H,nmf

    def predict_train(self, W):
        pred_raw = np.argmax(W, axis=1)
        # the pred_raw order is not the same as self.classes
        mapped_indices = list()
        for i in range(len(self.classes)):
            mask = np.where(np.argmax(W, axis=1) == i)[0]
            mapped_cid = Counter(bbc.target[mask]).most_common(1)[0][0] #use
the most common as mapping
            mapped_indices.append(mapped_cid)
        # convert pred_raw to ordered prediction
        y_pred = [mapped_indices[raw_idx] for raw_idx in pred_raw]
        return y_pred,mapped_indices

    def predict_test(self, What, mapped_indices):
        y_pred_raw = np.argmax(What, axis=1)
        return [mapped_indices[i] for i in y_pred_raw]


# prepare fm, encoder and test_features
bbc = NMF(train_csv_path=train_csv_path)
fm,encoder = bbc.build_feature_matrix(encoding_method='tfidf')
test_df = pd.read_csv(test_csv_path)
test_features = encoder.transform(test_df['Text'].values)

# init NMF
W,H,nmf = bbc.NMF(fm)
print('W shape: {}'.format(W.shape))
print('H shape: {}'.format(H.shape))

# predicting training dataset
y_pred,mapped_indices = bbc.predict_train(W)
acc = accuracy_score(bbc.target, y_pred)
print('Accuracy on the training set: {:.4f}'.format(acc))

# predicting test dataset
What = nmf.transform(test_features)
y_pred = bbc.predict_test(What, mapped_indices)
/opt/conda/lib/python3.7/site-packages/sklearn/decomposition/_nmf.py:294:
FutureWarning: The 'init' value, when 'init=None' and n_components is less
than n_samples and n_features, will be changed from 'nndsvd' to 'nndsvda' in
1.1 (renaming of 0.26).
  FutureWarning,
W shape: (1490, 5)
H shape: (5, 9927)
```

```
Accuracy on the training set: 0.9195
# Replace submission df
submission = pd.read_csv(sample_solutions_path)
submission['Category'] = [bbc.classes[i] for i in y_pred]
submission.to_csv('submission.csv', index=False)
submission
```

	ArticleId	Category
0	1018	sport
1	1319	tech
2	1138	sport
3	459	business
4	1020	sport
...	...	...
730	1923	business
731	373	entertainment
732	1704	tech
733	206	business
734	471	politics

735 rows × 2 c