Data Science Bowl 2017

Professor:

Dr. Lawrence V. Fulton

Team Members:
Ratnam Dubey

KAGGLE USERNAME: -
RATNAM

TABLE OF CONTENTS

Preface

DATA SCIENCE BOWL 2017 .... **1**

1. PROBLEM DISCUSSION **3**

2. LITERATURE **3**

3. DATA MINING / CLEANING **4**

4. DATA VISUALIZATION **5**

5. PREDICTIVE TECHNIQUES **7**

5.1 LOGISTIC REGRESSION IN PYTHON 7
5.2 RANDOM FOREST IN PYTHON 7
5.3 XGB BOOSTING REPRESSOR IN PYTHON 8

6 FORMULATION / LIBRARIES **8**

6.1 NUMPY 8
6.2 PANDAS 8
6.3 STANDARDSCALER 8
6.4 PRE-PROCESSING 8

7 MODEL PERFORMANCE **9**

8 LIMITATIONS **10**

9 LEARNING **10**

10 REFERENCES **11**

1. Problem Discussion

In the United States, lung cancer strikes 225,000 people every year, and accounts for $12 billion in health care costs. Early detection is critical to give patients the best chance at recovery and survival. One year ago, the office of the U.S. Vice President spearheaded a bold new initiative, the Cancer Moonshot, to make a decade's worth of progress in cancer prevention, diagnosis, and treatment in just 5 years.

In 2017, the Data Science Bowl will be a critical milestone in support of the Cancer Moonshot by convening the data science and medical communities to develop lung cancer detection algorithms. Using a data set of thousands of high-resolution lung scans provided by the National Cancer Institute, participants will develop algorithms that accurately determine when lesions in the lungs are cancerous. This will dramatically reduce the false positive rate that plagues the current detection technology, get patients earlier access to life-saving interventions, and give radiologists more time to spend with their patients.

2. Literature

Understood the medical terminologies from the various discussions from Kaggle and several other websites also understood various Classification models and Bayesian Recognition Procedure (BPR) from the different sources. Some of the sources are rich in parameter tuning also helped to predict the Classification Correctly.

https://www.kaggle.com/c/data-science-bowl-2017/discussion/30686
https://kaggle2.blob.core.windows.net/forum-message-attachments/171018/6199/AnalysMRI.pdf
http://deeplearning.net/tutorial/logreg.html
http://xgboost.readthedocs.io/en/latest/R-package/xgboostPresentation.html
http://wgrass.media.osaka-cu.ac.jp/gisideas10/viewpaper.php?id=342

Various techniques were used in the Cancer detection and Classification all of the techniques which I have discovered in the Competition are provided above. Most of the techniques were based on the image learning but it will surely take more time to complete so as to enhance the process and get the better result I have converted the images into the Flat csv file and then train the model based on the inputs which I get. I Extracted the features from the R Code which take some time to get complete whereas I have trained the model using python code.

3. Data Mining / Cleaning

3.1 Data Understanding

The datasets we used in our project came from an on-going Kaggle(*source:* https://www.kaggle.com/c/data-science-bowl-2017*)* competition. Data has been provided in below files: -

1) DCM files for the Images of the Patients
2) Sample Submission Files
3) Sample Images Data
4) Labels file for the Patient Cancer Information (Training Data)

3.2 Data Preparation and Feature identification

We need to predict the Cancer for the given patient. Information need to be Extracted from the DCM files. With the help of R Script, we have extracted the features from the all the training and test data. Data extracted from the dcm files have columns like

1) PatientId: - Unique identification for the patient.
2) ImageId: - Unique identification of the images
3) SliceID: - Slice of the heart Information
4) V1-V1024: - Pixel information for the Patient

Variables are only in string and integers where PatientId and ImageId are objects whereas SliceID and V1-V1024 are numerical values.

Other Files like Labels have the information about Cancer or Not identified with Patient ID. It is used for the Training Data

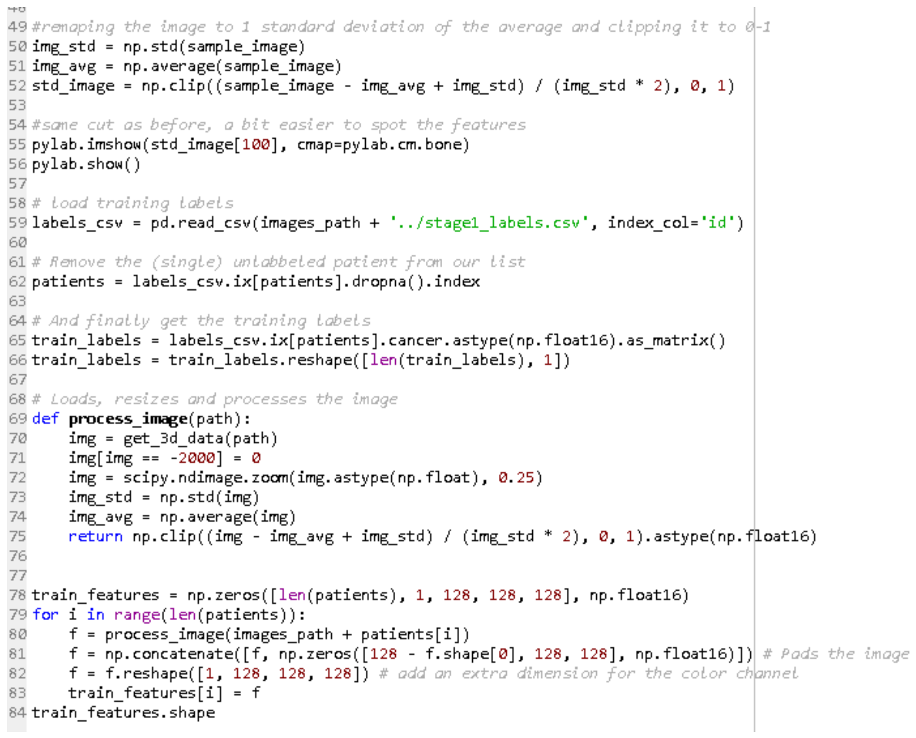
3.3 Missing Values

Considering the datasets this shows that the dataset is complete and there is no need of doing to clean it from empty entries'. there is no Null value in the training data, however in the test data slice ID's are missing which been replaced with the patient SliceID mean data.

4. Data Visualization

I used both R and Python to generate the models. But for the Data visualization I have used python for the heart Images.

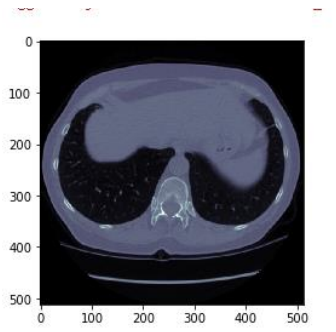
Data visualization code for the Image generation

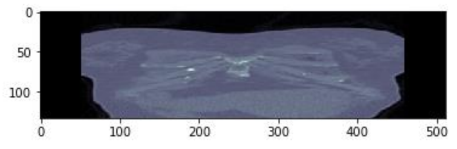


```python
#remaping the image to 1 standard deviation of the average and clipping it to 0-1
img_std = np.std(sample_image)
img_avg = np.average(sample_image)
std_image = np.clip((sample_image - img_avg + img_std) / (img_std * 2), 0, 1)

#same cut as before, a bit easier to spot the features
pylab.imshow(std_image[100], cmap=pylab.cm.bone)
pylab.show()

# load training labels
labels_csv = pd.read_csv(images_path + '../stage1_labels.csv', index_col='id')

# Remove the (single) unlabbeled patient from our list
patients = labels_csv.ix[patients].dropna().index

# And finally get the training labels
train_labels = labels_csv.ix[patients].cancer.astype(np.float16).as_matrix()
train_labels = train_labels.reshape([len(train_labels), 1])

# Loads, resizes and processes the image
def process_image(path):
    img = get_3d_data(path)
    img[img == -2000] = 0
    img = scipy.ndimage.zoom(img.astype(np.float), 0.25)
    img_std = np.std(img)
    img_avg = np.average(img)
    return np.clip((img - img_avg + img_std) / (img_std * 2), 0, 1).astype(np.float16)


train_features = np.zeros([len(patients), 1, 128, 128, 128], np.float16)
for i in range(len(patients)):
    f = process_image(images_path + patients[i])
    f = np.concatenate([f, np.zeros([128 - f.shape[0], 128, 128], np.float16)]) # Pads the image
    f = f.reshape([1, 128, 128, 128]) # add an extra dimension for the color channel
    train_features[i] = f
train_features.shape
```

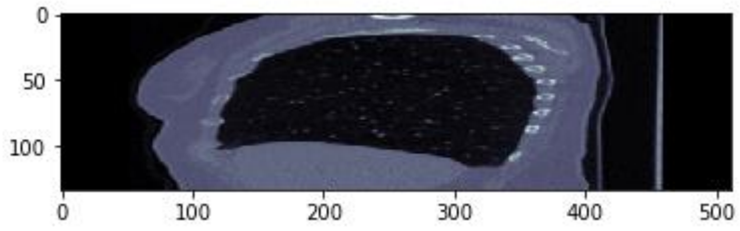
Some of the Generated images for the Data Visualization

- Z and Y Axis

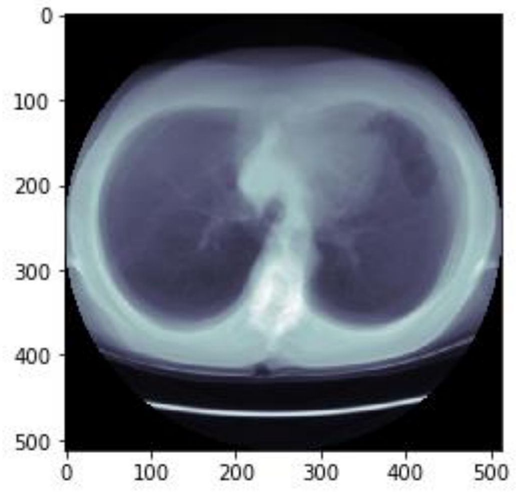


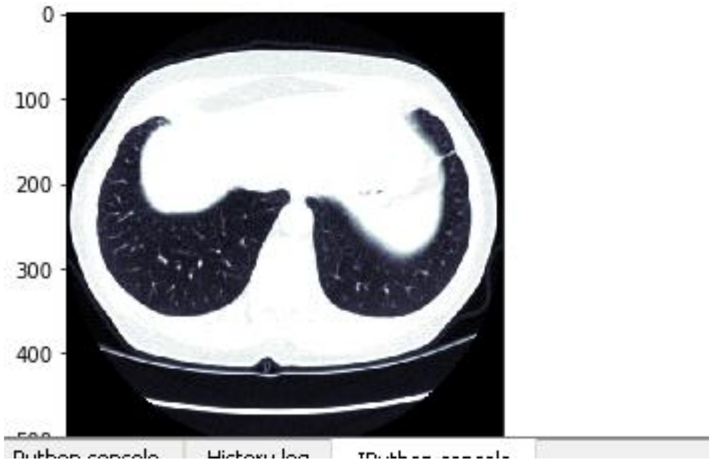


- X axis



- Slicing





5. Predictive Techniques

I used both R and Python to generate of data and modelling the models. However, R seems to be an easy choice where we could do the analysis and Image Extraction in a quick time. To train the data I used python for the training the dataset. I have used scikit learn, Numpy, Pandas package as the part data manipulation, file generation and modelling techniques.

Apart from XGB algorithms using the python package, I also tried Logistic Regression algorithms and Random Forest algorithms in python. However, the best accuracy we got is with XGB Boost having tuned parameters and well-structured data. Board score of 0.57 which was better than the other three models.
I got 11.57 with random forest without tuned parameters, and score of 2.59 with Logistic regression.

These output files are produced using 3 different models: -

- ✓ Logistic Regression in Python
- ✓ Random Forest algorithm in Python
- ✓ XGB algorithm in python

5.1 Logistic Regression in Python

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross- entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag' and 'newton-cg' solvers.) This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag' and 'lbfgs' solvers. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied). The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty.

5.2 Random Forest in Python

Random Forest (RF) is a powerful classification tool. When given a set of data, RF generates a forest of classification trees, rather than a single classification tree. Each of these trees generates a classification for a given set of attributes. A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

5.3 XGB Boosting Repressor in Python

XGB Boosted Regression or shorter XGB Boosting is a flexible non-parametric statistical learning technique for classification and regression. XGB boosting Regression is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

6 Formulation / Libraries

We have used several libraries to analyze text as well as image Information. We have.

6.1 Numpy

Numpy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

6.2 Pandas

pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. pandas are free software released under the three-clause BSD license.

6.3 StandardScaler

Standardize features by removing the mean and scaling to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method.

6.4 Pre-Processing

Standardization of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.

7 Model Performance

We have Enhanced the model performance from *(Kaggle competition score*) 11.56485 to 0.5739 by adding several features and by tuning the parameters.

7.1 XGB Boosting Repressor in Python

XGB Boosting for regression builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative XGB of the given loss function. As the part of the parameter tuning used different parameters, in tuning parameter we have used estimators as 700, learning rate as 0.05, maximum depth as 10 and loss as ls and fitted the model based on the same parameters. generated the predicted values using the same parameter. Using ` XGB Boosting Repressor in Python` best accuracy I got is with leadership Board score of 0.57

7.2 Scikit Random Forest in Python

We could further experiment with deeper trees or a higher percentage of columns used (mtries). The general guidance is to lower the number to increase generalization (avoid overfitting), increase to better fit the distribution. Though usually unnecessary, if a problem has a very important categorical predictor, this can improve performance. In a production setting where fine-grain accuracy is beneficial, it is common to set the learn rate to a very small number, such as 0.01 or less, and add trees to match. Use of early stopping is very powerful to allow the setting of a low learning rate and then building as many trees as needed until the desired convergence is met. Using `random Forest algorithms` best accuracy I got is with leadership Board score of 11.57.

7.3 Logistic Regression in Python

Logistic regression has taken the less time as compare to the other two models. The general idea is to lower the number to increase generalization (avoid overfitting), increase to better fit the distribution. Though usually unnecessary, if a problem has a very important categorical predictor, this can improve performance. Used "lbfgs" as the kernel and tuned the parameter parameters also taken the C as 1000 and tol as 0.05 as the part of performance tuning. Board score from logistic model is 2. 59.

8 Limitations

8.1 Process Time

Each model took a lot of time to process the execution which has certainly became a drawback because of which we couldn’t make more hit and trails to the exiting model’s.

8.2 Complexity

Most of the data was categorical which limits the usage of the models however we have changed into numeric values but that didn’t show much improvement in the performance and model accuracy.

8.3 Kaggle Limitations

In Data Bowl Cancer Prediction, we have limited upload of the submission file as due to which much more experiments are limited.

9 Learning

We have got the more exposure to various algorithms and classifiers, we have learned to tune parameters. Now the GBM is close to the initial random forest. However, we used a default random forest. Random forest's primary strength is how well it runs with standard parameters. And while there are only a few parameters to tune, we can experiment with those to see if it will make a difference. The main parameters to tune are the tree depth.

10 References

- ✓ XGB Boosting Machine- https://www.rdocumentation.org/packages/h2o/versions/3.10.0.8/topics/h2o.gbm
- ✓ Build A Big Data Random Forest Model- https://www.rdocumentation.org/packages/h2o/versions/3.10.0.8/topics/h2o.randomForest
- ✓ https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/
- ✓ https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/
- ✓ https://www.datarobot.com/blog/XGB-boosted-regression-trees/
- ✓ http://scikit-learn.org/stable/auto_examples/linear_model/plot_logistic_l1_l2_sparsity.html
- ✓ http://scikit-learn.org/stable/modules/preprocessing.html