ML Pipeline Preparation

March 4, 2019

1 ML Pipeline Preparation

Follow the instructions below to help you create your ML pipeline. ### 1. Import libraries and load data from database. - Import Python libraries - Load dataset from database with `read_sql_table` - Define feature and target variables X and Y

In [19]:
```
import nltk

nltk.download(['punkt', 'wordnet'])
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

Out[19]: True

In [79]:
```
# import libraries
import pandas as pd
import numpy as np

from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputClassifier
from sklearn.pipeline import Pipeline

from sqlalchemy import create_engine
```

In [22]:
```
# load data from database
engine = create_engine('sqlite:///DisasterResponses.db')
df = pd.read_sql_table('Messages', engine)
X = df['message'].values
Y = df.drop(['id','message','original','genre'], axis=1)
```

```
In [34]: Y.columns
```

```
Out[34]: Index(['related', 'request', 'offer', 'aid_related', 'medical_help',
                'medical_products', 'search_and_rescue', 'security', 'military',
                'child_alone', 'water', 'food', 'shelter', 'clothing', 'money',
                'missing_people', 'refugees', 'death', 'other_aid',
                'infrastructure_related', 'transport', 'buildings', 'electricity',
                'tools', 'hospitals', 'shops', 'aid_centers', 'other_infrastructure',
                'weather_related', 'floods', 'storm', 'fire', 'earthquake', 'cold',
                'other_weather', 'direct_report'],
               dtype='object')
```

1.0.1 2. Write a tokenization function to process your text data

```
In [17]: def tokenize(text):
             tokens = word_tokenize(text)

             lemmatizer = WordNetLemmatizer()

             clean_tokens = []
             for tok in tokens:
                 clean_tok = lemmatizer.lemmatize(tok).lower().strip()
                 clean_tokens.append(clean_tok)

             return clean_tokens
```

1.0.2 3. Build a machine learning pipeline

This machine pipeline should take in the `message` column as input and output classification results on the other 36 categories in the dataset. You may find the MultiOutputClassifier helpful for predicting multiple target variables.

```
In [11]: pipeline = Pipeline([
             ('vect', CountVectorizer(tokenizer=tokenize)),
             ('tfidf', TfidfTransformer()),
             ('clf', MultiOutputClassifier(RandomForestClassifier()))
         ])
```

1.0.3 4. Train pipeline

- Split data into train and test sets
- Train pipeline

```
In [69]: X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state=42, test_size=0.

         pipeline.fit(X_train, y_train)

         y_pred = pd.DataFrame(pipeline.predict(X_test))
```

1.0.4 5. Test your model

Report the f1 score, precision and recall for each output category of the dataset. You can do this by iterating through the columns and calling sklearn's `classification_report` on each.

```
In [107]: classification_report(y_test.values, y_pred, target_names=y_test.columns)
```

```
    ---------------------------------------------------------------------------

    ValueError                                Traceback (most recent call last)

    <ipython-input-107-6bbeaafcf7db> in <module>()
----> 1 classification_report(y_test.values, y_pred, target_names=y_test.columns)


    /opt/conda/lib/python3.6/site-packages/sklearn/metrics/classification.py in classificati
   1419
   1420     if labels is None:
-> 1421         labels = unique_labels(y_true, y_pred)
   1422     else:
   1423         labels = np.asarray(labels)


    /opt/conda/lib/python3.6/site-packages/sklearn/utils/multiclass.py in unique_labels(*ys)
     95     _unique_labels = _FN_UNIQUE_LABELS.get(label_type, None)
     96     if not _unique_labels:
---> 97         raise ValueError("Unknown label type: %s" % repr(ys))
     98
     99     ys_labels = set(chain.from_iterable(_unique_labels(y) for y in ys))


    ValueError: Unknown label type: (array([[1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 1, 1, 0],
       ...,
       [1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]]),       0   1   2   3   4   5   6   7   8   9  ...  26  27
0      0   0   0   0   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
1      1   0   0   0   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
2      1   0   0   1   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
3      1   0   0   0   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
4      1   0   0   0   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
5      1   0   0   0   1   0   0   0   0   0 ...   0   0   0   0   0   0   0
6      0   0   0   0   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
7      1   0   0   0   0   0   0   0   0   0 ...   0   0   1   0   0   0   0
8      1   0   0   0   0   0   0   0   0   0 ...   0   0   1   0   0   0   1
9      1   0   0   0   0   0   0   0   0   0 ...   0   0   0   0   0   0   0
```

10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
11	1	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
14	1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
15	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
16	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
17	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	1	0	0
20	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	1
21	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
22	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
24	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
25	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
27	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
28	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
29	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
...	..	..	..	..	..	..	..	..	..	..	...	..	..	..	..	..	..	..
7835	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7836	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7837	1	0	0	1	0	0	0	0	0	0	...	0	0	1	0	0	0	1
7838	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0
7839	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7840	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	1
7841	1	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7842	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7843	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	1	0	0
7844	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7845	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7846	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7847	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7848	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7849	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7850	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7851	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7852	1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7853	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7854	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7855	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7856	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7857	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7858	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	1	0	0
7859	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7860	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7861	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

7862	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	1
7863	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
7864	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

	33	34	35
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0
26	0	0	0
27	0	0	0
28	0	0	0
29	0	0	0
...	..	..	..
7835	0	0	0
7836	0	0	0
7837	0	0	0
7838	0	0	0
7839	0	0	0
7840	0	0	0
7841	0	0	0
7842	0	0	0
7843	0	0	0
7844	0	0	0
7845	0	0	0
7846	0	0	0

```
7847   0   0   0
7848   0   0   0
7849   0   0   0
7850   0   0   0
7851   0   0   0
7852   0   0   1
7853   0   0   0
7854   0   0   0
7855   0   0   0
7856   0   0   0
7857   0   0   0
7858   0   0   0
7859   0   0   0
7860   0   0   0
7861   0   0   0
7862   0   0   0
7863   0   0   0
7864   0   0   0

[7865 rows x 36 columns])
```

1.0.5 6. Improve your model

Use grid search to find better parameters.

```
In [ ]: parameters =

        cv =
```

1.0.6 7. Test your model

Show the accuracy, precision, and recall of the tuned model.

Since this project focuses on code quality, process, and pipelines, there is no minimum performance metric needed to pass. However, make sure to fine tune your models for accuracy, precision and recall to make your project stand out - especially for your portfolio!

```
In [ ]:
```

1.0.7 8. Try improving your model further. Here are a few ideas:

- try other machine learning algorithms
- add other features besides the TF-IDF

```
In [ ]:
```

1.0.8 9. Export your model as a pickle file

```
In [ ]:
```

1.0.9 10. Use this notebook to complete `train.py`

Use the template file attached in the Resources folder to write a script that runs the steps above to create a database and export a model based on a new dataset specified by the user.

In [ ]: