LR_Models_VGG19

March 28, 2019

1 Library Import

```
In [1]: import os
        import pandas as pd
        import numpy as np
        import pickle
        import time
        # Machine Learning Algorithms
        from sklearn.model_selection import GridSearchCV
        from sklearn.linear_model import LogisticRegression
        from sklearn.pipeline import make_pipeline
        from sklearn.preprocessing import StandardScaler
        # Metrics
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import precision_score,recall_score,f1_score
        from sklearn.metrics import precision_recall_fscore_support
        from sklearn.model_selection import validation_curve,learning_curve
        from joblib import dump
```

2 Magnification Identification

```
In [2]: train_path="A:\\Projects\\Major Project\\Extracted CNN Features\\VGG19\\train"
        test_path="A:\\Projects\\Major Project\\Extracted CNN Features\\VGG19\\test"
```

```
In [3]: # Training Paths
        X_train=np.load(train_path+"\\data_cnn_VGG19_train.npy")
        Y_train=np.load(train_path+"\\data_mag_VGG19_train.npy")
        # Cancer class
        cancerclass_train=np.load(train_path+"\\data_cancerclass_VGG19_train.npy")
        # Cancer type
        cancertype_train=np.load(train_path+"\\data_cancertype_VGG19_train.npy")
        # Testing Paths
        X_test=np.load(test_path+"\\data_cnn_VGG19_test.npy")
        Y_test=np.load(test_path+"\\data_mag_VGG19_test.npy")
        # Cancer class
        cancerclass_test=np.load(test_path+"\\data_cancerclass_VGG19_test.npy")
        # Cancer type
        cancertype_test=np.load(test_path+"\\data_cancertype_VGG19_test.npy")

In [4]: param_grid={'C':[.001,.01,.1,1,10]}

In [5]: start_time=time.clock()
        gs1=GridSearchCV(LogisticRegression(),param_grid=param_grid,scoring="accuracy",cv=10,n

        start_time = time.clock()
        #Training of Model
        gs1.fit(X_train,Y_train)
        print(time.clock() - start_time, "seconds")

        print(gs1.best_score_)
        print(gs1.best_params_)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  """Entry point for launching an IPython kernel.
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  after removing the cwd from sys.path.
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)
```

```
677.887985028 seconds
0.8343354430379747
{'C': 0.001}
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  import sys
```

```
In [6]: clf=gs1.best_estimator_
        clf.fit(X_train,Y_train)
        print(clf.score(X_test,Y_test))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)
```

```
0.8949367088607595
```

```
In [7]: clf2=LogisticRegression(C=.001)
        clf2.fit(X_train,Y_train)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)
```

```
Out[7]: LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, max_iter=100, multi_class='warn',
                  n_jobs=None, penalty='l2', random_state=None, solver='warn',
                  tol=0.0001, verbose=0, warm_start=False)
```

```
In [8]: print(clf2.score(X_test,Y_test))
```

```
0.8949367088607595
```

```
In [9]: pred=clf2.predict(X_test)
```

```
In [10]: con=confusion_matrix(Y_test,pred)
```

```
In [11]: print(con)
```

```
[[382  15   0   0]
 [ 21 330  41   0]
 [  5  26 336  27]
 [  0   0  31 366]]
```

```
In [12]: precision_score(Y_test, pred, average='micro')
```

```
Out[12]: 0.8949367088607595
```

```
In [13]: recall_score(Y_test, pred, average='micro')
```

```
Out[13]: 0.8949367088607595
```

```
In [14]: f1_score(Y_test, pred, average='micro')
```

```
Out[14]: 0.8949367088607594
```

```
In [15]: precision_recall_fscore_support(Y_test,pred)
```

```
Out[15]: (array([0.93627451, 0.88948787, 0.82352941, 0.93129771]),
          array([0.96221662, 0.84183673, 0.85279188, 0.92191436]),
          array([0.94906832, 0.86500655, 0.83790524, 0.92658228]),
          array([397, 392, 394, 397], dtype=int64))
```

3 CancerClass Identification

```
In [7]: Y_train_40=[]
        X_train_40=[]

        Y_train_100=[]
        X_train_100=[]

        Y_train_200=[]
        X_train_200=[]

        Y_train_400=[]
        X_train_400=[]

        for i in range(0,len(Y_train)):
            if(Y_train[i]==40):
                Y_train_40.append(cancerclass_train[i])
                X_train_40.append(X_train[i])
            if(Y_train[i]==100):
                Y_train_100.append(cancerclass_train[i])
                X_train_100.append(X_train[i])
            if(Y_train[i]==200):
                Y_train_200.append(cancerclass_train[i])
                X_train_200.append(X_train[i])
            if(Y_train[i]==400):
                Y_train_400.append(cancerclass_train[i])
                X_train_400.append(X_train[i])
```

```
In [8]: X_train_40=np.array(X_train_40)
        X_train_100=np.array(X_train_100)
        X_train_200=np.array(X_train_200)
        X_train_400=np.array(X_train_400)
        Y_train_40=np.array(Y_train_40)
        Y_train_100=np.array(Y_train_100)
        Y_train_200=np.array(Y_train_200)
        Y_train_400=np.array(Y_train_400)
        print(Y_train_40.size)
```

```
1596
```

```
In [9]: Y_test_40=[]
        X_test_40=[]

        Y_test_100=[]
        X_test_100=[]

        Y_test_200=[]
        X_test_200=[]
        Y_test_400=[]
        X_test_400=[]

        for i in range(0,len(Y_test)):
            if(Y_test[i]==40):
                Y_test_40.append(cancerclass_test[i])
                X_test_40.append(X_test[i])
            if(Y_test[i]==100):
                Y_test_100.append(cancerclass_test[i])
                X_test_100.append(X_test[i])
            if(Y_test[i]==200):
                Y_test_200.append(cancerclass_test[i])
                X_test_200.append(X_test[i])
            if(Y_test[i]==400):
                Y_test_400.append(cancerclass_test[i])
                X_test_400.append(X_test[i])
```

```
In [10]: X_test_40=np.array(X_test_40)
         X_test_100=np.array(X_test_100)
         X_test_200=np.array(X_test_200)
         X_test_400=np.array(X_test_400)
         Y_test_40=np.array(Y_test_40)
         Y_test_100=np.array(Y_test_100)
         Y_test_200=np.array(Y_test_200)
         Y_test_400=np.array(Y_test_400)
```

4 CancerClass Magnification classification 40

```
In [20]: param_grid={'C':[.001,.01,.1,1,10]}
         gs1=GridSearchCV(LogisticRegression(),param_grid=param_grid,scoring="accuracy",cv=10,

         start_time = time.clock()
         #Training of Model
         gs1.fit(X_train_40,Y_train_40)
         print(time.clock() - start_time, "seconds")

         print(gs1.best_score_)
         print(gs1.best_params_)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  after removing the cwd from sys.path.
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)


20.615935965000062 seconds
0.8427318295739349
```

```
{'C': 0.001}


c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  import sys
```

```
In [21]: clf3=gs1.best_estimator_
         clf3.fit(X_train_40,Y_train_40)
         clf3.score(X_test_40,Y_test_40)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)


Out[21]: 0.7758186397984886
```

```
In [22]: clf=LogisticRegression(C=.01)
         clf.fit(X_train_40,Y_train_40)
         clf.score(X_test_40,Y_test_40)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)


Out[22]: 0.7808564231738035
```

```
In [23]: pred=clf.predict(X_test_40)

In [24]: con=confusion_matrix(Y_test_40,pred)

In [25]: print(con)
```

```
[[138  62]
 [ 25 172]]
```

```
In [26]: precision_score(Y_test_40,pred)

Out[26]: 0.8466257668711656

In [27]: recall_score(Y_test_40,pred)

Out[27]: 0.69

In [28]: f1_score(Y_test_40,pred)

Out[28]: 0.7603305785123967

In [29]: precision_recall_fscore_support(Y_test_40,pred)

Out[29]: (array([0.84662577, 0.73504274]),
          array([0.69      , 0.87309645]),
          array([0.76033058, 0.79814385]),
          array([200, 197], dtype=int64))
```

5 CancerClass Magnification classification 100

```
In [30]: gs2=GridSearchCV(LogisticRegression(),param_grid=param_grid,scoring="accuracy",cv=10,r

         start_time = time.clock()
         #Training of Model
         gs2.fit(X_train_100,Y_train_100)
         print(time.clock() - start_time, "seconds")

         print(gs2.best_score_)
         print(gs2.best_params_)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
```

```
24.790742195999997 seconds
0.8446947243627742
{'C': 0.01}
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
```

```
In [31]: c=LogisticRegression(C=.01)
         c.fit(X_train_100,Y_train_100)
         c.score(X_test_100,Y_test_100)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
```

```
Out[31]: 0.8239795918367347
```

6 CancerClass Magnification classification 200

```
In [32]: gs3=GridSearchCV(LogisticRegression(),param_grid=param_grid,scoring="accuracy",cv=10,r

         start_time = time.clock()
         #Training of Model
         gs3.fit(X_train_200,Y_train_200)
         print(time.clock() - start_time, "seconds")

         print(gs3.best_score_)
         print(gs3.best_params_)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
```

```
20.969849230000023 seconds
0.8744588744588745
{'C': 0.001}
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
```

```
In [33]: c=LogisticRegression(C=.001)
         c.fit(X_train_200,Y_train_200)
         c.score(X_test_200,Y_test_200)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
```

```
Out[33]: 0.8223350253807107
```

7 CancerClass Magnification classification 400

```
In [34]: gs4=GridSearchCV(LogisticRegression(),param_grid=param_grid,scoring="accuracy",cv=10,r

         start_time = time.clock()
         #Training of Model
         gs4.fit(X_train_400,Y_train_400)
         print(time.clock() - start_time, "seconds")

         print(gs4.best_score_)
         print(gs4.best_params_)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
```

```
18.39875611800005 seconds
0.8704225352112676
{'C': 0.001}
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launch
```

```
In [35]: c=LogisticRegression(C=.001)
         c.fit(X_train_400,Y_train_400)
         c.score(X_test_400,Y_test_400)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_m
  FutureWarning)
```

```
Out[35]: 0.7783375314861462
```

7.1 Benign Sub-Classification Using Cancer Classification

```
In [24]: Y_train_1=[]
         X_train_1=[]

         for i in range(0,len(Y_train)):
             if(cancerclass_train[i]==1):
                 Y_train_1.append(cancertype_train[i])
                 X_train_1.append(X_train[i])

         X_train_1=np.array(X_train_1)
         Y_train_1=np.array(Y_train_1)
         print(Y_train_1.size)

         Y_test_1=[]
         X_test_1=[]

         for i in range(0,len(Y_test)):
             if(cancerclass_test[i]==1):
                 Y_test_1.append(cancertype_test[i])
                 X_test_1.append(X_test[i])

         X_test_1=np.array(X_test_1)
         Y_test_1=np.array(Y_test_1)
```

```
1683
```

```
In [39]: classes=[11,12,13,14]
```

```
In [ ]:
```

```
In [40]: from sklearn.utils.class_weight import compute_class_weight
```

```
In [41]: class_weight=compute_class_weight(class_weight='balanced', classes=classes,y=Y_train_
In [42]: print(class_weight)

[1.66964286 0.51752768 1.67629482 1.14645777]


In [43]: print(np.unique(Y_train_1))

[11 12 13 14]


In [44]: print(len(X_train_1))

1683


In [45]: print(len(Y_test_1))

792


In [46]: d = dict(enumerate(class_weight, 1))

In [47]: print(d)

{1: 1.6696428571428572, 2: 0.5175276752767528, 3: 1.6762948207171315, 4: 1.146457765667575}


In [48]: d1={1:11,2:12,3:13,4:14}

In [49]: d=dict((d1[key], value) for (key, value) in d.items())

In [50]: d

Out[50]: {11: 1.6696428571428572,
          12: 0.5175276752767528,
          13: 1.6762948207171315,
          14: 1.146457765667575}

In [51]: gs3=GridSearchCV(LogisticRegression(class_weight=d),param_grid=param_grid,scoring="ac

         start_time = time.clock()
         #Training of Model
         gs3.fit(X_train_1,Y_train_1)
         print(time.clock() - start_time, "seconds")

         print(gs3.best_score_)
         print(gs3.best_params_)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\model_sel
  DeprecationWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)
```

```
84.736160165 seconds
0.679144385026738
{'C': 0.01}
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
```

```
In [52]: clf4=gs3.best_estimator_
         clf4.fit(X_train_1,Y_train_1)
         print(clf4.score(X_test_1,Y_test_1))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)
```

```
0.44823232323232326
```

```
In [53]: pred=clf4.predict(X_test_1)
```

```
In [54]: precision_recall_fscore_support(Y_test_1,pred)
```

```
Out[54]: (array([0.68867925, 0.34899329, 0.4047619 , 0.59354839]),
          array([0.38219895, 0.78      , 0.16915423, 0.46      ]),
          array([0.49158249, 0.48222566, 0.23859649, 0.51830986]),
          array([191, 200, 201, 200], dtype=int64))
```

```
In [55]: confusion_matrix(Y_test_1,pred)
```

```
Out[55]: array([[ 73,  64,  16,  38],
                [ 11, 156,  16,  17],
                [  9, 150,  34,   8],
                [ 13,  77,  18,  92]], dtype=int64)
```

7.2 Malignant Sub-Classification Using Cancer Classification

```
In [28]: Y_train_2=[]
         X_train_2=[]

         for i in range(0,len(Y_train)):
             if(cancerclass_train[i]==2):
                 Y_train_2.append(cancertype_train[i])
                 X_train_2.append(X_train[i])

         X_train_2=np.array(X_train_2)
         Y_train_2=np.array(Y_train_2)
         print(Y_train_2.size)

         Y_test_2=[]
         X_test_2=[]

         for i in range(0,len(Y_test)):
             if(cancerclass_test[i]==2):
                 Y_test_2.append(cancertype_test[i])
                 X_test_2.append(X_test[i])

         X_test_2=np.array(X_test_2)
         Y_test_2=np.array(Y_test_2)
```

```
4637
```

```
In [69]: classes=[21,22,23,24]
```

```
In [70]: from sklearn.utils.class_weight import compute_class_weight
```

```
In [71]: class_weight=compute_class_weight(class_weight='balanced', classes=classes,y=Y_train_
```

```
In [72]: print(class_weight)
```

```
[0.35669231 2.72764706 1.96150592 3.12466307]
```

```
In [73]: print(np.unique(Y_train_2))
```

```
[21 22 23 24]
```

```
In [74]: print(len(X_train_2))
```

```
4637
```

```
In [75]: print(len(Y_test_2))
```

```
788
```

```
In [76]: d = dict(enumerate(class_weight, 1))
```

```
In [77]: print(d)
```

```
{1: 0.3566923076923077, 2: 2.7276470588235293, 3: 1.9615059221658206, 4: 3.1246630727762805}
```

```
In [78]: d1={1:21,2:22,3:23,4:24}
```

```
In [79]: d=dict((d1[key], value) for (key, value) in d.items())
```

```
In [80]: d
```

```
Out[80]: {21: 0.3566923076923077,
          22: 2.7276470588235293,
          23: 1.9615059221658206,
          24: 3.1246630727762805}
```

```
In [81]: gs3=GridSearchCV(LogisticRegression(class_weight=d),param_grid=param_grid,scoring="acc

         start_time = time.clock()
         #Training of Model
         gs3.fit(X_train_2,Y_train_2)
         print(time.clock() - start_time, "seconds")

         print(gs3.best_score_)
         print(gs3.best_params_)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)
```

```
459.60440757799995 seconds
0.6957084321759759
{'C': 0.01}
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
```

```
In [82]: clf4=gs3.best_estimator_
         clf4.fit(X_train_2,Y_train_2)
         print(clf4.score(X_test_2,Y_test_2))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)


0.3730964467005076
```

```
In [83]: pred=clf4.predict(X_test_2)

In [84]: precision_recall_fscore_support(Y_test_2,pred)

Out[84]: (array([0.33865248, 0.54411765, 0.43103448, 0.4       ]),
          array([0.955     , 0.185     , 0.25      , 0.08510638]),
          array([0.5       , 0.2761194 , 0.3164557 , 0.14035088]),
          array([200, 200, 200, 188], dtype=int64))

In [85]: confusion_matrix(Y_test_2,pred)

Out[85]: array([[191,   6,   1,   2],
                [147,  37,  15,   1],
                [116,  13,  50,  21],
                [110,  12,  50,  16]], dtype=int64)
```

8 Dumping Models

```
In [14]: clf=LogisticRegression(C=.001)
         clf.fit(X_train,Y_train)
         clf.score(X_test,Y_test)

c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)


Out[14]: 0.8949367088607595

In [15]: dump(clf,'models/LR/LR_Models_VGG19_Magnification.joblib')

Out[15]: ['models/LR/LR_Models_VGG19_Magnification.joblib']

In [ ]:

In [16]: clf=LogisticRegression(C=.001)
         clf.fit(X_train_40,Y_train_40)
         clf.score(X_test_40,Y_test_40)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_m
  FutureWarning)
```

```
Out[16]: 0.7758186397984886
```

```
In [17]: dump(clf,'models/LR/LR_Models_VGG19_Magnification_40.joblib')
```

```
Out[17]: ['models/LR/LR_Models_VGG19_Magnification_40.joblib']
```

```
In [ ]:
```

```
In [18]: clf=LogisticRegression(C=.01)
         clf.fit(X_train_100,Y_train_100)
         clf.score(X_test_100,Y_test_100)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_m
  FutureWarning)
```

```
Out[18]: 0.8239795918367347
```

```
In [19]: dump(clf,'models/LR/LR_Models_VGG19_Magnification_100.joblib')
```

```
Out[19]: ['models/LR/LR_Models_VGG19_Magnification_100.joblib']
```

```
In [ ]:
```

```
In [ ]:
```

```
In [32]: clf=LogisticRegression(C=.001)
         clf.fit(X_train_200,Y_train_200)
         clf.score(X_test_200,Y_test_200)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_m
  FutureWarning)
```

```
Out[32]: 0.8223350253807107
```

```
In [33]: dump(clf,'models/LR/LR_Models_VGG19_Magnification_200.joblib')
```

```
Out[33]: ['models/LR/LR_Models_VGG19_Magnification_200.joblib']
```

```
In [22]: clf=LogisticRegression(C=.001)
         clf.fit(X_train_400,Y_train_400)
         clf.score(X_test_400,Y_test_400)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_m
  FutureWarning)
```

```
Out[22]: 0.7783375314861462

In [23]: dump(clf,'models/LR/LR_Models_VGG19_Magnification_400.joblib')

Out[23]: ['models/LR/LR_Models_VGG19_Magnification_400.joblib']

In [ ]:

In [25]: clf=LogisticRegression(C=.01)
         clf.fit(X_train_1,Y_train_1)
         clf.score(X_test_1,Y_test_1)

c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)


Out[25]: 0.4356060606060606

In [26]: dump(clf,'models/LR/LR_Models_VGG19_CancerType_Benign.joblib')

Out[26]: ['models/LR/LR_Models_VGG19_CancerType_Benign.joblib']

In [ ]:

In [29]: clf=LogisticRegression(C=.01)
         clf.fit(X_train_2,Y_train_2)
         clf.score(X_test_2,Y_test_2)

c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  FutureWarning)
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_mo
  "this warning.", FutureWarning)


Out[29]: 0.36421319796954316

In [31]: dump(clf,'models/LR/LR_Models_VGG19_CancerType_Malignant.joblib')

Out[31]: ['models/LR/LR_Models_VGG19_CancerType_Malignant.joblib']

In [ ]:
```