Brand Classification (NLP)

Description :

This document is a short summary of an approach for classifying text snippets retrieved from websites into their respective brands to which the web page belongs to. This solution is submission for coding exercises for Machine Learning Engineer at Bolster AI. This is an unstructured data problem where patterns are extracted from high dimensional non linear data which are used to train a classification ML model using supervised learning algorithm. Next sections provide an overview of the various steps required to solve the text classification problem, followed by results.

Data Exploration :

Exploring the data helps in gaining insights into the high level details of the given dataset and draw some inferences about the appropriate training pipeline required to solve the problem.

1. **Basic check for dataset** : We check for dataset size, the number of unique labels and average length of text size. The dataset provided in JSON format is first converted to pandas dataframe

```python
import json
import pandas as pd

with open('exercise_data.json', 'r') as f:
    data = json.load(f)

df = pd.json_normalize(data)
classes = df["brand"].unique()
textLengths = [len(text.split(" ")) for text in df["raw_text"]]
print("Number of data points : {}".format(len(df)))
print("Number of classes  : {}".format(len(classes)))
print("Classes names : ", classes)
print("Average length of text : ", sum(textLengths)/len(textLengths))
```

✓ 0.2s

```
Number of data points : 3662
Number of classes  : 10
Classes names :  ['chase' 'microsoft' 'costco' 'netflix' 'opensea' 'steam' 'walmart'
 'discover' 'tesco' 'lego']
Average length of text :  603.3601856908793
```

2. **Data distribution across classes** : Another important source of dataset inference is checking how the data points are distributed across the different classes. This is an important check as the dataset may be severely imabalanced and steps need to be taken to ensure the training process is not affected due to this imbalance. For the given dataset, it was found that the class imbalance is severe, the extent of the imbalance is so high that the class "lego" has only one data point in entire dataset.

```
pd.value_counts(df['brand'])
```

✓ 0.2s

```
costco       1405
microsoft     809
opensea       442
chase         380
netflix       356
steam         170
walmart        87
tesco           8
discover        4
lego            1
Name: brand, dtype: int64
```

3. **Character level analysis** : In this analysis, we investigate the number of spaces and non-english word in the text corpus of each data point. This is important and spaces and indecipherable character which do not belong to english langauge vocabulary do not help in enhancing the generalizing capability of the machine learning model. These characters usually need to be filtered out, and this process is called data cleaning. This not only reduces the data dimensionality but also preserve the important patterns which have high correlation to the classification output.

```
from nltk.corpus import stopwords

text = list(df['raw_text'])
spacesCount = 0
nonEnglishWordCount = 0
for i in range(len(text)):
    words = text[i].split(" ")
    spacesCount += len(words)
    nonEnglishWordCount += len([word for word in words if word in stopwords.words('english')])

print("Average number of spaces : {}".format(spacesCount/len(text)))
print("Average number of non english words : {}".format(nonEnglishWordCount/len(text)))
```

✓ 1m 40.8s

```
Average number of spaces : 603.3601856908793
Average number of non english words : 58.723102129983616
```

This step indicates that before any training, the data needs to be cleaned to remove unwanted features.

Algorithm Steps:

1. **Load Dataset** : The dataset is given in json format, it needs to be converted to pandas dataframe format. This can be done by the json_normalize method of pandas API.
2. **Preprocess Data** : As mentioned in the data analysis section, the dimensionality of the data can be reduced significantly by removing the characters and words from the data that do not contribute to the learning of the algorithm. In this approach, for the preprocessing step, firstly all the characters in each datapoint's text body is converted to lowercase. Next, all non alphabetic characters are removed from each text body. Finally, all the white spaces are also removed from the text body. This helps model to converge faster and reduces the training resource requirements.
3. **Feature extraction** : In this step, feature engineering is performed to use Bag of Words approach to learn vocabulary from the given dataset. Intuitively, this step selects the most prominent words in the text corpus to be selected to form the vocabulary as they correlate with approximation of the output class more than the rarely occuring words which do not contribute towards learning. This vocabulary essentially becomes a mapping of words in the text to numerical value which can the form a feature vector suitable to be processed by a machine learning algorithm. Feature extraction also involves the Oversampling step to

address the class imbalance problem mentioned in data exploration step. Oversampling allows for increase in the representation of the under-represented labels by including duplicate samples.

4. **Ensemble Learning** : In this step we use the Random Forest Classification technique to implement supervised learning on this dataset. Random Forest Classifier uses an ensemble of decision trees and randomly samples segments of the dataset to populate these decision trees, finally it uses a voting mechanism from all these trees to predict the majority class as the output. This technique of sampling random dataset segments is known as bagging or bootstrap aggregrating and it addresses the problem of overfitting which arises out of class imbalance. In this implementation a Random Forest Classifier of 1000 decision trees is used to perform ensemble learning. This model took about 6 seconds to train.
5. **Performance evaluation** : This step is critical to measure the performance of the trained model on test dataset. In classification problems, accuracy is usually not a great metric to evaluate the performance of the model, especially if the class imbalance is severe. Since this is a classification problem with high class imbalance, the best metrics to evalaute the performance of the model are Precision and Recall. Precision is a measure of how model performs on positive predictions, whereas Recall is the measure of the number of positive class predictions made out of all positive examples in the dataset. F1 Score is another more advanced metric that takes into account both Precision and Recall and is even more robust metric. This implementation evaluates the model using the F1 score and also depicts the performance of the model on test set using the confusion matrix.

Following is the classification report for each class for the test dataset which depicts the precision, recall and F1 Score:

```
Accuracy:
              precision    recall  f1-score   support

       chase       1.00      0.96      0.98        78
      costco       1.00      1.00      1.00       279
        lego       0.00      0.00      0.00         1
   microsoft       0.95      1.00      0.98       144
     netflix       1.00      0.96      0.98        77
     opensea       0.97      1.00      0.98        95
       steam       1.00      0.92      0.96        39
       tesco       0.50      0.50      0.50         2
     walmart       0.94      0.94      0.94        18

    accuracy                           0.98       733
   macro avg       0.82      0.81      0.81       733
weighted avg       0.98      0.98      0.98       733
```

Following is the confusion matrix for each class for the test dataset:

```
Confusion Matrix:
 [[ 75   0   0   2   0   1   0   0   0]
 [  0 278   0   0   0   0   0   1   0]
 [  0   0   0   0   0   0   0   0   1]
 [  0   0   0 144   0   0   0   0   0]
 [  0   0   0   3  74   0   0   0   0]
 [  0   0   0   0   0  95   0   0   0]
 [  0   0   0   1   0   2  36   0   0]
 [  0   0   0   1   0   0   0   1   0]
 [  0   1   0   0   0   0   0   0  17]]
```

It can be observed that for severly under-represented classes like "lego" and "tesco", model is very prone to making errors even after oversampling. This can be resolved by collecting more data for these classes or dropping these classes from the learning algorithm.

6. **Serving the model** : Once trained, the model is ready to be deployed on a server to handle incoming request for real world data. This server can be created using the Flask library of python. This server is capable of receiving HTTP requests using POST method with a text body to classify it into the target class.

Production and Deployment:

1. **Deployment** : Although the model is deployed on Flask in the previous step, in a production environment, it would be prudent to wrap the application in a container which can then run in a docker environment on a Kubernetes server. Docker allows the packaged isolation of code for an app along with its dependencies by sharing the exisiting resources on a system. Additionally Kubernetes ecosystem allows for seamless scaling up and down of machine learning applications along with ability to deploy on multiple cloud networks.
2. **Maintenance** : A machine learning application's performance usually degrades over time. This happens due to data drift and concept drift. Concept drift occurs when the mapping between the inputs and outputs which the model was trained on changes, for example in this case if a new class of brand is introduced on the data. Data drift on the other hand occurs when the underlying data patterns change however the mappings between input and output still hold. Both these process eventually downgrade model performance and need to be detected and addressed. While concept drift is easier to detect, detecting the data drift is more difficult. Usually, data drift is detected by studying the changes in the statistical distributions of the dataset.

   For the given problem, we can plot the statistical distribution of various aspects of the dataset, these can be length of text, number of out of vocabulary words. Data drift is said to occur when the distribution of production dataset loses it similarity with the dataset the model has trained. This loss of similarity can be detected by conducting a statistical KS test of curve fitting for existing and the production dataset. If p-value of this test is close to zero, we can be assured that a data drift has occured.
3. **Continous Training** : Once a data drift is detected the problem of performance degradation due to this drift needs to be addressed. The production data no longer shares the same statistical distribution as the training data, hence this is a good time to label the production data, integrate with existing training data, re-calculate the statistical distribution and re-train the model on the enhanced dataset. Re-training can be a scheduled weekly or

bi-weekly event or trigger based event.