Notebook

August 1, 2018

Contents

List of Figures

List of Tables

List of Codes

1 Incorporating Textual Features

Continuing with the Donor's Choose example, we will examine how to make use of the textual information in columns like the `project_essay_1` column.

```python
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
```

```python
d_train = pd.read_csv('data/train.csv')
```

```python
d_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 182080 entries, 0 to 182079
Data columns (total 16 columns):
id                                              182080 non-null object
teacher_id                                      182080 non-null object
teacher_prefix                                  182076 non-null object
school_state                                    182080 non-null object
project_submitted_datetime                      182080 non-null object
project_grade_category                          182080 non-null object
project_subject_categories                      182080 non-null object
project_subject_subcategories                   182080 non-null object
project_title                                   182080 non-null object
project_essay_1                                 182080 non-null object
project_essay_2                                 182080 non-null object
project_essay_3                                 6374 non-null object
project_essay_4                                 6374 non-null object
project_resource_summary                        182080 non-null object
teacher_number_of_previously_posted_projects    182080 non-null int64
project_is_approved                             182080 non-null int64
dtypes: int64(2), object(14)
memory usage: 22.2+ MB
```

```python
d_train.project_essay_1[0]
```

```
'Most of my kindergarten students come from low-income households and
are considered \\"at-risk\\". These kids walk to school alongside
their parents and most have never been further than walking distance
from their house. For 80% of my students, English is not their first
language or the language spoken at home. \\r\\n\\r\\nWhile my
kindergarten kids have many obstacles in front of them, they come to
school each day excited and ready to learn. Most students started the
year out never being in a school setting. At the start of the year
many had never been exposed to letters. Each day they soak up more
knowledge and try their hardest to succeed. They are highly motivated
to learn new things every day. We are halfway through the year and
they are starting to take off. They know know all letters, some sight
```

```
words, numbers to 20, and a majority of their letter sounds because of
 their hard work and determination. I am excited to see the places we
will go from here!'
```

```python
essay_sample = d_train.project_essay_1[0]
```

```python
for i in essay_sample[:10]:
    print(i)
```

```
M
o
s
t

o
f

m
y
```

```python
sent = ["This is a     very boring      example    . ",
       "Now maybe another.   ",
       "  Will this be different ? "]
```

```python
[string.strip() for string in sent]
```

```
['This is a     very boring      example    .',
 'Now maybe another.',
 'Will this be different ?']
```

```python
[string.replace('.', '') for string in sent]
```

```
['This is a     very boring      example     ',
 'Now maybe another   ',
 '  Will this be different ? ']
```

```python
[string.lower() for string in sent]
```

```
['this is a     very boring      example    . ',
 'now maybe another.   ',
 '  will this be different ? ']
```

```python
import re
```

```python
[re.sub(r"[aeiouAEIOU]", "X", string) for string in sent]
```

```
['ThXs Xs X    vXry bXrXng       XxXmplX     . ',
 'NXw mXybX XnXthXr.  ',
 '  WXll thXs bX dXffXrXnt ? ']
```

1.0.1 Problem

Using the `essay_sample` variable (first essay from first row of our Donor's Choose data), use the basic text strategies to do the following:

- remove any punctuation, if important to nature of word use (! vs. ?) determine a way to account for this.
- make sure all words are lowercase
- choose a few words that you believe to be the most important in the essay. Why did you choose these?

1.0.2 Tokenizing Text

```python
from sklearn.feature_extraction.text import CountVectorizer
```

```python
vect = CountVectorizer()
```

```python
vect.fit(sent)
```

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content
',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

```python
vect.vocabulary_
```

```
{'another': 0,
 'be': 1,
 'boring': 2,
 'different': 3,
 'example': 4,
 'is': 5,
 'maybe': 6,
 'now': 7,
 'this': 8,
 'very': 9,
 'will': 10}
```

```python
vect.transform(sent)
```

```
<3x11 sparse matrix of type '<class 'numpy.int64'>'
  with 12 stored elements in Compressed Sparse Row format>
```

```
bag = vect.transform(sent)
print("Content {}".format(repr(bag)))
```

```
Content <3x11 sparse matrix of type '<class 'numpy.int64'>'
  with 12 stored elements in Compressed Sparse Row format>
```

```
bag.toarray()
```

```
array([[0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0],
       [1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0],
       [0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1]])
```

```
vect.transform(sent)
```

```
<3x11 sparse matrix of type '<class 'numpy.int64'>'
  with 12 stored elements in Compressed Sparse Row format>
```

```
vect.vocabulary_
```

```
{'another': 0,
 'be': 1,
 'boring': 2,
 'different': 3,
 'example': 4,
 'is': 5,
 'maybe': 6,
 'now': 7,
 'this': 8,
 'very': 9,
 'will': 10}
```

```
sent
```

```
['This is a    very boring      example    . ',
 'Now maybe another.  ',
 '  Will this be different ? ']
```

```
example_slice = d_train[:50]
example_slice.head()
```

```
        id                        teacher_id teacher_prefix
school_state  \
0  p036502  484aaf11257089a66cfedc9461c6bd0a            Ms.
NV
1  p039565  df72a3ba8089423fa8a94be88060f6ed           Mrs.
GA
2  p233823  a9b876a9252e08a55e3d894150f75ba3            Ms.
UT
3  p185307  525fdbb6ec7f538a48beebaa0a51b24f            Mr.
NC
4  p013780  a63b5547a7239eae4c1872670848e61a            Mr.
CA

  project_submitted_datetime project_grade_category  \
0        2016-11-18 14:45:59          Grades PreK-2
1        2017-04-26 15:57:28             Grades 3-5
2        2017-01-01 22:57:44             Grades 3-5
3        2016-08-12 15:42:11             Grades 3-5
4        2016-08-06 09:09:11             Grades 6-8

            project_subject_categories  \
0                  Literacy & Language
1    Music & The Arts, Health & Sports
2  Math & Science, Literacy & Language
3                      Health & Sports
4                      Health & Sports

            project_subject_subcategories  \
0                              Literacy
1           Performing Arts, Team Sports
2  Applied Sciences, Literature & Writing
3                       Health & Wellness
4                       Health & Wellness

                                       project_title  \
0                            Super Sight Word Centers
1                              Keep Calm and Dance On
2                               Lets 3Doodle to Learn
3  \"Kid Inspired\" Equipment to Increase Activit...
4   We need clean water for our culinary arts class!

                                     project_essay_1  \
0  Most of my kindergarten students come from low...
1  Our elementary school is a culturally rich sch...
2  Hello;\r\nMy name is Mrs. Brotherton. I teach ...
3  My students are the greatest students but are ...
4  My students are athletes and students who are ...

                                     project_essay_2 project_essay_3
\
0  I currently have a differentiated sight word c...             NaN
1  We strive to provide our diverse population of...             NaN
2  We are looking to add some 3Doodler to our cla...             NaN
3  The student's project which is totally \"kid-i...             NaN
4  For some reason in our kitchen the water comes...             NaN

  project_essay_4                          project_resource_summary
\
0             NaN  My students need 6 Ipod Nano's to create and d...
1             NaN  My students need matching shirts to wear for d...
2             NaN  My students need the 3doodler. We are an SEM s...
3             NaN  My students need balls and other activity equi...
4             NaN  My students need a water filtration system for...

   teacher_number_of_previously_posted_projects  project_is_approved
0                                            26                    1
1                                             1                    0
2                                             5                    1
3                                            16                    0
```

```
4                                                  42                        1
```

```
vect.fit(example_slice.project_essay_1)
```

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content
',
        lowercase=True, max_df=1.0, max_features=None, min_df=5,
        ngram_range=(1, 1), preprocessor=None, stop_words='english',
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

```
X = vect.transform(example_slice.project_essay_1)
```

```
feature_names = vect.get_feature_names()
```

```
feature_names[:10]
```

```
['able',
 'academically',
 'achieve',
 'activities',
 'area',
 'autism',
 'backgrounds',
 'best',
 'breakfast',
 'challenges']
```

```
feature_names[-10:]
```

```
['title',
 'use',
 'variety',
 'want',
 'way',
 'work',
 'working',
 'world',
 'year',
 'years']
```

```
len(feature_names)
```

```
103
```

1.0.3 Using the essay as features

```
from sklearn.linear_model import LogisticRegression
```

```
X_train = X[:40]
X_test = X[-10:]
y_train = example_slice.project_is_approved[:40]
y_test = example_slice.project_is_approved[-10:]
```

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
0.7
```

```
clf = LogisticRegression()
params = {'C': [0.1, 1.0, 5.0, 10, 100, 1000]}
grid = GridSearchCV(clf, param_grid = params)
grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=
False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol
=0.0001,
          verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 1.0, 5.0, 10, 100, 1000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
grid.best_estimator_.score(X_test, y_test)
```

```
0.8
```

```
grid.best_params_
```

```
{'C': 0.1}
```

1.0.4 `min_df`

```
When building the vocabulary ignore terms that have a document     frequency strictly
lower than the given threshold. This value is also     called cut-off in the
literature.     If float, the parameter represents a proportion of documents, integer
absolute counts.     This parameter is ignored if vocabulary is not None.
```

```
vect = CountVectorizer(min_df=3)
vect.fit(example_slice.project_essay_1)
X = vect.transform(example_slice.project_essay_1)
```

```
X_train = X[:40]
X_test = X[-10:]
y_train = example_slice.project_is_approved[:40]
y_test = example_slice.project_is_approved[-10:]
```

```
clf = LogisticRegression()
params = {'C': [0.1, 1.0, 5.0, 10, 100, 1000]}
grid = GridSearchCV(clf, param_grid = params)
grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=
False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol
=0.0001,
          verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 1.0, 5.0, 10, 100, 1000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
grid.best_estimator_.score(X_test, y_test)
```

```
0.8
```

1.0.5 Stop Words

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
```

```
list(ENGLISH_STOP_WORDS)[:10]
```

```
['nothing',
 'are',
 'they',
 'above',
 'thru',
 'forty',
 'other',
 'eight',
 'ie',
 'third']
```

```
vect = CountVectorizer(min_df=5, stop_words='english')
```

```
vect.fit(example_slice.project_essay_1)
X = vect.transform(example_slice.project_essay_1)
```

```
X_train = X[:40]
X_test = X[-10:]
y_train = example_slice.project_is_approved[:40]
y_test = example_slice.project_is_approved[-10:]
```

```
clf = LogisticRegression()
params = {'C': [0.1, 1.0, 5.0, 10, 100, 1000]}
grid = GridSearchCV(clf, param_grid = params)
grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=
False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol
=0.0001,
          verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 1.0, 5.0, 10, 100, 1000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
grid.best_estimator_.score(X_test, y_test)
```

```
0.8
```

1.0.6 tf-idf

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
pipe = make_pipeline(TfidfVectorizer(), LogisticRegression())
```

```
tfi = TfidfVectorizer()
X = example_slice.project_essay_1
tfi.fit(X)
X_tfidf = tfi.transform(X)
```

```
X_tfidf.toarray()[:10]
```

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.
 ,
         0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.
 ,
         0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.
 ,
         0.17757928],
       ...,
       [0.        , 0.        , 0.        , ..., 0.25583394, 0.
 ,
         0.        ],
       [0.         , 0.         , 0.         , ..., 0.         , 0.
,
        0.        ],
       [0.         , 0.         , 0.         , ..., 0.         , 0.
,
        0.        ]])
```

```
max_vals = X_tfidf.max(axis=0).toarray().ravel()
```

```
feature_names = np.array(tfi.get_feature_names())
```

```
tfidf_sorted = max_vals.argsort()
```

```
feature_names[tfidf_sorted[:10]]
```

```
array(['00', 'american', 'since', 'shelving', 'round', 'asking', '
proper',
       'nthis', 'monday', 'look'], dtype='<U17')
```

```
feature_names[tfidf_sorted[-10:]]
```

```
array(['highly', 'physical', 'to', 'world', 'band', 'computer', 'town
',
       'very', 'me', 'delays'], dtype='<U17')
```

1.0.7 Problem

Use these features in a `LogisticRegression` model. Create a barplot of the top five and bottom five coefficients of the model and their feature name. What do these mean?

1.0.8 n-Grams

```
vect = CountVectorizer(ngram_range=(2,2))
```

```
ex = train_slice.project_essay_1[:5]
y = train_slice.project_is_approved[:5]
```

```
vect.fit(ex)
```

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content
',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(2, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

```
vect.get_feature_names()[:10]
```

```
['00 in',
 '20 and',
 '25 new',
 '30 minutes',
 '580 students',
 '5th grade',
 '80 of',
 '88 5th',
 '8th grade',
 '92 of']
```

```
pipe = make_pipeline(TfidfVectorizer(min_df = 3),
    ↪ LogisticRegression())
params = {'logisticregression__C': [0.1, 1.0, 5.0, 10.0, 50, 100,
    ↪ 1000], 'tfidfvectorizer__ngram_range': [(1,1), (1,2), (1,3)]}
```

```
grid = GridSearchCV(pipe, param_grid=params)
```

```
grid.fit(ex, y)
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_split.
py:605: Warning: The least populated class in y has only 2 members,
which is too few. The minimum number of members in any class cannot be
 less than n_splits=3.
  % (min_groups, self.n_splits)), Warning)
```

```
GridSearchCV(cv=None, error_score='raise',
       estimator=Pipeline(memory=None,
     steps=[('tfidfvectorizer', TfidfVectorizer(analyzer='word',
binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content
',
        lowercase=True, max_df=1.0, max_features=None, min_df=3,
        ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_i...
ty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False))]),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'logisticregression__C': [0.1, 1.0, 5.0, 10.0, 50,
100, 1000], 'tfidfvectorizer__ngram_range': [(1, 1), (1, 2), (1, 3)]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
grid.best_score_
```

```
0.6
```

```
grid.best_params_
```

```
{'logisticregression__C': 0.1, 'tfidfvectorizer__ngram_range': (1, 1)}
```

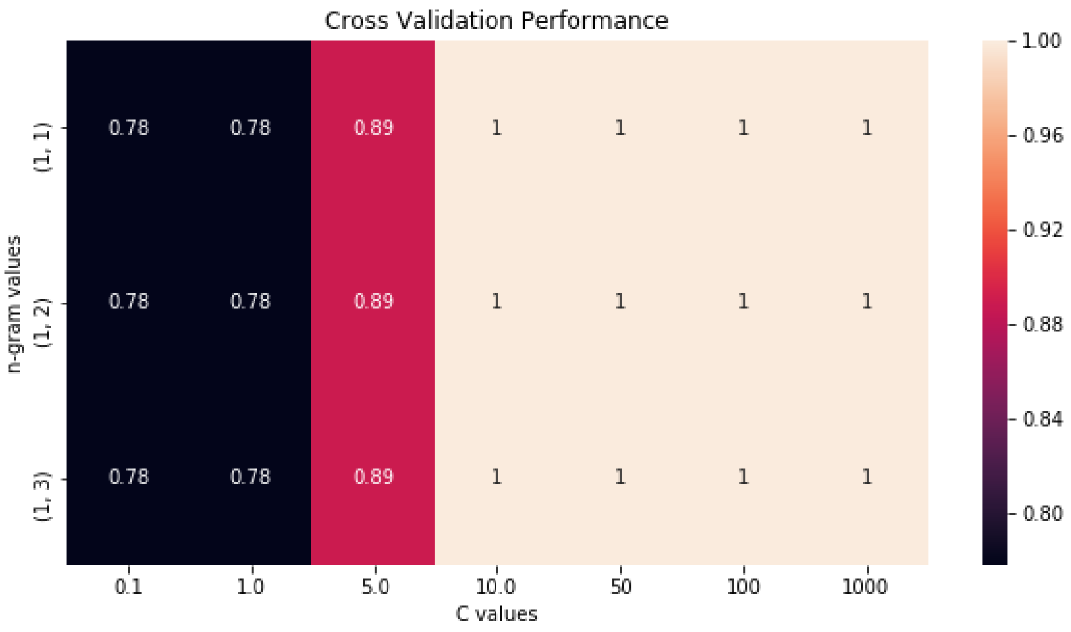
```
scores = grid.cv_results_['mean_train_score'].reshape(-1,3).T
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/utils/deprecation.py
:122: FutureWarning: You are accessing a training score ('
mean_train_score'), which will not be available by default any more in
 0.21. If you need training scores, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
```

```
import seaborn as sns
```

```
plt.figure(figsize = (10, 5))
sns.heatmap(scores, annot=True, xticklabels=params['
    ↪ logisticregression__C'], yticklabels=params['
    ↪ tfidfvectorizer__ngram_range'])
plt.xlabel('C values')
plt.ylabel('n-gram values')
plt.title('Cross Validation Performance')
```

```
Text(0.5,1,'Cross Validation Performance')
```


Cross Validation Performance
n-gram values
(1, 1)
(1, 2)
(1, 3)
C values