Feature Selection AZ

October 29, 2020

1 Goal

Take rows (all mal + twice num Benign)

SVM all features and find coeffs

take all +ve and -Ve coeffs

Sum by col

`take col if it appears a certain number of times`

Find num ben_cols and mal_cols.

`Balance them to an extent => based on lowest appearances`

2 Load Data and conv as NP Arrays

[1]:
```
import json
```

[2]:
```
with open("./DATA/apg-X.json", 'rt') as f:
    X_org = json.load(f)
with open("./DATA/apg-y.json", 'rt') as f:
    y_org = json.load(f)
with open("./DATA/apg-meta.json", 'rt') as f:
    meta_org = json.load(f)
```

[3]:
```
from sklearn.feature_extraction import DictVectorizer
```

[4]:
```
vec = DictVectorizer()
X_full = vec.fit_transform(X_org)
y_full = y_org
```

[5]:
```
import numpy as np
```

[6]:
```
y_full = np.asarray(y_full).reshape(-1,1)
```

[7]:
```
np.unique(y_full,return_counts=True)
```

[7]:
```
(array([0, 1]), array([135859,  15778], dtype=int64))
```

```
[8]: X_full.shape
```

```
[8]: (151637, 1537062)
```

```
[9]: meta_org = np.asarray(meta_org).reshape(-1,1)
```

3 Select Data

```
[10]: ben = []
      mal = []
      for i in range(len(y_full)):
          if y_full[i]==0:
              ben.append(i)
          else:
              mal.append(i)
      print(len(ben),len(mal))
```

```
135859 15778
```

```
[11]: import random
      ben_list = []
      ben_list = random.sample(ben,2*len(mal))
      print(len(ben_list))
```

```
31556
```

```
[12]: chosen = mal + ben_list
      chosen.sort()
      X = X_full[chosen]
      y = y_full[chosen]
      meta = meta_org[chosen]
      print(X.shape,y.shape,meta.shape)
      print(np.unique(y,return_counts=True))
```

```
(47334, 1537062) (47334, 1) (47334, 1)
(array([0, 1]), array([31556, 15778], dtype=int64))
```

4 SVM

```
[13]: from sklearn.model_selection import train_test_split
      import random
```

```
[14]: random_state = random.randint(0, 1000)
```

```
[15]: train_idxs, test_idxs = train_test_split(
                  range(X.shape[0]),
                  stratify=y,
                  test_size=0.33,
                  random_state=random_state)
```

[16]:
```
print(len(train_idxs),len(test_idxs))
```

```
31713 15621
```

[17]:
```
X_train1 = X[train_idxs]
X_test1 = X[test_idxs]
y_train1 = y[train_idxs]
y_test1 = y[test_idxs]
m_train1 = [meta[i] for i in train_idxs]
m_test1 = [meta[i] for i in test_idxs]
```

[18]:
```
X_train1.shape
```

[18]:
```
(31713, 1537062)
```

[19]:
```
y_train1.shape
```

[19]:
```
(31713, 1)
```

[20]:
```
from sklearn.svm import LinearSVC
```

[21]:
```
selector = LinearSVC(C=2)
selector.fit(X, y)
```

```
C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(**kwargs)
C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-packages\sklearn\svm\_base.py:977:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[21]:
```
LinearSVC(C=2)
```

[22]:
```
len(y)
```

[22]:
```
47334
```

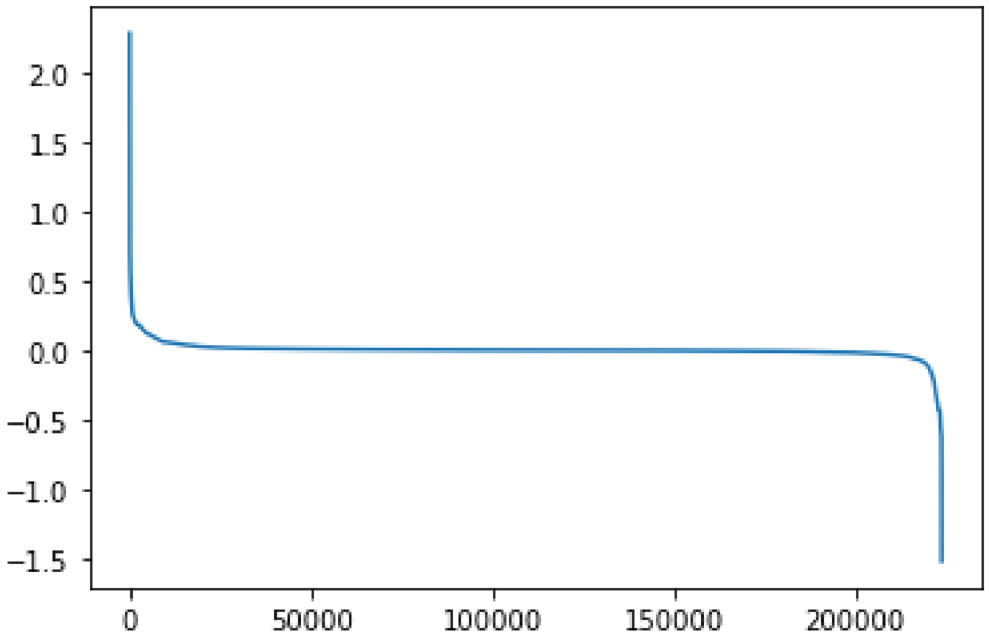
[23]:
```
cols1 = np.argsort(selector.coef_[0])[::-1]
p = n = z = 0
cols2 = []
```

```
for i in cols1:
    if selector.coef_[0][i] < 0:
        n+=1
        cols2.append(i)
    elif selector.coef_[0][i] > 0:
        p+=1
        cols2.append(i)
    else:
        z+=1
print(p,n,z)
print(len(cols2))
```

```
106665 117178 1313219
223843
```

[24]:
```
import matplotlib.pyplot as plt
plt.plot(selector.coef_[0][cols2])
```

[24]:
```
[<matplotlib.lines.Line2D at 0x2401c956a48>]
```



4.0.1 removing all cols from xtrain and xtest if they are 0 contrib

[25]:
```
X_train2 = X_train1[:,cols2]
X_test2 = X_test1[:,cols2]
y_train2 = y_train1
y_test2 = y_test1
```

```
m_train2 = m_train1
m_test2 = m_test1
coeff2 = selector.coef_[0][cols2]
print(X_train2.shape,X_test2.shape,len(coeff2))
coeff2
```

```
(31713, 223843) (15621, 223843) 223843
```

```
[25]: array([ 2.28352688,  2.15746016,  2.03675667, …, -1.51659126,
             -1.521736  , -1.52197476])
```

```
[26]: FC = X[:,cols2].sum(axis=0).reshape(-1,1)
      print(FC.shape)
      C = 0
      LOF = []
      for i in range(len(FC)):
          if FC[i][0] > 100: #guess
              C+=1
              LOF.append(i)
      print(len(LOF))
```

```
(223843, 1)
2948
```

```
[27]: X_train = X_train2[:,LOF]
      X_test = X_test2[:,LOF]
      y_train = y_train2
      y_test = y_test2
      m_train = m_train2
      m_test = m_test2
      coeff3 = coeff2[LOF]
      print(X_train.shape,X_test.shape,len(coeff3))
      coeff3
```

```
(31713, 2948) (15621, 2948) 2948
```

```
[27]: array([ 1.19514102,  1.1743613 ,  1.12972105, …, -1.16715654,
             -1.22070563, -1.40483014])
```

```
[28]: p = n = 0
      for i in coeff3:
          if i > 0:
              p+=1
          else:
              n+=1
      print(p,n)
```

```
1513 1435
```

Save these

HAD TO GO FOR SCIPY SPARSE ARRAY FOR CONVERTING TO ARRAY FOR SAVING X TRAIN N TEST

[29]:
```
import scipy
```

[30]:
```
X_train_S = scipy.sparse.csr_matrix.toarray(X_train)
X_test_S = scipy.sparse.csr_matrix.toarray(X_test)
```

[31]:
```
np.save('X_train.npy',X_train_S)
np.save('X_test.npy',X_test_S)
# scipy.sparse.save_npz('X_train.npz', X_train)
# scipy.sparse.save_npz('X_test.npz', X_test)
np.save('y_train.npy',y_train)
np.save('y_test.npy',y_test)
np.save('meta_train.npy',m_train)
np.save('meta_test.npy',m_test)
np.save('coeff_features.npy',coeff3)
```

[32]:
```
print("DONE")
```

```
DONE
```