p3

August 7, 2018

1 DMG2 Assignment : Problem 3

Naive Bayes Classifier, Decision Tree Classifier

```
In [11]: import numpy as np
         import pandas as pd
         import os
         import scipy
         import matplotlib.pyplot as plt
         import seaborn as sns

         from sklearn import tree
         from sklearn.feature_extraction import DictVectorizer
         from sklearn.preprocessing import LabelEncoder
         from sklearn.naive_bayes import MultinomialNB

         sns.set_style('whitegrid')
```

```
In [12]: DATA_DIR = '/home/jishnu/Documents/ISB/Term3/dmg2/assignments/hw_assignment1/dmg2/data
         train = pd.read_csv(os.path.join(DATA_DIR,'train.csv'),usecols=['V{0}'.format(i) for
         test = pd.read_csv(os.path.join(DATA_DIR,'test.csv'),usecols=['V{0}'.format(i) for i

         train.columns
```

```
Out[12]: Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',
                'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',
                'V22', 'V23'],
               dtype='object')
```

```
In [13]: # Vectorizing categorical data
         X_dict = train.iloc[:,1:].T.to_dict().values()
         X_vector = DictVectorizer(sparse=False).fit_transform(X_dict)

         X_test_dict = test.iloc[:,1:].T.to_dict().values()
         X_test_vector = DictVectorizer(sparse=False).fit_transform(X_test_dict)

         # Vectorizing class labels
         le = LabelEncoder()
         Y_train = le.fit_transform(train.iloc[:,0])
         Y_test = le.fit_transform(test.iloc[:,0])
```

1.1 Decision Tree Classifier

```
In [14]: dt_clf = tree.DecisionTreeClassifier(max_depth=10).fit(X_vector,Y_train)
```

```
In [15]: dt_clf.score(X_vector,Y_train)
```

```
Out[15]: 1.0
```

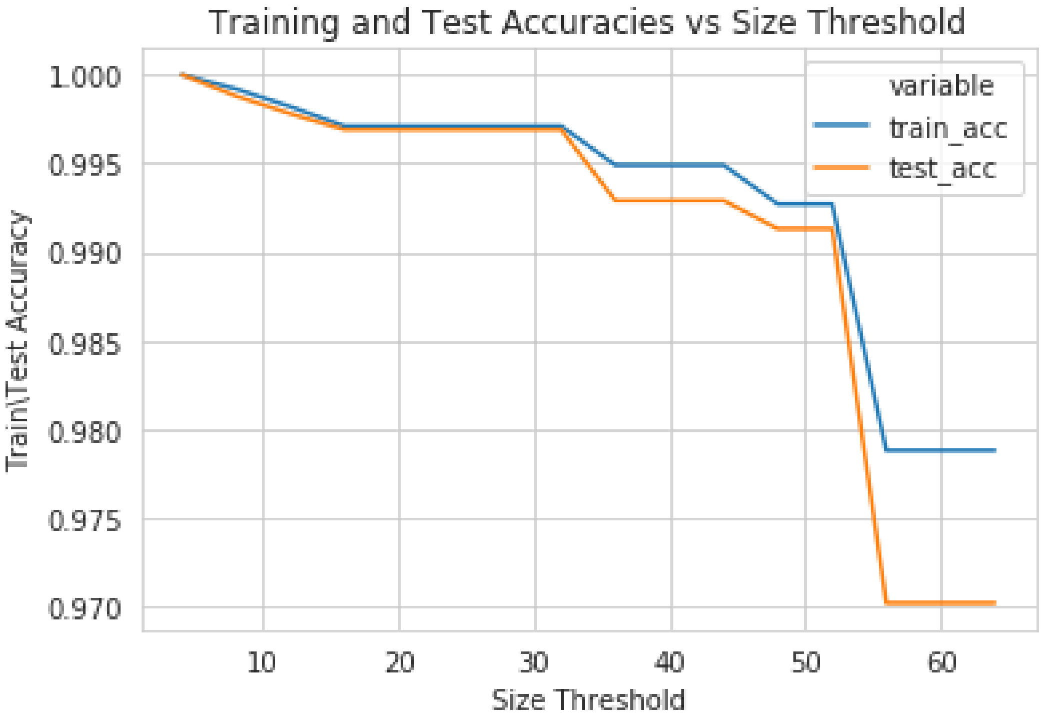
```
In [16]: dt_clf.score(X_test_vector,Y_test)
```

```
Out[16]: 1.0
```

```
In [23]: dt_accuracies = pd.DataFrame(columns=['size_threshold','train_acc','test_acc'])
         for size_threshold in range(4,65,4):
             dt_clf = tree.DecisionTreeClassifier(min_samples_leaf=size_threshold,criterion='en
             train_acc = np.round(dt_clf.score(X_vector,Y_train),4)
             test_acc = np.round(dt_clf.score(X_test_vector,Y_test),4)
             dt_accuracies = dt_accuracies.append({'size_threshold' : size_threshold,'train_acc
         dt_accuracies
```

```
Out[23]:     size_threshold  train_acc  test_acc
         0              4.0     1.0000    1.0000
         1              8.0     0.9992    0.9988
         2             12.0     0.9982    0.9978
         3             16.0     0.9971    0.9969
         4             20.0     0.9971    0.9969
         5             24.0     0.9971    0.9969
         6             28.0     0.9971    0.9969
         7             32.0     0.9971    0.9969
         8             36.0     0.9949    0.9929
         9             40.0     0.9949    0.9929
         10            44.0     0.9949    0.9929
         11            48.0     0.9927    0.9913
         12            52.0     0.9927    0.9913
         13            56.0     0.9788    0.9702
         14            60.0     0.9788    0.9702
         15            64.0     0.9788    0.9702
```

```
In [24]: sns.lineplot(x='size_threshold',y='value',hue='variable',
                      data=dt_accuracies.melt(id_vars=['size_threshold'],value_vars=['train_acc'
                      ci=0)
         plt.xlabel('Size Threshold')
         plt.ylabel('Train\Test Accuracy')
         plt.title('Training and Test Accuracies vs Size Threshold')
         plt.show();
```



The test accuracies start decreasing at around size threshold of 32.

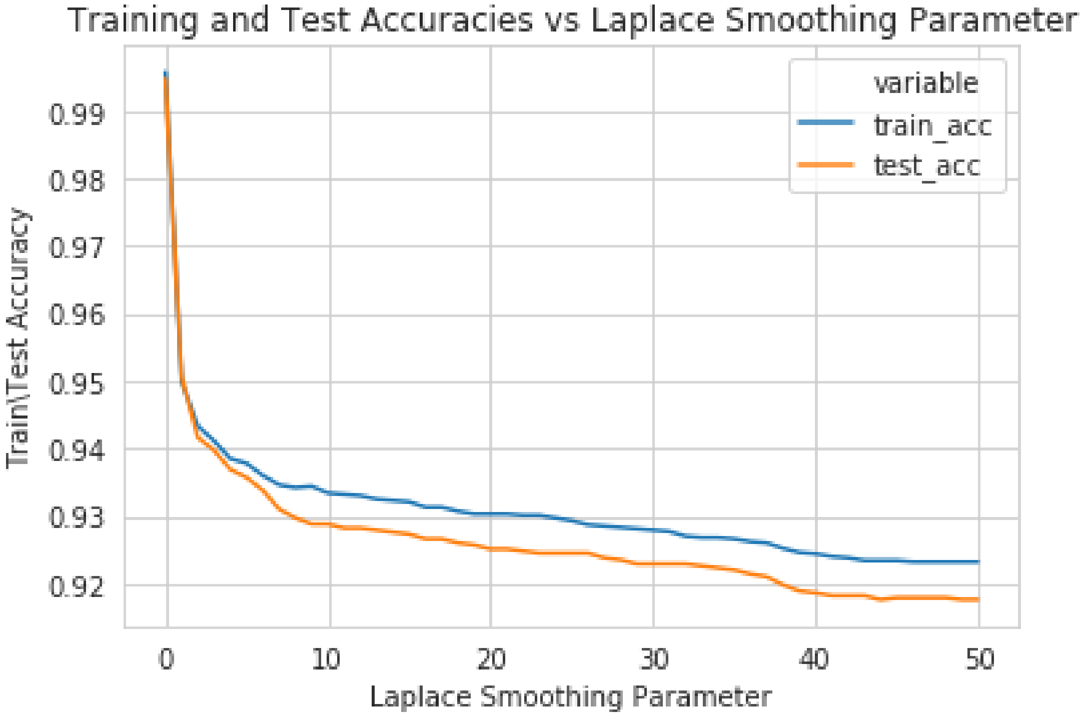
1.2 Naive Bayes Classifier

```
In [19]: nb_accuracies = pd.DataFrame(columns=['lap_sm_param','train_acc','test_acc'])
         for lap_sm_param in range(0,51):
             nb_clf = MultinomialNB(alpha=lap_sm_param).fit(X_vector,Y_train)
             train_acc = np.round(nb_clf.score(X_vector,Y_train),4)
             test_acc = np.round(nb_clf.score(X_test_vector,Y_test),4)
             nb_accuracies = nb_accuracies.append({'lap_sm_param' : lap_sm_param,'train_acc' :
         nb_accuracies.head()
```

```
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/naive_bayes.py:472: UserWarning: al
  'setting alpha = %.1e' % _ALPHA_MIN)
```

```
Out[19]:    lap_sm_param  train_acc  test_acc
         0           0.0     0.9957    0.9947
         1           1.0     0.9499    0.9506
         2           2.0     0.9433    0.9416
         3           3.0     0.9411    0.9397
         4           4.0     0.9385    0.9369
```

```
In [20]: sns.lineplot(x='lap_sm_param',y='value',hue='variable',
                    data=nb_accuracies.melt(id_vars=['lap_sm_param'],value_vars=['train_acc','
                    ci=0)
         plt.xlabel('Laplace Smoothing Parameter')
         plt.ylabel('Train\Test Accuracy')
         plt.title('Training and Test Accuracies vs Laplace Smoothing Parameter')
         plt.show();
```



The best value of test accuracy is achieved when setting smoothing parameter to zero.

The decision tree classifier gives much better accuracies when compared to naive bayes classifier.