Yahoo Troll Question Detection

Last Presentation

Trying Multiple Class Weights

We tried the following class weights in logistic regression

0: 0.9, 1: 2.1
0: 0.9, 1: 2
0: 0.9, 1: 1.9
0: 0.2, 1: 0.8
0: 0.25, 1: 0.75

Best Being

0: 0.9, 1: 2.1

The Best one had overall the least number of misclassifications

```
{0: 0.9, 1: 2.1}
[[182993   4465]
 [  4488   8054]]
{0: 0.9, 1: 2}
[[183170   4288]
 [  4617   7925]]
{0: 0.9, 1: 1.9}
[[183347   4111]
 [  4734   7808]]
{0: 0.2, 1: 0.8}
[[180777   6681]
 [  3526   9016]]
{0: 0.25, 1: 0.75}
[[182159   5299]
 [  4135   8407]]
{0: 0.3, 1: 0.7}
[[183118   4340]
 [  4729   7813]]
```

balanced

	precision	recall	f1-score	support
0	1.00	0.93	0.96	750672
1	0.49	0.98	0.65	49328
accuracy			0.94	800000
macro avg	0.74	0.95	0.81	800000
weighted avg	0.97	0.94	0.95	800000

{0: 0.9, 1: 2.1}

	precision	recall	f1-score	support
0	0.98	0.98	0.98	750672
1	0.71	0.74	0.73	49328
accuracy			0.97	800000
macro avg	0.85	0.86	0.85	800000
weighted avg	0.97	0.97	0.97	800000

{0: 0.9, 1: 2}

	precision	recall	f1-score	support
0	0.98	0.98	0.98	750672
1	0.71	0.73	0.72	49328
accuracy			0.97	800000
macro avg	0.85	0.86	0.85	800000
weighted avg	0.97	0.97	0.97	800000

{0: 0.9, 1: 1.9}

	precision	recall	f1-score	support
0	0.98	0.98	0.98	750672
1	0.72	0.72	0.72	49328
...				
accuracy			0.96	800000
macro avg	0.83	0.83	0.83	800000
weighted avg	0.96	0.96	0.96	800000

	precision	recall	f1-score	support
0	1.00	0.94	0.97	750672
1	0.53	0.99	0.69	49328
accuracy			0.94	800000
macro avg	0.76	0.97	0.83	800000
weighted avg	0.97	0.94	0.95	800000

{0: 0.2, 1: 0.8}

	precision	recall	f1-score	support
0	0.98	0.97	0.98	750672
1	0.60	0.77	0.68	49328
accuracy			0.95	800000
macro avg	0.79	0.87	0.83	800000
weighted avg	0.96	0.95	0.96	800000

{0: 0.25, 1: 0.75}

	precision	recall	f1-score	support
0	0.98	0.97	0.98	750672
1	0.64	0.73	0.68	49328
...				
accuracy			0.96	800000
macro avg	0.81	0.85	0.83	800000
weighted avg	0.96	0.96	0.96	800000

Precision and Recall

Increasing the '1' class weight causes the precision for that class to fall sharply. Hence to create more balanced precision recall values, class weights were manually set.

Precision measures that given a '1' prediction, what is the probability that that prediction was correct.

Effect of various thresholds on Logistic Regression (on training set)

For i = 0.05

0.4013511400758679

For i = 0.1

0.5252072768693252

For i = 0.15

0.6002843499161614

For i = 0.2

0.6504478899329049

For i = 0.25

0.6842665075028888

For i = 0.3

0.7083193901453628

For i = 0.35

0.722051671208306

For i = 0.4

0.7289884293317349

For i = 0.45

0.730371782650143

For i = 0.5

0.7247330320361557

For i = 0.55

0.7137793624161075

For i = 0.6

0.6955775010734221

For i = 0.65

0.6708563360068517

For i = 0.7

0.6392026740185912

For i = 0.75

0.5968993248738172

For i = 0.8

0.5406876830223957

For i = 0.85

0.4674079754601227

For i = 0.9

0.3688322500080049

For i = 0.95

0.22381146740962896

For i = 0.99

0.04576351564969965

Effect of various thresholds on Logistic Regression (on validation set)

For i = 0.05
0.39194372357962426
For i = 0.1
0.4992528100838153
For i = 0.15
0.5589438077447367
For i = 0.2
0.597011571733513
For i = 0.25
0.6208764455264759
For i = 0.3
0.6365795724465557
For i = 0.35
0.6451612903225807
For i = 0.4
0.6486348372025112
For i = 0.45
0.6466684567877556
For i = 0.5
0.640734476926794
For i = 0.55
0.6311793487430404
For i = 0.6
0.6162837030150087
For i = 0.65
0.593554996740244
For i = 0.7
0.5670471804419186
For i = 0.75
0.532871972318339
For i = 0.8
0.4880398489244075
For i = 0.85
0.42648604461053363
For i = 0.9
0.34009991778916077

For i = 0.95
0.2117597530171204

For i = 0.99
0.046011073851672774

Using GridSearchCV to find optimal class-weights

[28]:
```
model1 = LogisticRegression(max_iter=1000)
parameters = {'class_weight':[{0:0.1,1:0.2},{0:0.1,1:0.3},{0:0.1,1:0.5},{0:0.1, 1:1},{0:0.1,1:2},{
model=GridSearchCV(model1,parameters,verbose=10,cv=2,scoring='f1')
model.fit(train,y_train)
```

```
Fitting 2 folds for each of 7 candidates, totalling 14 fits
[CV 1/2; 1/7] START class_weight={0: 0.1, 1: 0.2}.............................
[CV 1/2; 1/7] END class_weight={0: 0.1, 1: 0.2};, score=0.570 total time=  52.9s
[CV 2/2; 1/7] START class_weight={0: 0.1, 1: 0.2}.............................
[CV 2/2; 1/7] END class_weight={0: 0.1, 1: 0.2};, score=0.560 total time=  48.6s
[CV 1/2; 2/7] START class_weight={0: 0.1, 1: 0.3}.............................
[CV 1/2; 2/7] END class_weight={0: 0.1, 1: 0.3};, score=0.609 total time=  55.3s
[CV 2/2; 2/7] START class_weight={0: 0.1, 1: 0.3}.............................
[CV 2/2; 2/7] END class_weight={0: 0.1, 1: 0.3};, score=0.600 total time=  58.9s
[CV 1/2; 3/7] START class_weight={0: 0.1, 1: 0.5}.............................
[CV 1/2; 3/7] END class_weight={0: 0.1, 1: 0.5};, score=0.614 total time= 1.2min
[CV 2/2; 3/7] START class_weight={0: 0.1, 1: 0.5}.............................
[CV 2/2; 3/7] END class_weight={0: 0.1, 1: 0.5};, score=0.606 total time= 1.2min
[CV 1/2; 4/7] START class_weight={0: 0.1, 1: 1}...............................
[CV 1/2; 4/7] END ..class_weight={0: 0.1, 1: 1};, score=0.572 total time=  33.9s
[CV 2/2; 4/7] START class_weight={0: 0.1, 1: 1}...............................
[CV 2/2; 4/7] END ..class_weight={0: 0.1, 1: 1};, score=0.569 total time=  46.5s
[CV 1/2; 5/7] START class_weight={0: 0.1, 1: 2}...............................
[CV 1/2; 5/7] END ..class_weight={0: 0.1, 1: 2};, score=0.506 total time=  37.4s
[CV 2/2; 5/7] START class_weight={0: 0.1, 1: 2}...............................
[CV 2/2; 5/7] END ..class_weight={0: 0.1, 1: 2};, score=0.506 total time=  44.4s
[CV 1/2; 6/7] START class_weight={0: 0.1, 1: 3}...............................
[CV 1/2; 6/7] END ..class_weight={0: 0.1, 1: 3};, score=0.466 total time= 1.6min
[CV 2/2; 6/7] START class_weight={0: 0.1, 1: 3}...............................
[CV 2/2; 6/7] END ..class_weight={0: 0.1, 1: 3};, score=0.468 total time= 1.6min
[CV 1/2; 7/7] START class_weight={0: 0.1, 1: 4}...............................
[CV 1/2; 7/7] END ..class_weight={0: 0.1, 1: 4};, score=0.440 total time= 1.7min
[CV 2/2; 7/7] START class_weight={0: 0.1, 1: 4}...............................
[CV 2/2; 7/7] END ..class_weight={0: 0.1, 1: 4};, score=0.443 total time= 1.5min
```

[28]:
```
GridSearchCV(cv=2, estimator=LogisticRegression(max_iter=1000),
             param_grid={'class_weight': [{0: 0.1, 1: 0.2}, {0: 0.1, 1: 0.3},
                                          {0: 0.1, 1: 0.5}, {0: 0.1, 1: 1},
                                          {0: 0.1, 1: 2}, {0: 0.1, 1: 3},
                                          {0: 0.1, 1: 4}]},
             scoring='f1', verbose=10)
```

[29]:
```
model.best_estimator_
```

[29]:
```
LogisticRegression(class_weight={0: 0.1, 1: 0.5}, max_iter=1000)
```

Trie

Alone

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingClassifier
estimators = [
    ('rf', RandomForestClassifier(n_estimators=10, random_st
    ('svc', LinearSVC(random_state=42)),
    ('lr', LogisticRegression(penalty='l2', max_iter=50000,
    solver='lbfgs', class_weight='balanced'))
]
model = StackingClassifier(
    estimators=estimators, final_estimator=LogisticRegressio
)
model.fit(train,y_train)
y_pred = model.predict(train)
print("train f1 score: ",f1_score(y_train,y_pred))
print("test f1 score: ",f1_score(y_test,model.predict(test)))
test_y_pred = model.predict(test_df_matrix)
print(metrics.classification_report(y_train,y_pred) )
```

256m 8.6s

Ensemble

Running the classifier with Logistic Regression, Linear SVM causes the code to run for over 250 minutes [Reset kernel after this]