BIG DATA AND MACHINE LEARNING

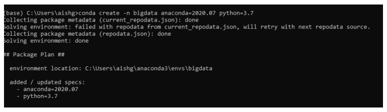
ASSIGNMENT-3

Aishwarya Gouru *dept. of computer Science)*

***Abstract*—This paper discusses about the big data concepts using machine learning techniques. The goal of the paper is to determine the accuracy of the machine. To learn about big data and machine learning, we used digital images as raw data and transformed them into feature vectors. Four feature spaces are used for training and testing the system. The accuracy is obtained by performing elastic net regression, random forest algorithms.**

I. Introduction

Big data and Machine learning is widely used in today's world. If we want to train the system with a raw image, then we first load the image to the system. https://docs.anaconda.com/anaconda/ is the website that is used to install the anaconda. After installing Anaconda, necessary libraries like openCV are installed. Python is the programming language that is used for the entire execution. The issues and challenges of integrating new networking technology and machine learning algorithms to solve the Big Data classification issue for the classification of data are discussed in this article.



```
(base) C:\Users\aishg>conda create -n bigdata anaconda=2020.07 python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\aishg\anaconda3\envs\bigdata

  added / updated specs:
    - anaconda=2020.07
    - python=3.7
```

Fig. 1: Anaconda Prompt

II. Requirement Acquisition

A specific dataset has been used for the entire assignment. The dataset has been downloaded from an online website [1]. This dataset has many number of different fruits and vegetables images as dataset. Of all the images of fruits and vegetables, we are using three different fruit images for the better understanding of this course. The three different images selected are cantaloupe, mango and raspberry. The selected images are attached to the assignment folder. All the images selected must not have a background! Background will lead to negative accuracy. Once the image is loaded to the system, we start to process the data. In Big data, each and every pixel of the image is considered as a data point. Similarly, the image that we loaded to the system, the machine considers each and every pixel of the image as a data point. The original dimensions of an image are (340, 530, 3). The '3' in the dimensions indicate that the image is having a three colour channel. This image is now converted to a gray scale by removing all the color channels. after the image is converted to gray scale image, the dimensions are changed to (340, 530). The gray scale image eliminates the color channels in the image. Once the image has no color channels, we proceed further by resizing the image.

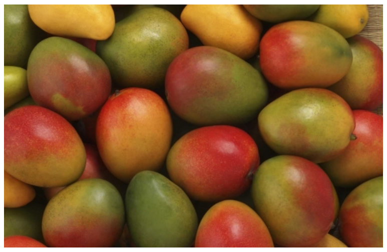


Fig. 2: Raw image of Mango

Code:

```
mangoC = cv2.imread("C:/images/57.jpg")
```

This command reads the image from the directory.

Code:

```
plt.imshow(mangoC[:,:,0], cmap=
                plt.get_cmap('Reds'))
plt.imshow(mangoC[:,:,1], cmap=
                plt.get_cmap('Greens'))
plt.imshow(mangoC[:,:,2], cmap=
                plt.get_cmap('Blues'))
```

This command displays the color channels of the image.

III. Image Resizing

This gray scale image is resized to match the aspect ratio of the original image. The image below shows the transition of raw image to gray image and then to image resizing. The gray-scale images that are generated in the above step are resized such that their output dimensions are divisible by 8 without changing their original aspect ratios. Resizing is done by maintaining the height of all the images to 256 pixels and the width selected maintains the aspect ratio that is also divisible by 8. It goes like, from (340, 530), the image is resized to match the aspect ratio as (128, 256).

Code:

```
mango = cv2.resize(mangoG, dsize=(256, 128),
interpolation=cv2.INTER\_CUBIC)
```

This command resizes the image to specific aspects.

Code:

```
mango1 = cv2.normalize(mango.astype('float'),
None, 0.0, 1.0, cv2.NORM\_MINMAX)*255
```

This tunes the image by normalizing the image.

Fig. 3: Resized gray scale image

IV. Non overlapping sliding window

Each image is divided into blocks of 8 x 8 pixels. This is done to transform them to vectors of size 64. Since we are generating block-feature vectors in a supervised learning, we assign labels to the feature spaces. The labels assigned for Mango, Cantaloupe and Raspberry are 0,1 and 2 respectively. For each image, feature spaces are generated. Labels are added to the feature spaces in the column 65 for each image respectively.

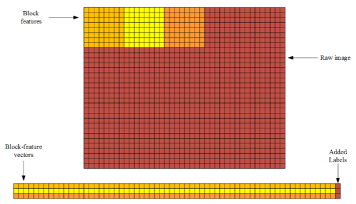


Fig. 4: Example of non overlapping window

Code:

```
mm = round(((heightm-8)*(widthm-8))/64)
flatm = np.zeros((mm, 65), np.uint8)
k = 0
for i in range(0,heightm-8,8):
    for j in range(0,widthm-8,8):
        crop_tmp1 = mango1[i:i+8,j:j+8]
        flatm[k,0:64] = crop_tmp1.flatten()
        k = k + 1
fspaceM = pd.DataFrame(flatm)
```

V. Overlapping sliding window

Each image is divided into sliding blocks of 8 x 8 pixels. This is done to transform them to vectors of size 64. Since we are generating sliding block-feature vectors in a supervised learning, we assign labels to the feature spaces similar to the above step. The labels assigned for Mango, Cantaloupe and Raspberry are 0,1 and 2 respectively. For each image, feature spaces are generated. These feature spaces contain so many features as the window is overlapped. The dataset generated is very huge depending upon the window size and the size of overlapping pixels.

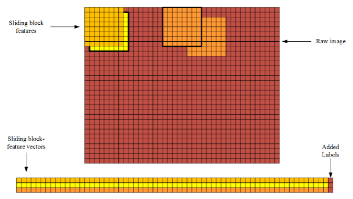


Fig. 5: Example of overlapping window

VI. Statistical information

- Normalization is performed so that the resized images are smoothed. This helps in making the image smooth instead of making the image look pixelated and sharp.
- The high dimensionality of the images is removed by removing the color channels. The gray channel image is means there is no high dimensionality.
- After generating the feature spaces by reducing the dimensionality of the data, the feature spaces that are generated are neither imbalanced nor inaccurate nor incomplete. This is answered by generating the histograms and scatter plots and carefully examining the data. The scatter plot gives the information of how the data is distributes in the feature space.

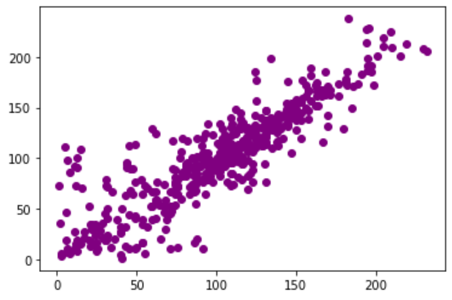


Fig. 6: Scatter plot of mango image

- The scatter plot shows that the data is evenly distributed. There is no inaccurate data as the data is not scattered in the plane. The 3-D distribution of merged dataset shows that there is no imbalanced data. That is, if we have more observations in one class than in the other class, then we can say the data is imbalanced[2].

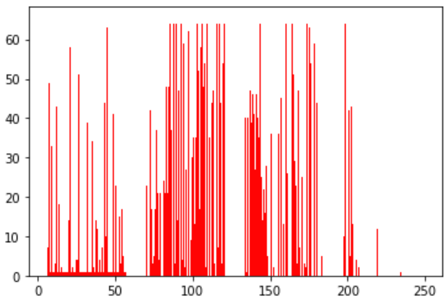


Fig. 7: Histogram of mango dataset

- The histograms give the information of the data where the feature vectors are highly densed in the feature spaces. In the above histogram of mango image, the data is continuous data as there are no outliers.
- The generated data is Big data and not trivial data. It is big data because the noisy data is removed and the data is simplified yet complicated for a human to understand the feature space.
- standardizing of the data is not performed in this paper.
- Having said this, the data does not have scalability issue as there is no missing data present in the feature spaces. In order to eliminate the scalability issue, we have chosen the images with the background.
- For the statistical information, we have generated all the information using the python code attached to this folder. The statistical information is determined for the non overlapping sliding window feature spaces as discussed below.
- By giving the command data.desc(), the following information is derived: The count, mean, standard deviation, minimum value, and maximum value of each observation.

VII. Merging of Feature Spaces

Feature spaces generated are used for training the system. By following the above steps, we will achieve 6 feature spaces. Merging the feature spaces in mango.csv and cant.csv we create a feature space for these images together. Each feature and label columns are aligned vertically to generate the correct feature space for these image classes. These feature space file is saved as nonSlideMC.csv

These features vectors are permuted. That means, they are randomly arranged and merged. This helps the machine to understand the dataset thoroughly.

Similarly the feature vectors of mango.csv, cant.csv, and rasp.csv are merged to create a feature space for these images. Each feature and label columns are aligned vertically to generate the correct feature space for these three classes. These feature space file is named as nonSlideMCR.csv. This dataset is also randomized.

Code:

```
mgedMCR = pd.concat(framesMCR)

indx = np.arange(len(mgedMCR))
rndmgedMCR = np.random.permutation(indx)
rndmgedMCR =mgedMCR.sample(frac=1).
reset_index(drop=True)
```

VIII. Displaying of Scatter plots

Two features are selected from the obtained merged feature space and the two-dimensional feature space is plotted with labeling the observations (vectors) of the fruits that are selected by using the spreadsheets that are generated. This also gives the information of the how the data is scattered in the plane and also explains if the data is inconsistent or imbalanced. The 2- Dimensional and 3-Dimensional plots are plotted.

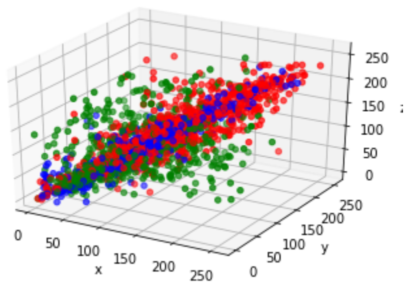


Fig. 8: Scatter plots of multi class observation

IX. Significance of the block size

For a sliding window, irrespective of overlapping or non overlapping, the size of the window matters. The window must be carefully chosen as there can be problems associated with it. If the window size id not correctly chosen, then it may incur out-of-handling error. That means, suppose, if the window is of fixed size of 8*8 pixels, then the sliding window works well if the dimensions are exactly divided by 8 otherwise, the window may come out of the data which leads to out-of-handling. Hence, for a non overlapping sliding window technique, we must be careful in choosing the window size. For an overlapping sliding window, the window size plays a huge role. If we chose the overlapping pixel size as 1 for an 8*8 sliding window, then that causes in developing a very huge feature space. Hence, carefully determine the window size for both overlapping and non overlapping sliding window.

X. Training and testing datasets

By following the above steps, we generated four distinct categories of datasets: non-overlapping feature vectors for two-class classification, overlapping feature vectors for two-class classification, non-overlapping feature vectors for multi-class classification, and overlapping feature vectors for multi-class classification. These 4 datasets are used further in this paper for performing several operations. We develop the training and testing datasets for training and testing the machine. This is achieved by randomizing the data and diving the data in 80:20 ration. 80% of the dataset is used for training the system while the rest of 20% of the dataset is used for testing the system. By performing this step, we will achieve a total datasets, 4 of training datasets and 4 testing datasets.

After dividing the dataset to training and testing dataset, we check if there is even distribution of data by plotting the histogram.

The histogram gives the information on how the data is distributed in the dataset. The shape of the histograms determines that the data is evenly distributed. There are no outliers also. This data can be used to successfully train the system. We may also determine the mean and variance of the data. This histogram shows the even distribution of two features as the histograms are overlapped.

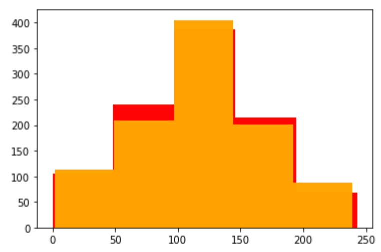


Fig. 9: Histogram of two merged MC observation

These datasets are also used to print the scatter plots of the datasets. This helps in determining the data if it is incomplete or inconsistent.

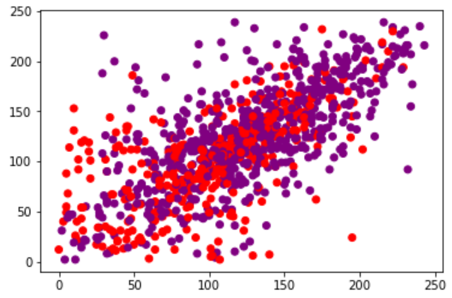


Fig. 10: Scatter plots of two merged MC observation

XI. Regression based model

Implementation and training with elastic-net regression is performed as a two-class classifier using the training sets of the overlapping and non-overlapping feature vectors (feature spaces) that is created. Application of the trained models to the test sets of all the categories of datasets is performed. Confusion matrices is developed for all categories of the datasets. These confusion matrices are built by converting the dataset to an array. This array is used as a matrix and matrix multiplication is performed followed by matrix inversion. This generates the confusion matrix. the labels are extracted from the dataset and are converted to confusion matrix. These two confusion matrices are used for the prediction of the accuracy, precision, sensitivity and specificity.

All the prediction is determined in both manual and inbuilt measures. We determine these values manually by using the true positive, false positive, true negative and false negative values.

For the non-overlapping feature vectors for two-class classification: the manual prediction values are:
Accuracy = 62.99
Precision = 77.51
Specificity = 69.39
Sensitivity = 59.38

The in-built prediction values:
Accuracy: 62.99
Precision: 77.51
Sensitivity: 59.38

For the overlapping feature vectors for two-class classification: the manual prediction values are:
Accuracy = 64.22
Precision = 78.52
Specificity = 59.68
Sensitivity = 72

The in-built prediction values:
Accuracy: 64.22
Precision: 78.52
Sensitivity: 59.68

For the non-overlapping feature vectors for multi-class classification: the manual prediction values are:
Accuracy = 61.60
Precision = 73.38
Specificity = 59.89
Sensitivity = 62.57

The in-built prediction values:

Accuracy: 42.27
Precision: 72.57
Sensitivity: 42.27

For the overlapping feature vectors for multi-class classification: the manual prediction values are:

Accuracy = 64.84
Precision = 76.51
Specificity = 66.09
Sensitivity = 64.11

The in-built prediction values:

Accuracy: 44.36
Precision: 81.1
Sensitivity: 44.36

XII. Implementation of random forest

The libraries sklearn.ensemble, keras.models, or keras.layers are used and the random forest technique. This is used for two-class and three-class classifiers using the training sets of the overlapping and non-overlapping feature vectors that was generated.

Training and testing models are used to create random forests.

By generating random forest, we predict the outcomes of the dataset. Similar to the prediction measures generated in elastic net prediction, we generate the predictions of the data using the true positive, false positive, true negative and false negative values from the confusion matrices. This also determines the feature importance of the entire dataset. Feature importance gives us the information of the important features that stores the important data.

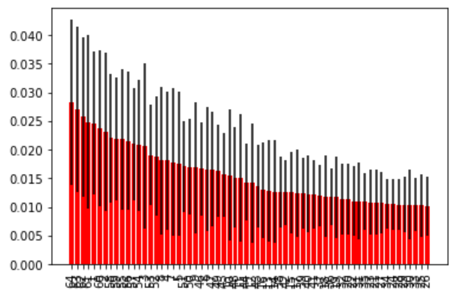


Fig. 11: Feature Importance for MC dataset

For the non-overlapping feature vectors for two-class classification: Accuracy: 87.45
Precision: 84.61

For the non-overlapping feature vectors for multi-class classification: Accuracy: 86.48
Precision: 93.50

For the overlapping feature vectors for two-class classification: Accuracy: 98.50
Precision: 97.98

For the overlapping feature vectors for multi-class classification:
Accuracy: 99.68
Precision:98.50

XIII. Evaluation of models

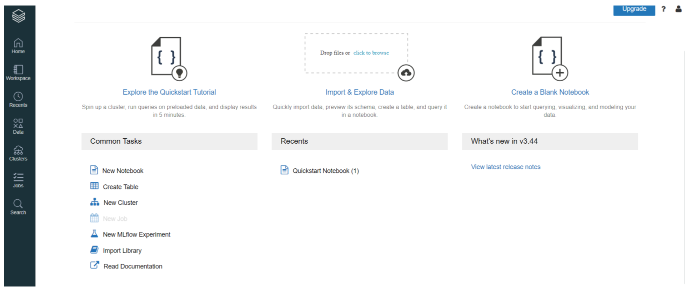
By evaluating the results of the models, we can conclude the results obtained by performing both elastic net regression and random forest model, Random forest give the most accurate results for the sliding window block features.

The feature importance gives the important information of the features. For a non overlapping two class classification,
Feature- 56 with 0.029 importance

For a non overlapping multi class classification,
Feature- 7 with 0.022 importance
For a overlapping two class classification,
Feature- 58 with 0.035 importance
For a overlapping multi class classification,
Feature- 62 with 0.028 importance
Clearly, Random forest for sliding window features gives accurate results compared to other results.

DataBricks

DataBricks is an open platform to perform data analysis and data related experiments. DataBricks is not a free platform, hence the students are encouraged to log in through community edition. In community edition, the user is limited to creating only 1 cluster. In a cluster, the user will be able to create tables that are used for the project. The tables are stored in the database. These tables can be accessed by using SQL commands. Below are steps to create the cluster and upload tables.



Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Quickly import data, preview its schema, create a table, and query it in a notebook.

Create a notebook to start querying, visualizing, and modeling your data.

Fig. 12: DataBricks Interface

- Once the user is logged into databricks, the user can create up to 1 cluster in the environment. Under the quick start, click on the create cluster option.
- On the top left, change the name of cluster to the preferred name.
- On the data bricks run time, check for 7.3 LTS ML(Scala 2.12, spark 3.0.0). Although latest versions are available, it is preferred to use basic version starting from 7.
- You are not required to make any changes for the intances and other information.
- Click on create cluster. This starts to create a cluster of the preferred runtime.
- Once the cluster is ready to use, it gives a green dot beside it. If not, it gives a buffering symbol beside the cluster name.
- Once the cluster is created, you are now able to create a data table inside the cluster.
- On the left tool bar, click on data option. This opens a two sided panel, one with the cluster name and the other with the table names.
- One the tables panel, click on create table. This opens the interface to create a table.
- Click on create table and upload the preferred data table you wish to enter to perform machine learning techniques.
- In this interface, you are able to choose the cluster you wish to further implement. After clicking on show preview, it gives options as ”First row is Header”, ”Infer schema” and etc.
- Click on create table with UI and the table is stored in the database.

On the home, you have an option to create a notebook. The notebook allows the user to perform the programming implementation. Importing pyspark is the first step in this implementation. The importing of tables is done by using sql commands as the data tables are stored in database. Later, we can follow the same steps and programming steps for performing the methods.

XIV. Examining the data

Once the data frames are loaded into the programming environment, they are examined if there are any outliers or if the data is consistent.

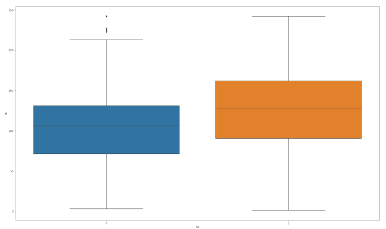


Fig. 13: Box plot of Merged MC dataset

The box plot of two class non overlapping classification says that there are some outliers in the data. There is not much difference in the median of the dataset.

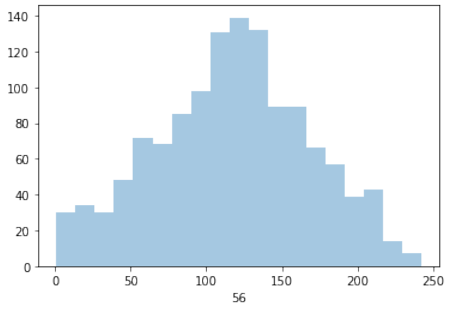


Fig. 14: Histogram of Merged MC dataset

The histogram gives the information on how the information is distributed. For one of the observation, the histogram plotted gives the information on how the data is distributed in that observation. In this histogram, the data is evenly distributed without outliers in the observation 56.

XV. Performing the Random Forest

Random forest technique is performed by dividing the data randomly in 80:20 ratio of training and testing datasets. First the training dataset is used to train the dataset. The test dataset is used for performing the prediction analysis. This gives the accuracy of prediction. These steps are performed on all the four different datasets. These four datasets are give different accuracy measures. By calling the auc() function and passing it the recall (x) and precision (y) values measured for each threshold, the region under the precision-recall curve can be approximated.

For a non overlapping two class classification,
accuracy = 85.82

For a non overlapping multi class classification,
accuracy = 99.59

For a overlapping two class classification,
accuracy = 99.59

For a overlapping multi class classification,
accuracy = 99.46

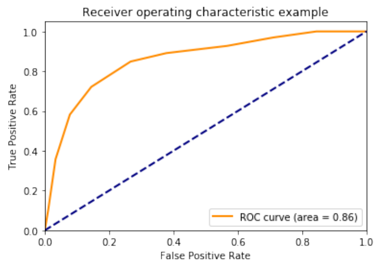


Fig. 15: ROC curve for two class classification.

As mentioned above, ROC curve for MC predicts the accuracy using the big data system.

XVI. Comparison of local system and DataBricks

Performing the random forest technique in the Spyder environment take more time than performing the same technique in Databricks environment. The accuracy scores are also different compared in both the systems. By performing the random forest in databricks interface, the accuracy is much higher which is almost equal to 100%. Since DataBricks is a big data environment, and when compared to companies, the datasets are smaller. Hence the compilation time in the big data environment for small datasets is fast.

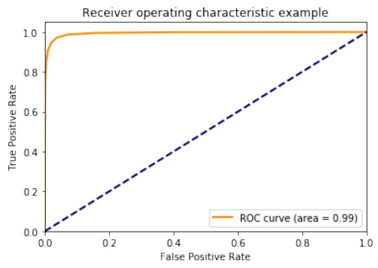


Fig. 16: ROC curve for sliding multi class classification.

The ROC curve for sliding window multi class classification shows the true positive vs true negative prediction which is 1. This implies that there is almost 100% accuracy.

References

[1] https://www.vicos.si/Downloads/FIDS30

Suthaharan, Shan. Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning. Springer, 2016.