# POLITECNICO DI TORINO

Master's degree course in Engineering & Management

Master's degree Thesis

# Application of image processing algorithms for the automatic assessment of workpiece quality

*Supervisor:*
Prof. Giulia Bruno
Prof. Franco Lombardi
Emiliano Traini

*Candidate:*
Krunal Viradia

JULY 2020

Please contact the author at the email address underneath:
S250071@studenti.polito.it

# Acknowledgements

First and foremost, I am highly grateful and indebted to my supervisor, Prof. Giulia Bruno. The help of Prof. Giulia Bruno was consistently open to me at whatever point I ran into obstacles or any difficulty with respect to the proposition or any related subjects. She reliably permitted me to have freedom to go on with my concept of proposal while controlling me the correct way at whatever point required.

I might likewise want to thank my Italian friends in Turin and furthermore Indian Friends in Turin-Milan for persuading and supporting me during the period. At last, I should offer my significant thanks to my folks and family for giving me nonstop consolation all through my degree and research with respect to the thesis. This would never be cultivated without the help of the referenced individuals. Much obliged to you!

KRUNAL VIRADIA

# Abstract

The Industry 4.0 or Fourth Industrial Revolution is causing a change of conventional manufacturing and mechanical practices, by adding smart IT innovation. This fundamentally centers around the utilization of large-scale machine to machine communication (M2M) and Internet of Things (IoT) arrangements to give expanded automation, improved communication and self-observing. Particularly, in this context, smart machines are developed, which can investigate and analyze issues without the requirement for human mediation. Casting products are important components and have a crucial role in large equipment.[1]

Casting is a manufacturing process in which defects may form while pouring the metal. Surface defects (e.g., blow holes, shrinkage, and metallurgical defects) are common in casting and they can be detected by visual inspection. However, since this is a time-consuming process, in industries with large production the quality control can be a crucial factor. Machine learning algorithms can be an aid to this problem by allowing time saving and neglecting human errors.

The aim of this thesis is to apply image processing and machine learning algorithms for the automatic assessment of the quality of a casting. A public dataset of casting images has been used as a case study. Three image processing techniques have been used (resized data, adaptive equalized data and HOG data), and different classification algorithms have been tested and compared.

Keywords: Image Processing, Machine Learning, Quality assessment, Classification Algorithms

# Contents

# List of Figures

# List of tables

# List of appendices

# 1. Introduction

Image Processing is a method which is used to perform operations on images to get enhanced image or to extract some information such as features or characteristics. Image processing can be divided into 3 steps: Importing the image, manipulating and analyzing, altered image as an output. Image processing can further be divided into 2 categories namely Analogue Image Processing and Digital Image Processing. Analogue Image Processing focus on hardcopies, photographs and printouts while Digital Image Processing focus on working digital images with the help of computers. Its applications range from entertainment to medicines, passing by remote sensing and geological processing. Various Image processing algorithms can be applied to various sets of images to extract useful information.

In this study, image dataset from Kaggle have been used. The dataset contains 2 folders with train and test data followed by defective and ok as subfolders in each main folder. All the images are captured from the top view of a casting manufacturing product. Casting is a manufacturing process in which liquid molten metal is poured into the mold be it sand casting, investment casting, permanent die casting etc. to get the desired shape and later allowed to solidify. The main defects that casting face are blow holes, shrinkage, burr, pinholes, mold material defects etc. Visual inspections are held by quality department which is very time consuming where the finished products are in very high numbers. In order to get this inspection fast, an image processing algorithm can be used along with various machine learning classifiers to train the models which can later be used to find the defective castings.

The experiments are done by utilizing supervised classifying algorithms in Python language on dataset. Also, Jupyter notebooks and Google Colaboratory notebooks have been used equally.

Various Classifiers like Support Vector Classifier(SVC), Multilayer Perceptron(MLP), K-Nearest Neighbor(KNN), Decision Tree, Logistic Regression have been used in this study to calculate the accuracies of models. Training of model is carried out after applying three different image processing techniques: Resizing, Adaptive Histogram Equalization, Histogram of Oriented Gradients. Hence three datasets for each classifier are used to train the models and classification reports are generated to get the best suitable classifier.

# 2. Related Work

In the following a brief review of the state of the art is performed based on the topics related to the thesis. Three areas will be covered: the first section includes papers regarding casting defects. In the second section some of the image preprocessing techniques. Finally, some examples of how Machine Learning algorithm are used in classifying defectives and good castings are analyzed to provide context and a base for future development. In spite of the fact that the papers inspected in this section were utilized in the thesis, a lot of study material additionally originated from distributions, for example, books and manuals, which are not secured here being already well established as working procedures.

## 2.1. Casting Defects:

Casting industries has variety of operations which may become a bottleneck for defect formation. Various steps are carried out by skilled labors which can be mentioned as follows:

1. Sand Preparation
2. Mold Making
3. Pouring
4. Shakeout

Cause and Effect Diagram is one of the ways to deal with list the potential causes. Figure 1, 2 and 3 are the cause and effect diagrams of Mold Shifting, Shrinkage and Surface Finish respectively. [2]

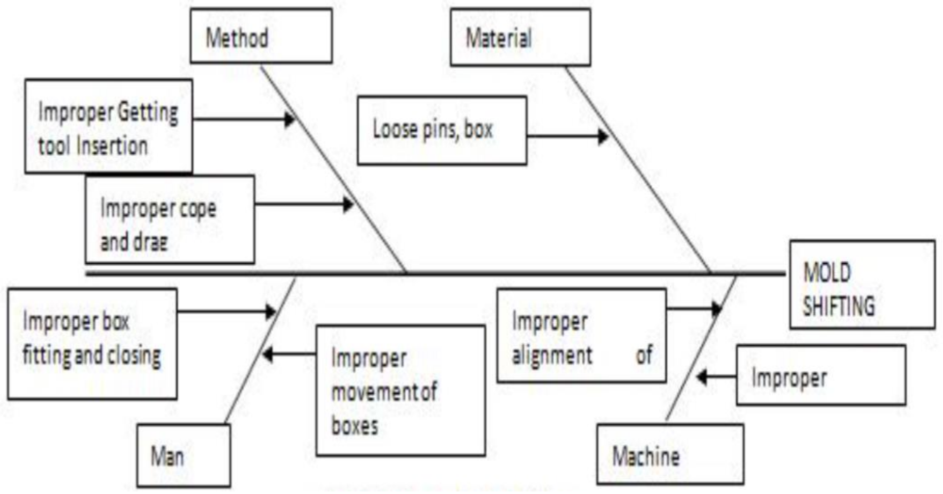


Figure 1 – c.e.d for mold shifting

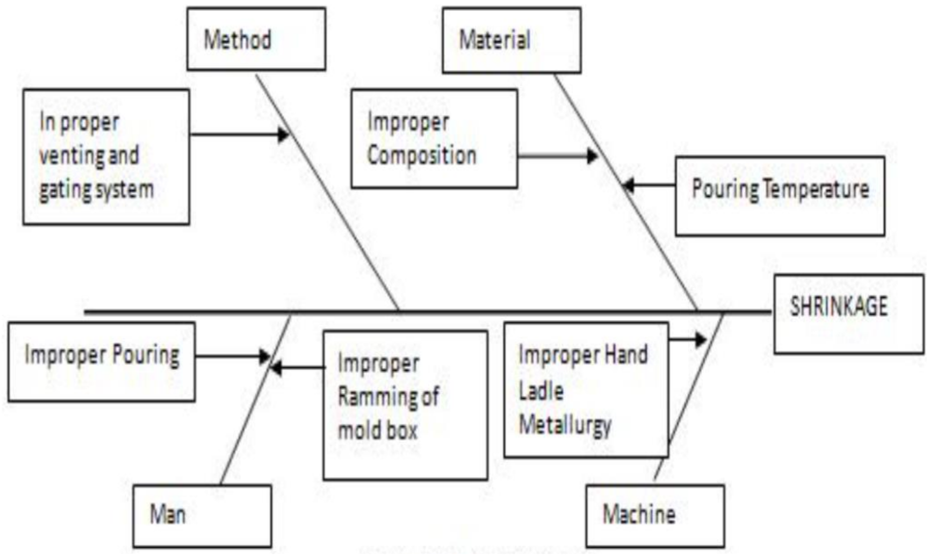

Method
Material
In proper venting and gating system
Improper Composition
Pouring Temperature
SHRINKAGE
Improper Pouring
Improper Ramming of mold box
Improper Hand Ladle Metallurgy
Man
Machine

Figure 2 - c.e.d for shrinkage

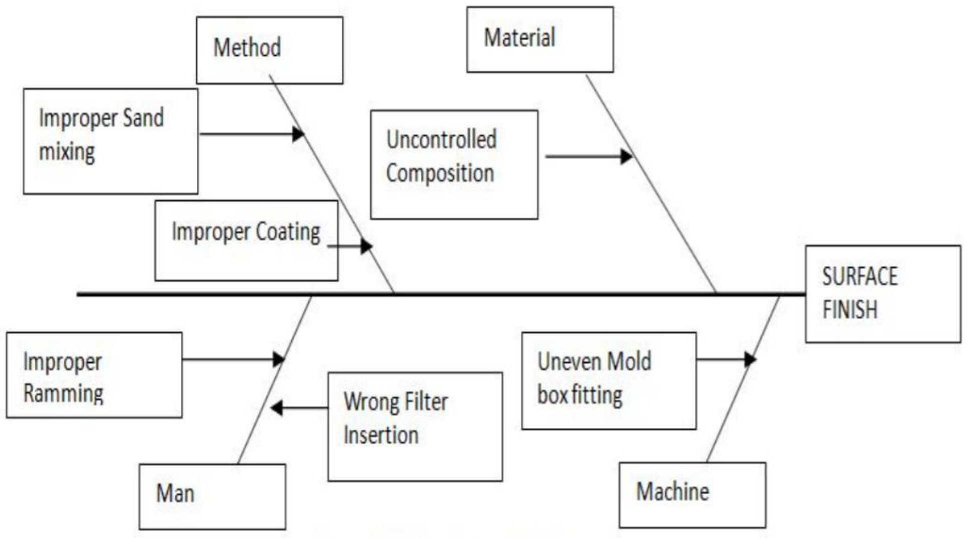


Figure 3 - c.e.d for surface finish

Any abnormality in the molding procedure causes defects in castings which may at some point be endured, at some point eliminated with appropriate machining process or fixed using welding and metallization.

**Gas** defects can be classified into open blows and blow holes, pin hole porosity and air inclusion. This defect can appear in all regions of the casting. Open Holes can be seen in Figure 4.

**Shrinkage** defects occurring due to volumetric contraction in both liquid and solid state can be seen in Figure 5. It may be a result of poor casting design.

**Metallurgical defects** such as hot tears (Figure 6) and hot spots may form due to some reasons mainly being poor casting design, rough handling or high temperature at shakeouts. [3]

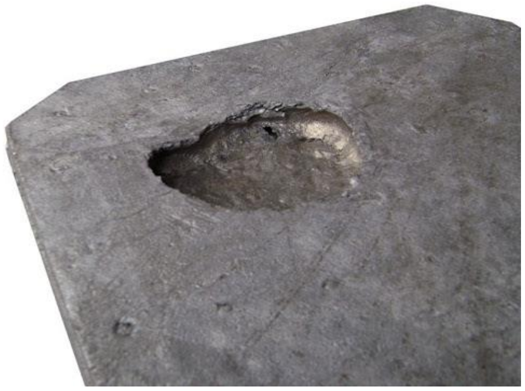


Figure 4 - open holes

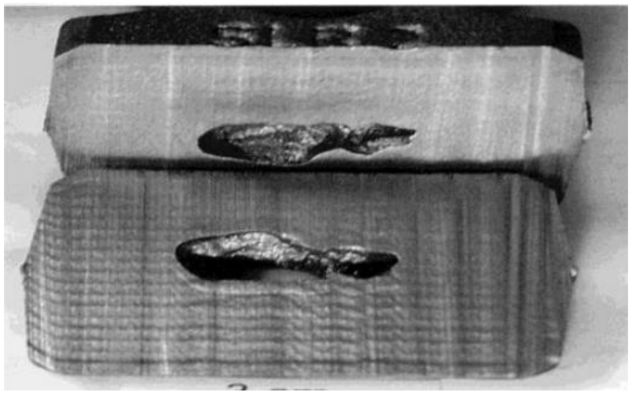


Figure 5 – shrinkage

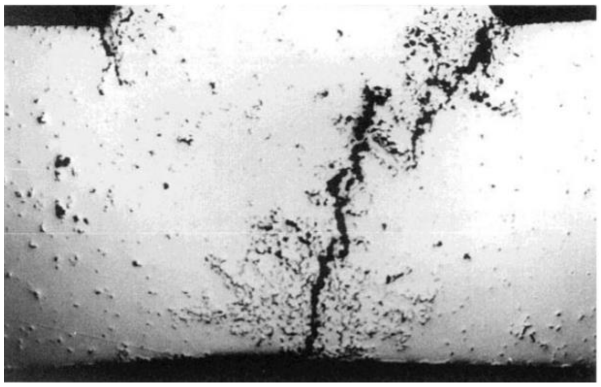


Figure 6 - hot tears

## 2.2. Image preprocessing:

Generally, visual inspection and quality control are performed by individuals. Despite the fact that individuals are very great at this assignment and at times far superior to machines, they are much increasingly slow and cannot work for significant stretches of time as their eyes get drained and need to relax. In numerous applications, data must be rapidly and repetitively separated, handled and decisions must be made. Advances in image processing innovation are making new points of view in expanding profitability, quality and proficiency in a wide scope of modern applications.

As per [4], a vision system has become an essential segment of advanced manufacturing systems ( Figure 7), for two fundamental reasons. Initially, it gives a method for controlling quality during manufacturing of goods, and secondly, robotic assemblies can be given the vital data so as to collect complex items from a lot of essential components. In specific conditions inspection can be hazardous or troublesome. Machine vision can replace human review in such cases. Automated assembly lines make adaptable assembling a reality, therefore costs due to underuse of production lines are drastically reduced.

The most well-known utilization of visual inspection is to confirm the quality of products and take activities for detailing and adjusting these faults and replacing or expelling defective parts from the production line.

In general, following sequence of steps are carried out in industrial inspection process:

1. Image acquisition: In vision everything relies upon acquisition of image. Any inefficiencies of initial images can have genuine implications in picture analysis and translation.

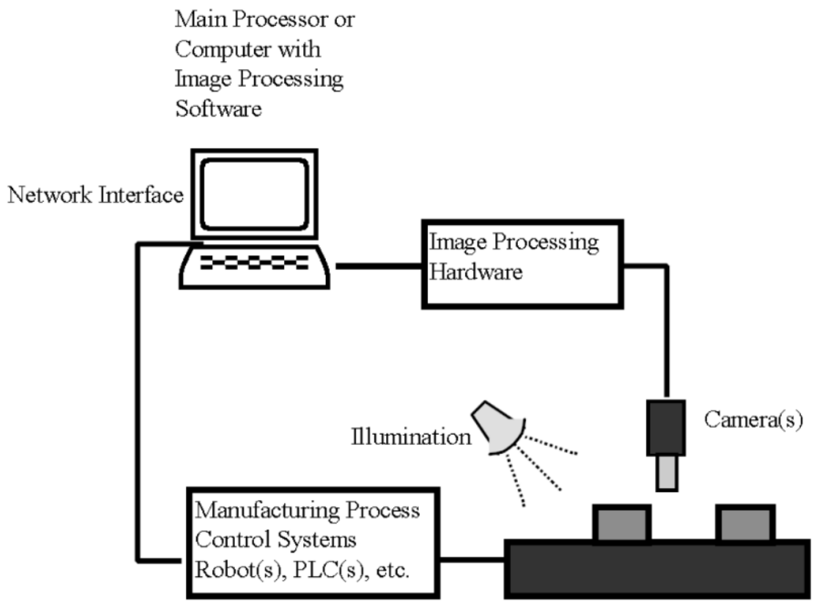

Main Processor or Computer with Image Processing Software
Network Interface
Image Processing Hardware
Illumination
Camera(s)
Manufacturing Process Control Systems Robot(s), PLC(s), etc.

Figure 7 - general industrial vision system

2. Image processing: When pictures are obtained, they are set up for the subsequent activities, by removing noise, or non-uniform lighting. The main operations of image processing performed in this work can be seen from the below given Table 1.

3. Segmentation: This step attempts to segment the picture into locales of interests that correspond to part or entire objects inside the scene.

4. Feature extraction: Feature extraction is a type of dimensional reduction. It includes decreasing the measure of data required to describe a large arrangement of information effectively. Instances of features incorporate size, position, shape estimation and surface location. These qualities can be extracted and analyzed utilizing statistical, structural, block matching, neural networks, or fuzzy systems. The arrangement of processed features frames the portrayal of the input image.

5. Decision-making: The initial phase in the dynamic procedure is the decrease of the element space to an inherent dimensionality of the issue. The decreased list of features is additionally prepared so as to arrive at a decision. The choice, just as different kinds of estimations or features processed, are application dependent. If there should arise an occurrence of visual investigation, the framework needs to choose if the consequence of assembling satisfies the quality guidelines, by coordinating the figured features with a known model. The model can be either explanatory or procedural.

| Point operations | Global operation | Neighborhood operation | Geometric operation | Temporal operation |
|---|---|---|---|---|
| Brightness modification | Histogram equalization | Image smoothing | Display adjustment | Frame-based operations |
| Contrast enhancement | _ | _ | Image wrapping | _ |
| Negation and thresholding | _ | _ | Magnification and rotation | _ |

TABLE 1 - GENERAL OPERATIONS PERFORMED IN IMAGE PROCESSING

The range of mechanical uses of image processing is very liberal. The industries that advantage from vision frameworks go from military, to medication and from food industry to automotive.

As shown in Figure 10, the improved calculation holds dispersion attributes of the picture, the grayscale merger is additionally diminished, and the picture quality is extraordinarily improved than the traditional algorithm. Histogram equalization utilizes a monotonic and a non-straight mapping which reassigns the pixel intensity values in the input picture so that the output picture has a uniform distribution of intensities(a level histogram), and along these lines improves the complexity of the picture.

After applying histogram equalization, the histogram obtained becomes almost uniform (pixel values represented on X axis and frequencies on Y axis)which can be seen in Figure 9, but on the other hand adaptive histogram equalization reveals more details of the image when compared to normal histogram equalization. The following screenshots shows distribution of pixels changing for local(near-uniform) versus adaptive histogram equalization and the image enhancement pictures using two different processing techniques. Figure 11 is yet another example of adaptive Equalization. [5]

The comparison of overall performance of HOG descriptors is compared with some other methods in[6]. Rectangular (R-HOG), Circular (C-HOG) based detectors are compared with PCA-SHIFT, Haar wavelet and shape context approaches. From the results performed on MIT (Figure 12) and INRIA (Figure 13) datasets, HOG outperformed wavelet on greater margin, PCA-SIFT and Shape Context ones, giving close ideal

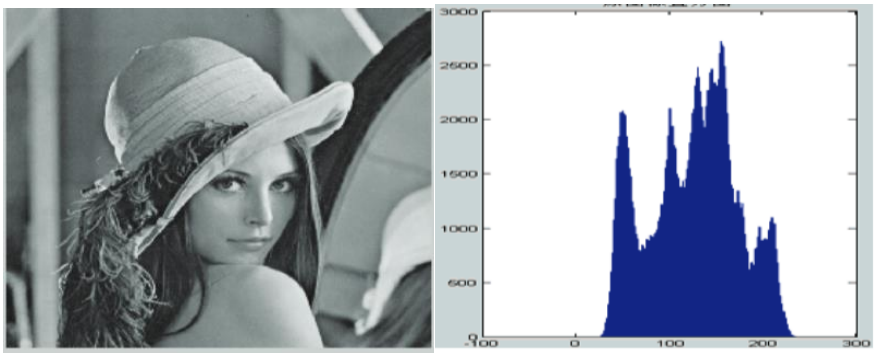


FIGURE 8 - ORIGINAL IMAGE AND HISTOGRAM

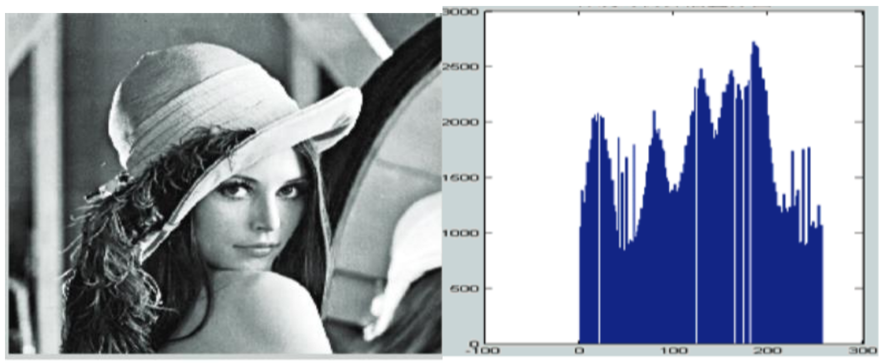


Figure 9 - conventional algorithm and histogram

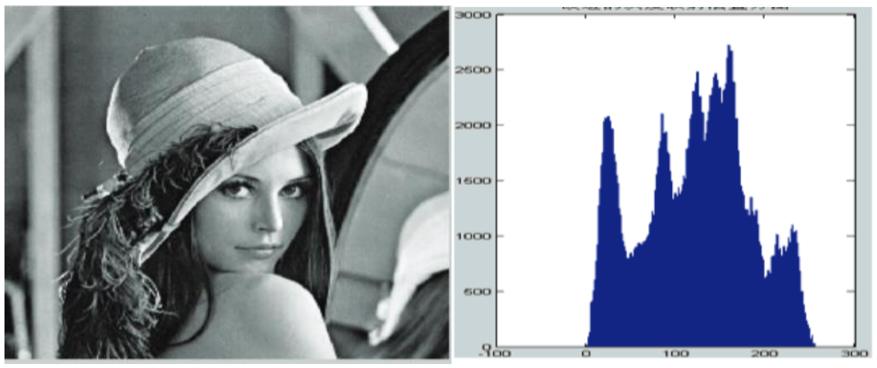


Figure 10 - improved algorithm and histogram

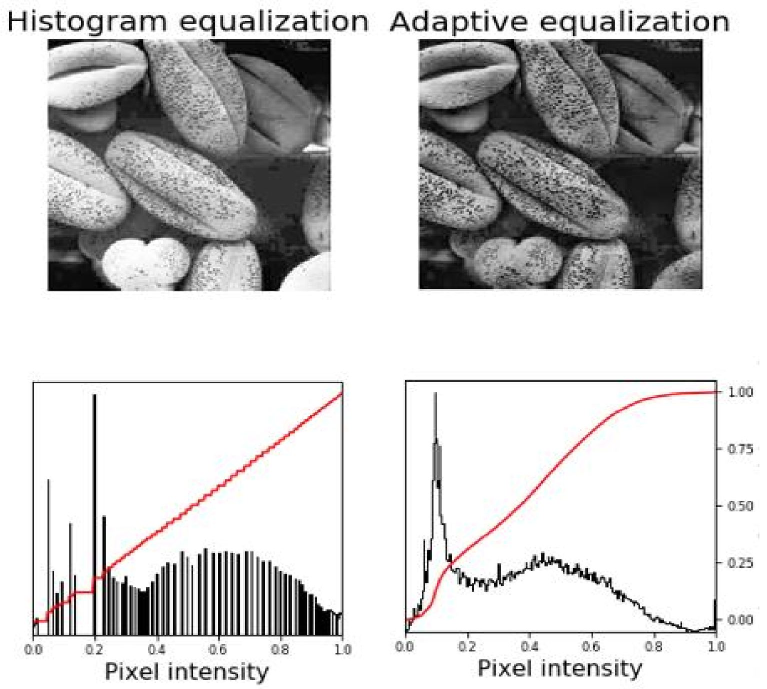


Figure 11 – Equalization

detachment on the MIT test set and in any event a significant degree decreases in FPPW on the INRIA one.

The performances of the final rectangular (R-HOG) and circular (C-HOG) detectors are fundamentally the same as, with C-HOG having the slight edge. Expanding R-HOG with primitive bar finders pairs the element measurement, however, also further improves the exhibition . Replacing the direct SVM with a Gaussian kernel, one improves execution at the expense of a lot of higher run times. Utilizing binary edge voting (EC-HOG) rather than gradient magnitude weighted voting (C-HOG) diminishes execution, while discarding direction data diminishes it by considerably more, regardless of whether additional spatial or outspread bins are included, for the two edges (E-ShapeC) and slopes (G-ShapeC).

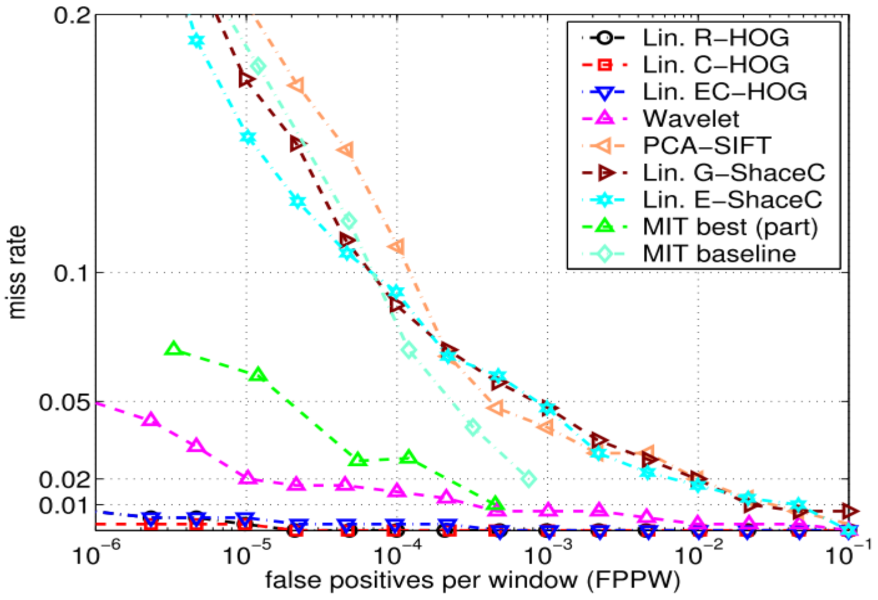


FIGURE 12 - DESCRIPTORS ON MIT DATASET

As per [7] algorithm can be framed into following steps:

1. Image can optionally be normalized globally
2. Computing gradient images horizontally and vertically
3. Computing gradient histogram
4. Normalize across blocks
5. Flattening into a feature descriptor vector

First stage can be used to reduce the influence of illumination effects. The second computes image gradients of first order

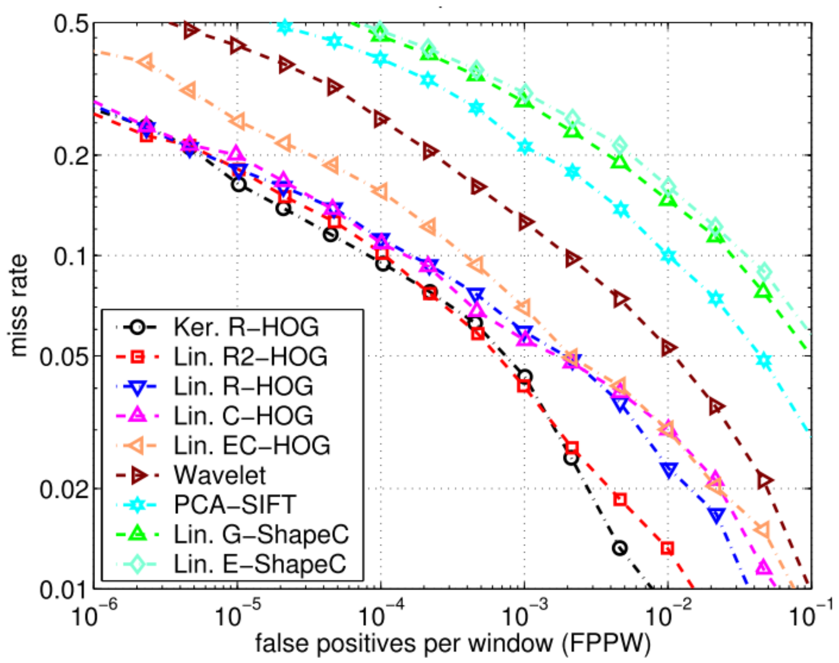


FIGURE 13 - DESCRIPTORS ON INRIA DATASET

which capture silhouette, contour, and some information regarding texture, while providing further resistance to illumination variations. The third stage aims to create an encoding that is sensitive to nearby picture content while staying resistant to little changes in posture or appearance.

The image window is separated into little spatial locales, called "cells". For every cell it gathers a nearby 1-D histogram of angle or edge orientations over all the pixels in the cell. This joined cell-level 1-D histogram shapes the fundamental "orientation histogram" representation. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are utilized to cast a vote into the orientation histogram.

The fourth stage figures standardization, which takes neighborhood groups of cells while contrast standardizes their general reactions before going to next stage. The last step gathers the HOG descriptors from all blocks of a dense overlapping matrix of blocks covering the detection window into a joined feature vector for use in the window classifier which can be seen in Figure 14.

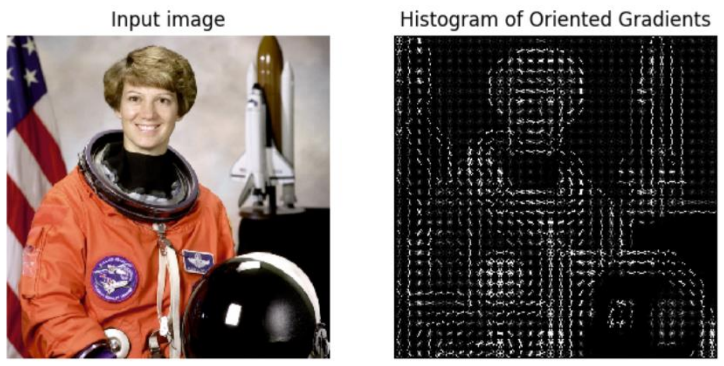


Figure 14 - hog descriptors

## 2.3. Machine Learning Algorithms:

***Supervised Machine Learning*** contains dataset with known class labels, which can be used to build classification models. Supervised learning is where we have input variables (x) and an output variable (Y) and we learn the mapping function using an algorithm, from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when we have new input data (x), we can predict the output variables (Y) for that data. It is called supervised learning on the grounds that the procedure of an algorithm gaining from the training dataset can be thought of as a teacher administering the learning procedure. We know the right answers, the algorithm iteratively makes predictions on the training data and is revised by the instructor. Learning stops when the algorithm accomplishes an adequate degree of performance. Supervised learning can be categorized into Regression and Classification:

1. Classification is when the output variable is any category like "banana" or "apple" , "cat" or "dog".
2. Regression is when the output variable is real value like "length" or "Rupee". [8]

The classification techniques that will be discussed about in this segment are those centered around anticipating a subjective reaction by breaking down information and recognizing designs. There are many different classifiers or classification techniques, but some of the widely used ones include:

1. Logistic regression.
2. Support vector machines.
3. K-Nearest Neighbors.
4. Multilayer Perceptron(MLP).
5. Decision trees. [9]

Author in this study used four different classifiers : Naïve Bayes for ML, Random Forest (RF), Multi-Layer Perceptron in case of ANN, and LibSVM for Support Vector Machine. For the latter part, radial-basis-function (RBF) and polynomial kernels (POLY) were used. GridSearchCV was used for parameter optimization. The resulting values were applied to parameter search to get best results. ANN was optimized using two different steps : parameter's search and choice of architecture. Classifiers were iterated 100 times splitting randomly training and testing set. All the results including kappa, overall accuracy and testing and training times were exported to csv file. The overall accuracy was then analyzed utilizing measurable parameters such as mean and standard deviation. 5-fold cross-validation was used to classify the training data. Producer's accuracy(PA), user's accuracy(UA) and F-Measure (below given equation) were used for quality assessment.

$$F = \frac{2 \bullet PA \bullet UA}{PA + UA}$$

Overall classification accuracy shifted significantly among the classifiers utilized(Figure 15). Besides, the number of images used for the classification significantly influenced accuracy. SVM-RBF showed the most noteworthy mean overall accuracy. SVM-RBF and SVM-POLY delivered about indistinguishable outcomes. Outstandingly lower accuracies were noticed using ANN and RF. As indicated by the t-test the distinctions were factually noteworthy between SVM, ANN, and RF. With just 45 % in general precision, ML displayed the worst outcome, which were additionally different from the other ones.[10]

As per [11], the performance of three machine learning algorithms was studied with the designed word similarity and word overlap. Initially, the three classifiers SVM, k-NN and MaxEnt were trained and tested with the word overlap feature set. Although all three classifiers used the same attributes, the results were different due to their varied machine learning philosophy. The best results were achieved by SVM. MaxEnt and k-NN algorithms gained 68.29% and 63.36% accuracy. Comparing these results to a baseline that counts the number of common words, only k-NN could not outperform it. Figure 16 shows the Accuracy, Precision, Recall and F scores for different classifiers.

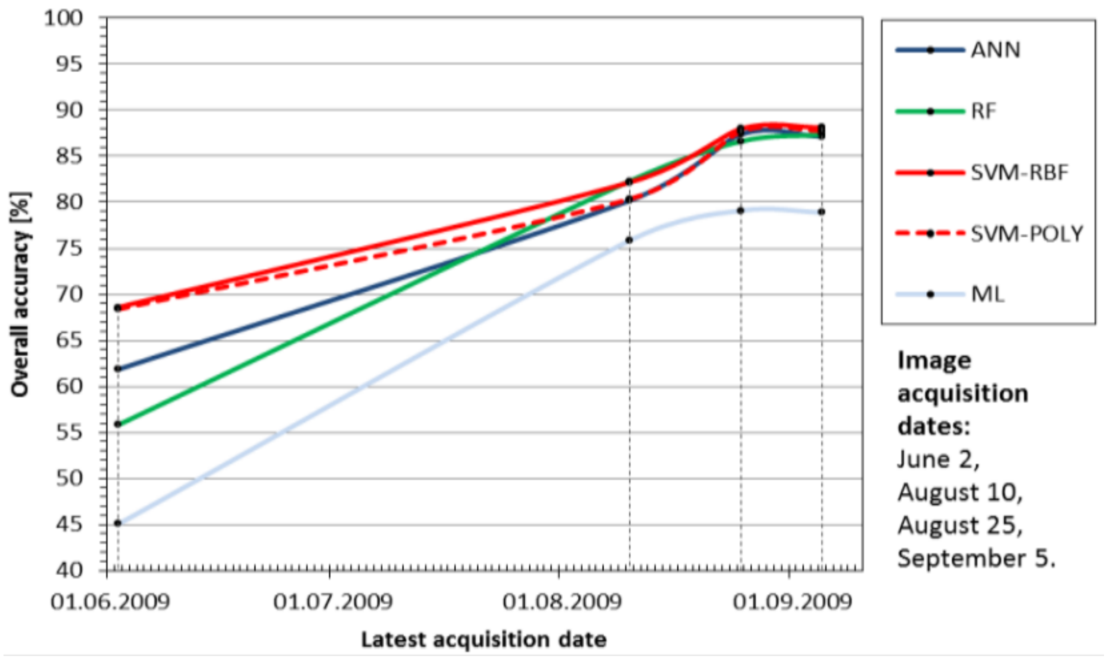


Figure 15 - mean overall accuracies of all classifiers

| System | Acc. | Prec. | Rec. | F-score |
|---|---|---|---|---|
| SVM | 69.86 | 93.46 | 70.66 | 80.48 |
| MaxEnt | 68.29 | 69.16 | 59.53 | 63.98 |
| k-NN | 63.36 | 74.45 | 71.58 | 72.99 |
| C-M | 68.80 | 74.10 | 81.70 | 77.70 |
| word match | 66.10 | 72.20 | 79.80 | 75.80 |

Figure 16 - word overlap calculations

In order to improve overall accuracy of the study presented in [12], datasets were divided into 10-fold cross-validation methodology. Each dataset was divided into 10 subsets of equal parts containing 50% of absence and 50% of presence data. The performance of each classifier was given by the average of the performances observed in the test folds. The AUC (Area Under the ROC Curve) was used to evaluate the classifiers effectiveness in the classification of presence/absence data. For calculating SVM, LibSVM tool was used. Weka tool was used for other ML classifiers.

The AUC performance of all the ML classifiers were compared using Demsar(2006). This comparison considers the exhibition of the various methods in different datasets in general and a ranking network is constructed containing the datasets as lines and the techniques as segments. In this network, a component $r_{ij}$ represents the position of technique *j* execution in dataset *i*. Line by line, rank of the algorithm execution is made. The most appropriate AUC method gets the primary spot in the position, the second higher AUC procedure gets the subsequent spot, etc. The mean and standard deviation AUC values of each classification technique on the 35 datasets were recorded and were presented in a table. RF was the best performing outranking all the other ML techniques SVM. KNN, DT, NB were the worst performers.

In the starting datasets LR was performing good but on later stages it fumbled explaining its instability. The low performance of DTs can be credited to the trouble of symbolic methods in managing continuous-valued properties. Nevertheless, the comprehensiveness of the models created by these procedures can be a decent argument toward their thought in ecological analysis. KNN indicated the worst outcomes. It would be fascinating to check if the use of other distance measures between information things would change these results, although a clear disadvantage of KNN in relation to the other techniques is its lack of an explicit model, which additionally brings about a high computational expense for making predictions.

The statistical comparison of the methods is summed up in Figure 17. This figure presents a size of the techniques ranking. Best performing strategies lay on the right of the scale, while worst performing techniques are in the left of the scale. CD corresponds to the critical difference interval of the test. In the event that the ranks of two methods vary by at most CD, their outcomes can be considered statistically similar at 95% of certainty. Those methods with similar factual outcomes are participated in Figure 17 by a thick flat line.

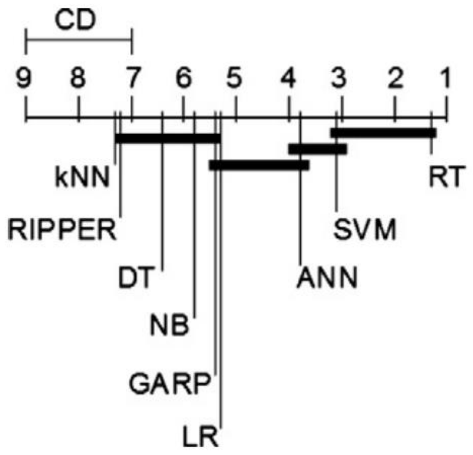


*FIGURE 17 - RESULTS OF STATISTICAL COMPARISON OF TECHNIQUES*

# 3. Data

The public dataset that I am utilizing here, contains images of the casting manufacturing process. Casting is a process which can be carried out using different methods like Investment Casting, Die Casting, Sand Casting, Permanent Mold Casting, Shell Molding etc..[13]. In the casting process, metal is melted at high temperature according to requirement and then poured into the hollow cavity of the mold which depends on the type of casting. It is then allowed to rest at room temperature or even cooled using water depending on the desirable properties of the product.

Many types of casting defects can be seen in this product like blow holes, pin holes, gas bubbles, shrinkage, pouring metal defects, metallurgical. The dataset that I am going to work on in this study contains images of the casting manufacturing process.

Defects are undesired factors in casting and hence a quality inspection department is provided to rule out the defects. This quality inspection is carried out manually and hence it is time consuming where thousands of products. Also, human error can be a considered factor which can lead to the rejection of the orders claiming a big loss to the industry. Machine learning classification models can be an aid to this problem.

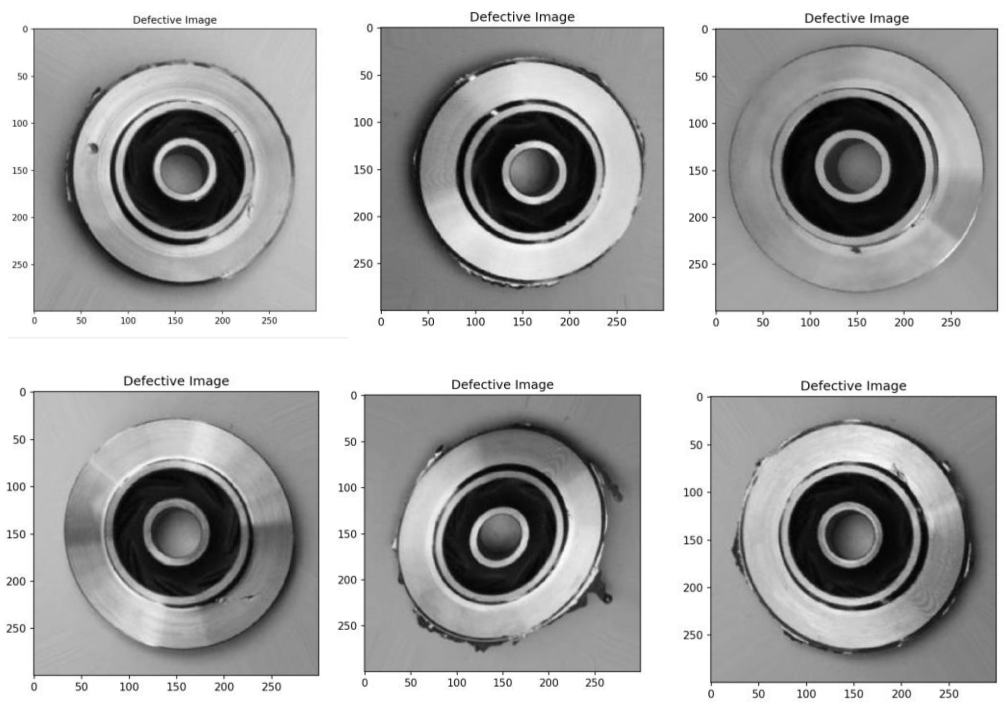


Figure 18 - defective images

All the images are the top views of the submersible pump impeller. Image acquisition was done through proper lighting. Dataset contains total 7348 images of 300*300 pixels gray scaled images. Images are divided into two folders : TEST and TRAIN. Each folder contains two subfolders : “def”(Defective) and “ok” with defective casting images (Figure 18) and good condition or ok casting images (Figure 19) respectively. TRAIN folder contains 3758 images in def folder and 2875 images in ok folder whereas TEST folder contains 453 in def folder and 262 in ok folder.

Now as we can see the images in each folder are not equal and hence the data is unbalanced. In order to balance the data two techniques can be applied: Downsampling and Upsampling. Upsampling randomly duplicates the images from the minority class while Downsampling randomly removes the images from the majority class[14]. I have applied Downsampling here and hence TRAIN folder contains 2875 images in both def and ok folder.[15]

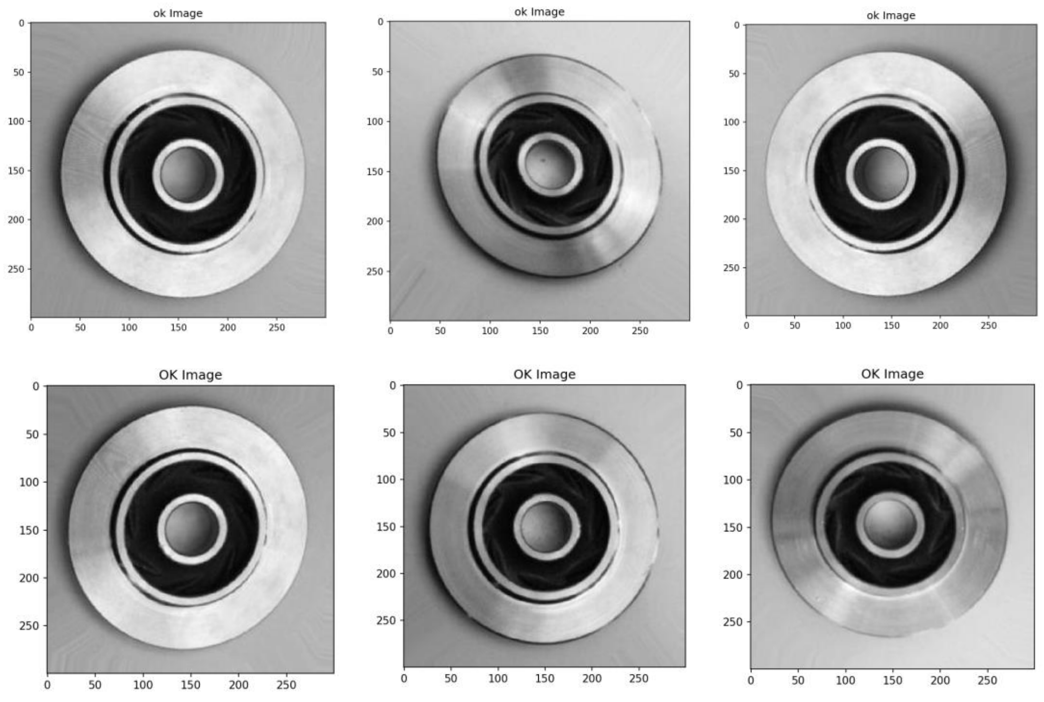


Figure 19 - ok images

# 4. Method

## 4.1. Pre-processing and experimental datasets

In this study three different types of experimental datasets or training datasets have been considered in order to get the best scores. Cross validation technique has been used to get the best parameters suitable for the working datasets. Hyper parameters range for different classifiers were set based on research for the particular classifiers.

Learning the parameters of a predicting function and testing it on similar information is a methodological mix-up: a model that would simply repeat the labels of the samples that it has quite recently observed would have an ideal score however would neglect to foresee anything valuable on yet-unseen data. This circumstance is called overfitting. To avoid it, it is a basic practice when performing out a (Supervised) machine learning to hold out piece of the accessible data as a test set X_test, y_test. A typical cross validation workflow can be best explained through the following flowchart (Figure 20) in model training.

The best parameters can be determined by grid search procedures. While evaluating different hyper-parameters for estimators, for example, the C setting that must be physically set for a SVM, there is as yet a danger of overfitting on the test set in light of the fact that the parameters can be changed until the estimator performs ideally. To take care of this issue, one

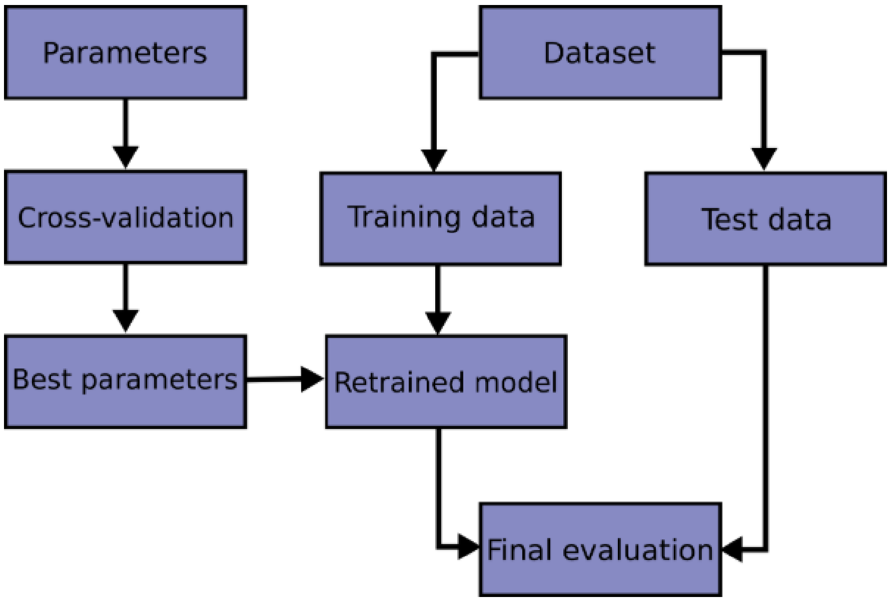


FIGURE 20 - FLOWCHART OF CROSS VALIDATION WORKFLOW

more piece of the dataset can be held out as an alleged "validation set": preparing continues on the training set, after which assessment is done on the validation set, and when the examination is by all accounts fruitful, last evaluation can be done on the test set. In any case, by dividing the accessible data into three sets, we definitely diminish the quantity of tests which can be utilized for learning the model, and the outcomes can rely upon a specific arbitrary decision for the pair of (train, validation) sets.

An answer to this issue is a strategy called cross-validation (CV). A test set should at present be waited for final assessment, yet the validation set is not required while doing CV. In the fundamental methodology, called k-fold CV, the training set is split into k smaller sets which can be seen in Figure 21. The following strategy is followed for each one of the k "folds": A model is trained using *k-1* of the folds as trained data, the resulting model is validated on the rest of the part of the data. The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This methodology can be computationally costly, however doesn't waste an excessive amount of information (similar to the situation when fixing an arbitrary validation set), which is a significant advantage in issues such as inverse inference where the quantity of tests is small.[16]

First Training dataset contains data which is resized to 128*128 pixels and are gray scaled images. In some cases, higher pixels results in more features and may lead to overfitting. A for loop is run for all the images in the train folders. Figure 22 and Figure 23 represents the defective and ok resized images, respectively.

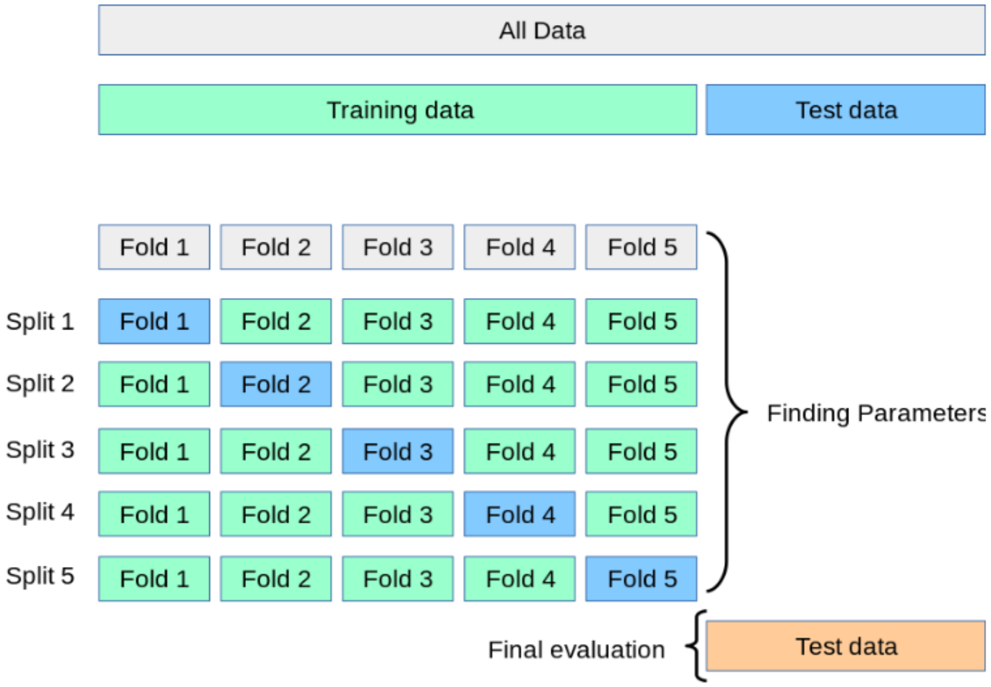


FIGURE 21 - K-FOLD CROSS-VALIDATION

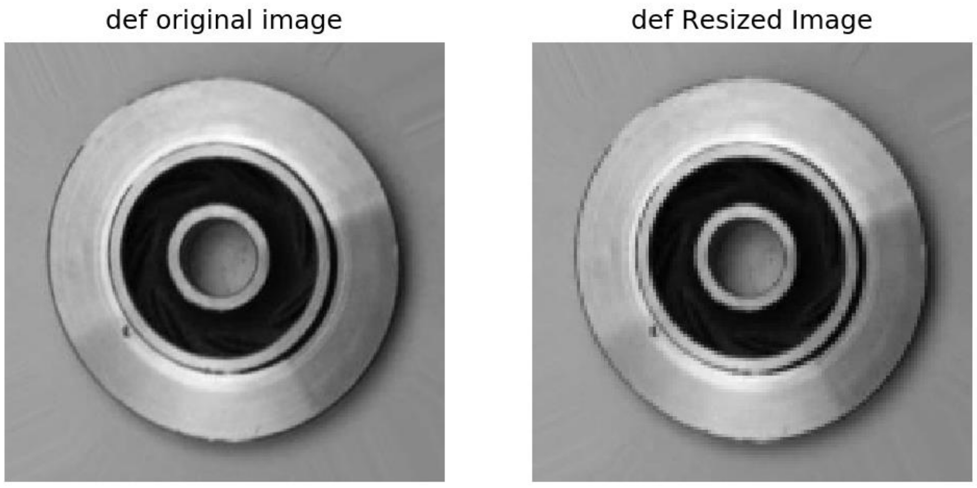


FIGURE 22 - DEFECTIVE RESIZED IMAGES

Second Training dataset contains images which are first being resized and then Histogram of Gradient(HOG) is used for feature descriptors. Figure 24 and 25 represent defective HOG and ok HOG images, respectively.

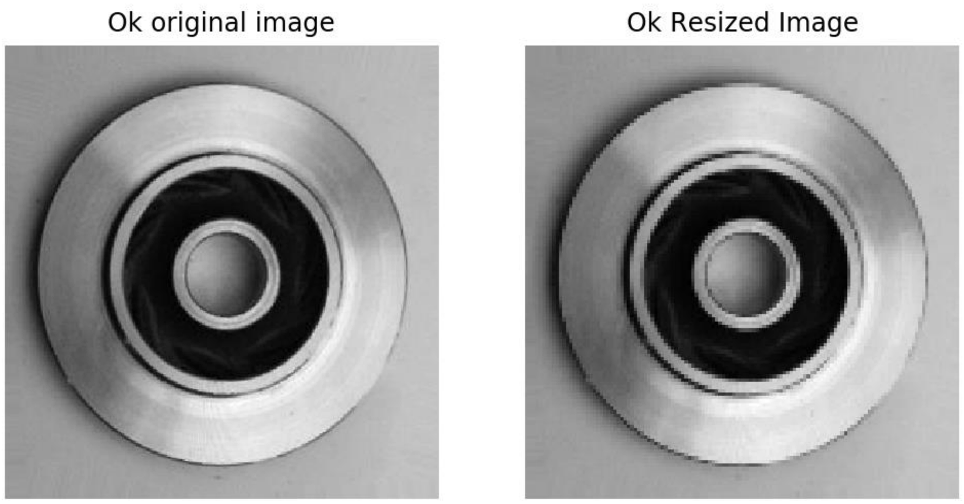


FIGURE 23 - OK RESIZED IMAGES

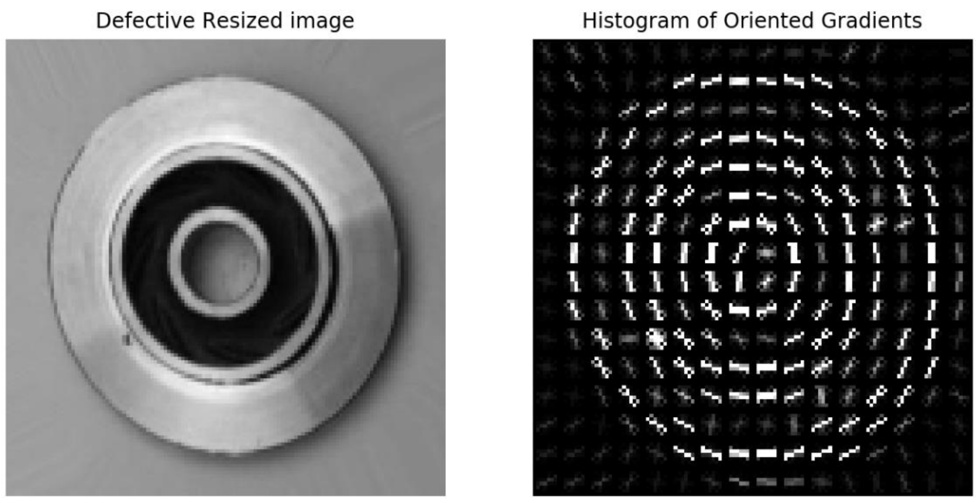


FIGURE 24 - HOG OF DEF RESIZED IMAGES

Third Training dataset contains images which are first resized to 128*128 pixels and then Adaptive Equalization preprocessing technique has been applied. Figure 26 and 27 represents defective and ok resized adaptive equalized images, respectively.

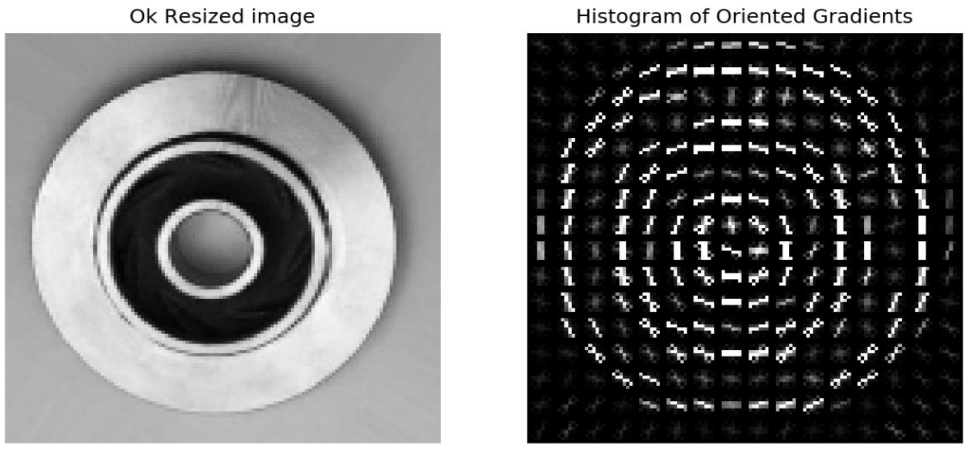


FIGURE 25 - HOG OF OK RESIZED IMAGES

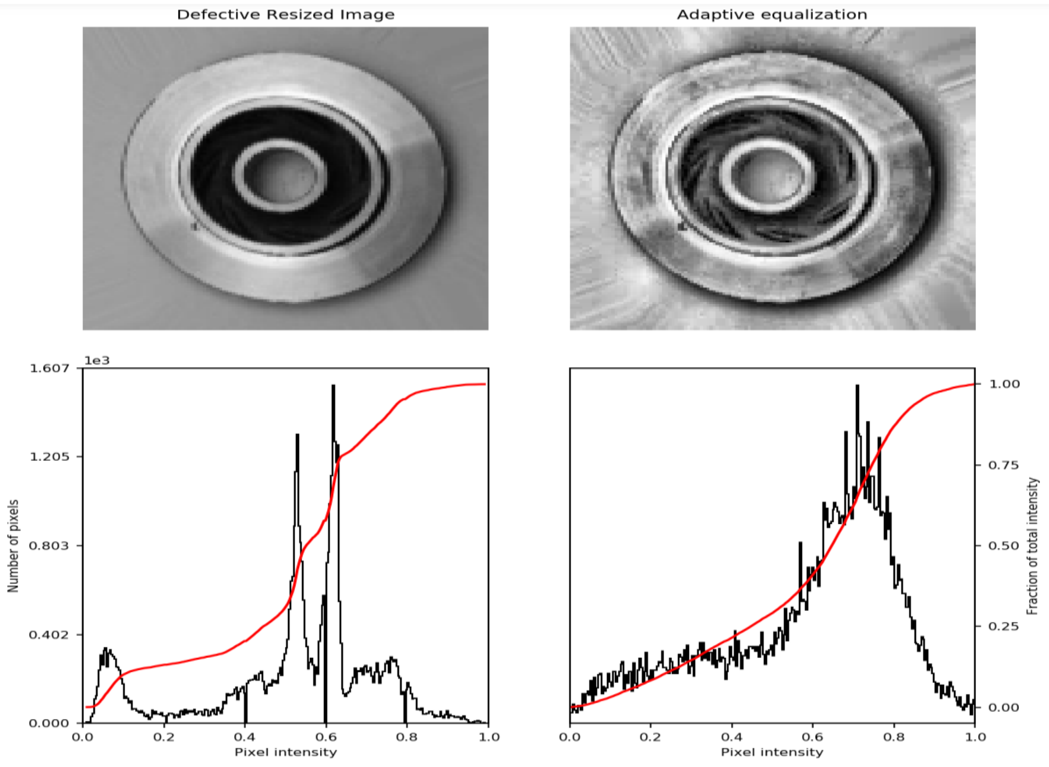


FIGURE 26 – ADAPTIVE EQUALIZATION OF DEFECTIVE RESIZED IMAGES

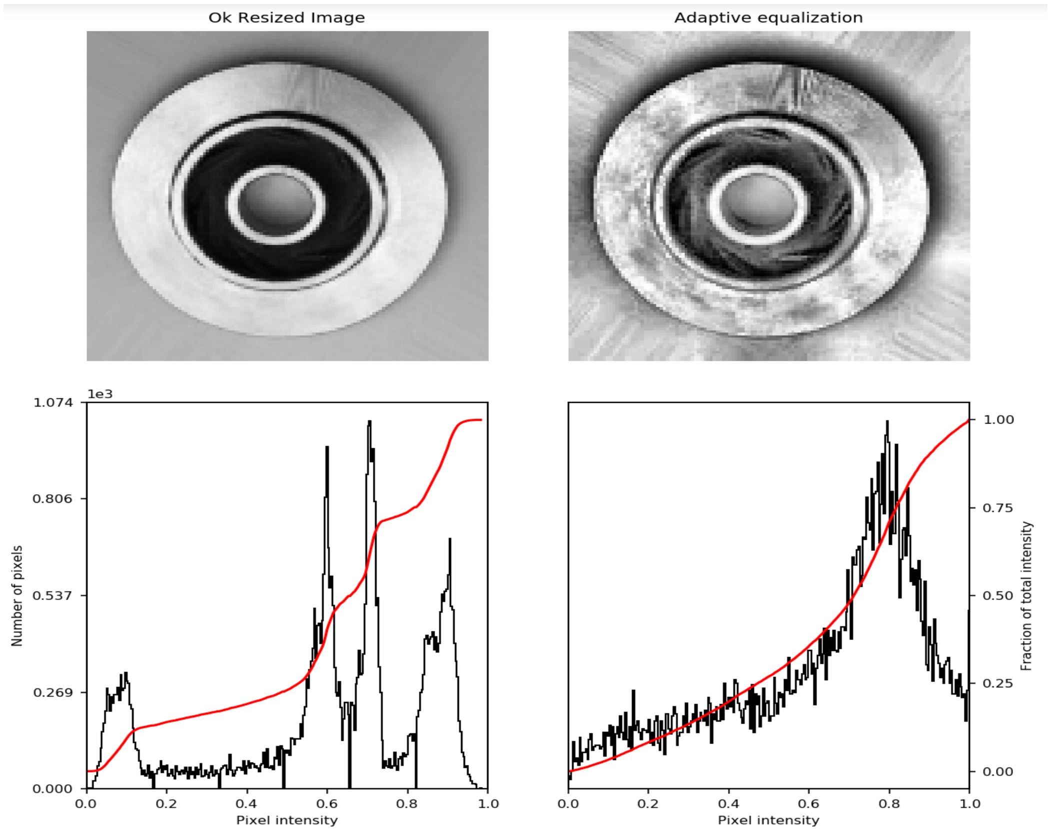


Figure 27 - adaptive equalization of ok resized images

## 4.2. Classification

Classification is a procedure of sorting a given arrangement of data into classes. It can be performed on both organized and unstructured data. The procedure begins with anticipating the class of given data focuses. The classes are frequently alluded to as target, label or categories. The classification predictive modeling is the task of approximating the mapping capacity from input factors to discrete yield variables. The principle objective is to distinguish which class/classification the new data will fall into.

Heart disease discovery can be recognized as a classification problem. This is a binary arrangement since there can be just two classes i.e. has heart illness or does not have heart illness. The classifier, for this situation, needs preparing information to see how the given input variables are identified with the class. When the classifier is prepared precisely, it tends to be utilized to distinguish whether heart disease is there or not for a specific patient. Some terminology used in machine learning classification are as follows:

1. Classifier : It is an algorithm that is utilized to map the input information to a particular class.
2. Classification Model : The model predicts or makes a conclusion to the input data given for preparing and it will anticipate the class or category for the information.

3. Feature : A feature is an individual measurable property of the phenomenon being observed.
4. Binary Classification : It is a type of classification with dual outcomes, as in either true or false.
5. Multi-Class Classification : The type of classification with more than two classes in which each sample is assigned to one and only one label or target.
6. Multi-label Classification : It is a type of classification where each sample is assigned to targets or a set of labels.
7. Train the Classifier : Every classifier in sci-kit learn utilizes the fit(X, y)technique to fit the model for training the train X and train label y.
8. Predict the Target : With the trained model, an unlabeled observation can be labeled with the predict method.
9. Evaluate : This generally includes the evaluation of the model i.e. classification report, accuracy score, etc.

Learner in classification can further be divided into Lazy and Eager learners. Lazy learners store the training data waiting for the testing data to appear and classification is done utilizing the most suitable data from the stored one. When time is considered they are lazy as the name suggest and takes more time compared to Eager learners. K-nearest Neighbors is an example of lazy learner. Eager Learners build a classification model dependent on the given training information before getting data for predictions. It must have the option to focus on a solitary hypothesis that will work for the whole space. Because of this, they take a great deal of time in training and less for prediction. Some examples are Naive Bayes, Decision Tree, Artificial Neural Networks.[17]

## 4.3. Classification algorithms

There is a wide range of classifiers available but it is not possible to conclude the best suitable classifier for the datasets as it depends on the nature and application of data sets.

### 4.3.1. decision tree

Decision Tree builds classification or regression models as a tree structure. It uses an if-then standard set which is fundamentally unrelated and thorough for classification. The guidelines are found out successively utilizing the training data, each in turn. Each time a standard is found out, the tuples secured by the principles are expelled. This procedure is proceeded on the training set until termination condition is met.

The tree is built in a top-down recursive divide-and-conquer way which can be seen in Figure 28[18]. All the properties ought to be categorical. Else, they should be discretized in advance. Properties in the highest point of the tree have more effect towards in the classification and they are distinguished utilizing the data gain idea.

A decision tree can be effectively overfitted producing such a large number of branches and may reflect abnormalities because of noise or anomalies. An over-fitted model has a poor performance on the poor information despite the fact that it gives an amazing performance on training data. This can be maintained by pre-pruning which ends tree development early or post-pruning which expels branches from the completely developed tree.

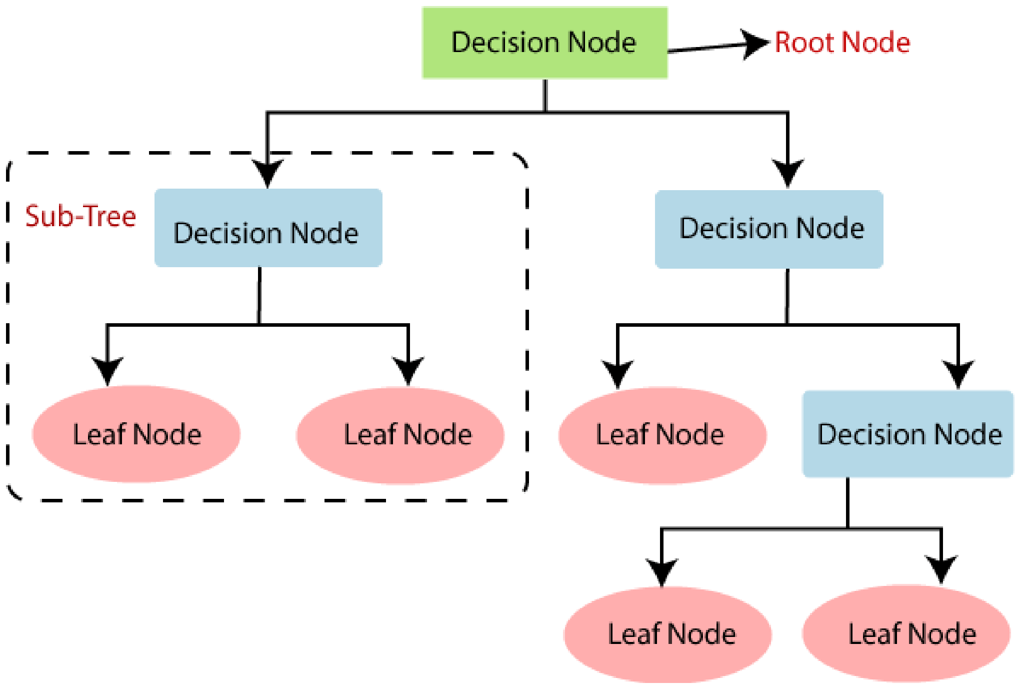


FIGURE 28 - DECISION TREE CLASSIFICATION ALGORITHM

## 4.3.2. K-NEAREST NEIGHBORS

k-Nearest Neighbor is a lazy learning calculator which stores all occurrences related to training data points in n-dimensional space. At the point when an unknown discrete data is given, it breaks down the nearest k number of examples saved (closest neighbors) and restores the most widely recognized class as the prediction and for real-valued data it returns the mean of k closest neighbors which can be descripted in Figure 29.

In the distance-weighted nearest neighbor algorithm, it loads the contribution of each one of the k neighbors as indicated by their separation utilizing the accompanying query giving more noteworthy load to the nearest neighbors.

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

Generally, KNN is robust to noisy data since it is averaging the k-nearest neighbors.[19]

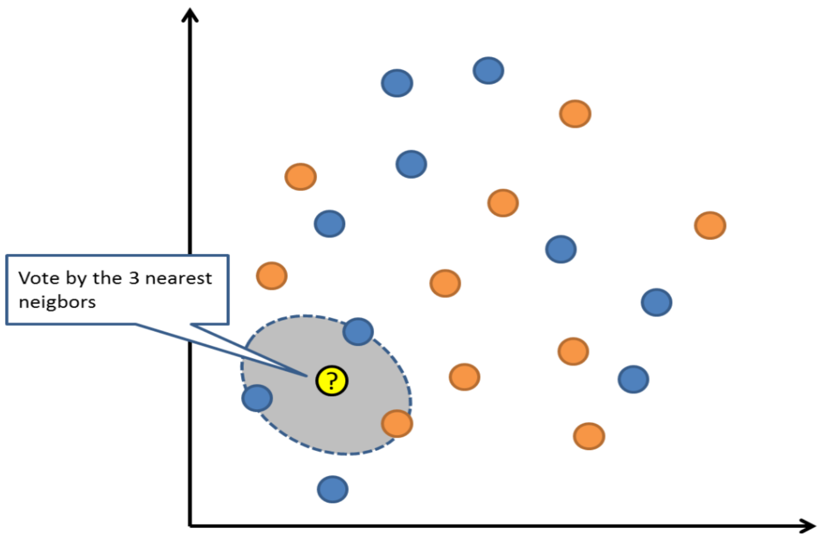


Figure 29 - k-nearest neighbor

### 4.3.3. SUPPORT VECTOR MACHINE

A Support Vector Machine (SVM) is a discriminative classifier, officially characterized by an isolating hyperplane. An SVM model is a portrayal of the models as points in space, mapped with the goal that the instances of the different classes are isolated by a clear gap that is as wide as could be expected under the circumstances. In other words, given labeled training data (supervised learning), the algorithm yields an optimal hyperplane which categorizes new models.

Given a lot of training examples, each set apart as having a place with either of two categories, a SVM training algorithm constructs a model that allocates new guides to one class or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can productively play out a non-linear classification, implicitly mapping their contributions to high-dimensional feature spaces.

What Support vector machines do, is to not just draw a line between two classes here, however considering a region regarding the line of some given width. Figure 30 is the intuition of support vector machines, which optimize a linear discriminant model representing the perpendicular separation between the datasets.[20]

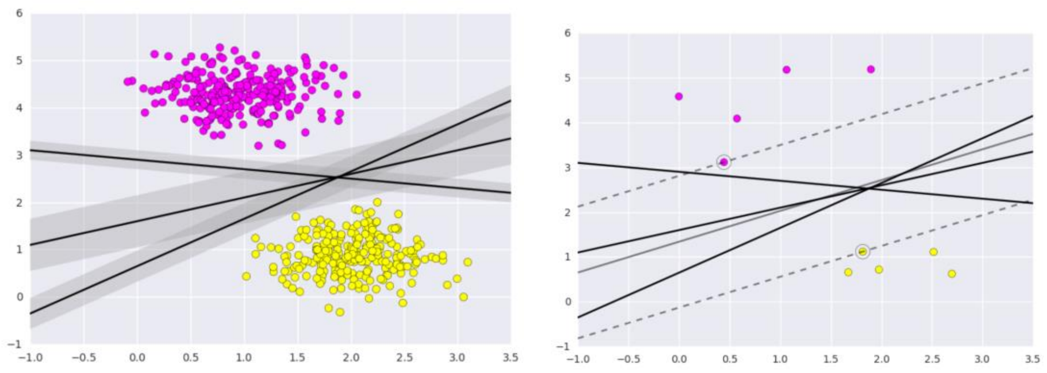


FIGURE 30 - SVC CLASSIFIER

### 4.3.4. LOGISTIC REGRESSION

Logistic regression is a classification algorithm used to allocate observations to a discrete arrangement of classes while for Linear regression the outcome is continuous (Figure 31). Logistic regression changes its yield utilizing the strategic sigmoid function to return a probability value.

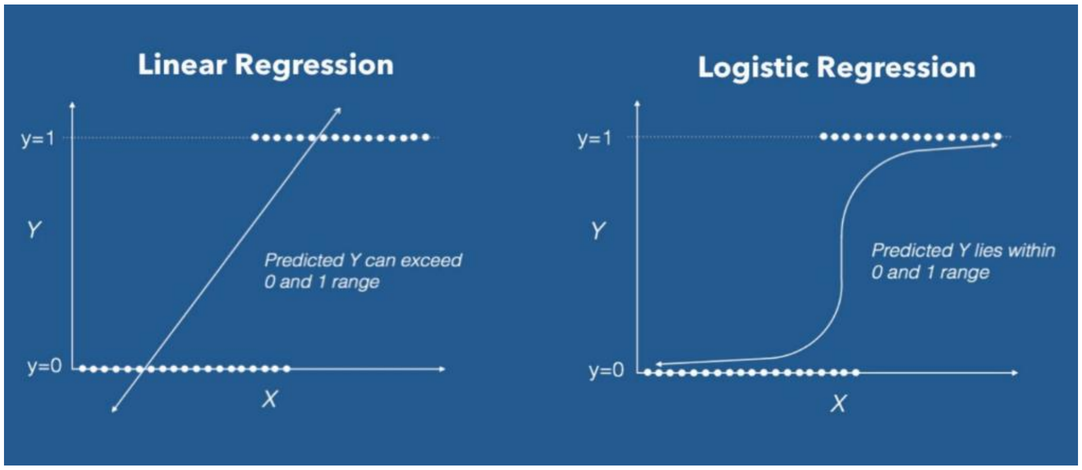

Linear Regression
y=1
Y
Predicted Y can exceed 0 and 1 range
y=0
X
Logistic Regression
y=1
Y
Predicted Y lies within 0 and 1 range
y=0
X

FIGURE 31 - LINEAR REGRESSION VS LOGISTIC REGRESSION

Logistic regression can be further classified into Binary and Multi-linear. It can likewise be called Linear Regression model yet the Logistic Regression utilizes a more unpredictable cost function (Sigmoid function). The speculation of logistic regression tends it to restrict the cost function somewhere in the range of 0 and 1. Subsequently linear functions neglect to represent it as it can have a worth more prominent than 1 or under 0 which is not possible as per the speculation of logistic regression.

$$0 \leq h_\theta(x) \leq 1$$

So as to map predicted values to probabilities, Sigmoid function is utilized. The function maps any genuine values into another value somewhere in the range of 0 and 1 which can be seen in Figure 32. A threshold value is kept between 0 and 1 (generally 0.5) which suggest the result based on the values above or below the threshold.[21]

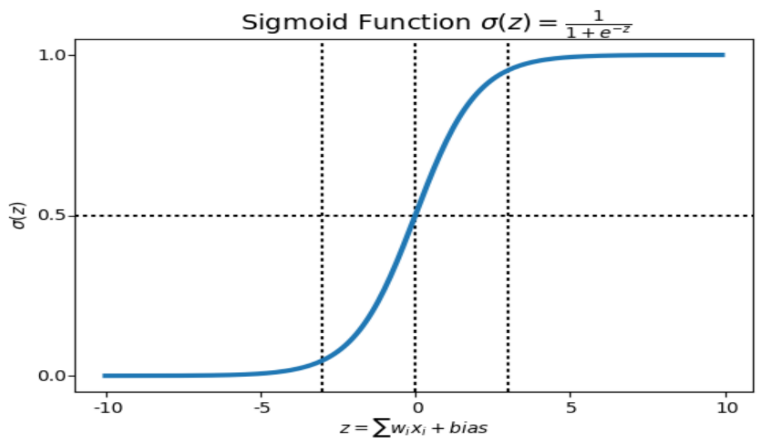


FIGURE 32 - SIGMOID FUNCTION GRAPH

## 4.3.5. MULTILAYER PERCEPTRON

In the Multilayer perceptron, there can be more than one linear layer (combinations of neurons). If an example is taken of normal case of the three-layer arrangement,  first layer will be the input layer, middle layers will be considered as hidden layers and last layer swill be output layer which can be seen in Figure 33. Data is fed to the input layer and yield is taken from the output layer. Hidden layers can be included as much as one need, to make the model progressively complex as indicated by the task.

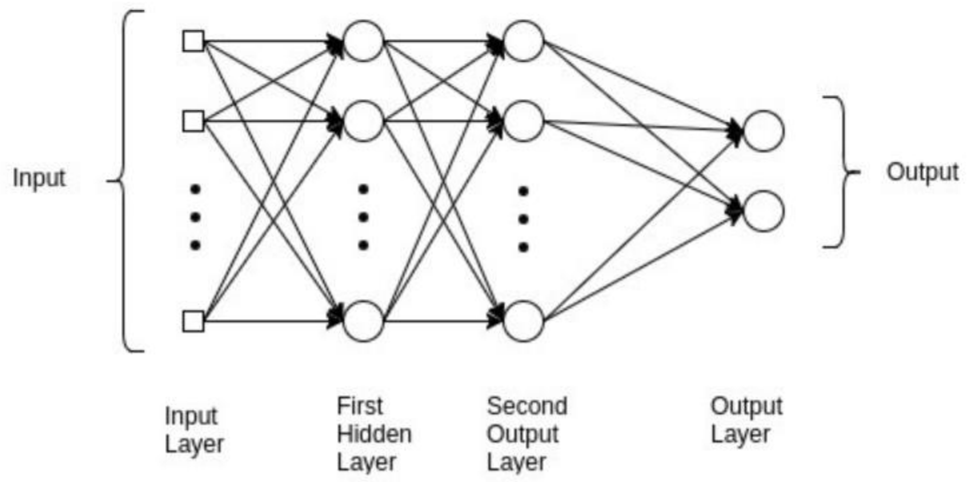


FIGURE 33 - MLP LAYER STRUCTURE

In a supervised classification framework, each input vector is related with a label, or ground truth, characterizing its class or class name is given with the data. The yield of the system gives a class score, or prediction, for each input given. To measure the exhibition of the classifier, the loss function is characterized. The loss will be high if the  predicted class does not relate to the true class, or else it will be low in any other cases. Very often the issue of overfitting and underfitting happens at the hour of training the model. For this situation, the model performs very well on preparing training data yet not on testing data. So as to prepare the system, an optimization procedure is required for which loss function and optimizer is used. This technique will discover the values for the set of weights, W that limits the loss function.

Activation functions or so-called non-linearity describes the working of the process in a non-linear way. This gives the model, the capacity to be increasingly adaptable in describing arbitrary relations. Some of the widely used activation functions are ReLU, TanH, Sigmoid etc. Model training includes mainly 3 steps: Forward pass, Loss calculate and Backward pass.[22]

# 5. Implementation Tools

## 5.1. Python

To overcome the contrast among C and shell programming, Python, a straightforward yet robust tool is accessible in the plenty of programming languages. It is an ideal fit to "Throw-away programming" and fast prototyping. Python's linguistic structure is produced using different languages like Icon, C, Modula-3, ABC etc. The Python interpreter is extended with new function and data type implemented in C.

For highly customizable C applications, for example, editors or window supervisors, python can fill in as an extension language. It is accessible for many operating systems like UNIX (including Linux), MS-DOS, Windows NT, MS-Windows 3.1, and OS/2 the Apple Macintosh working framework. In the Python Library Reference there is a portrayal of the implicit capacities and modules and of the non-essential built in object types.

With python one can isolate the program into modules that might be reused in other Python programs. It has a colossal collection of standard modules that one can use as the premise of the programs — or as guides to begin figuring out how to program in Python. Things like document I/O, attachments, framework calls, and even interfaces to graphical UI toolkits like TK are given by a portion of these modules.

Since Python is an interpreted language, there is no accumulation and connecting is vital and henceforth one can spare considerable time during program improvement. The benefits of such an interactive interpreter are, that one can without much of a stretch test with features of the language, compose throw-away programs and even test capacities during bottom-up program advancement. Python permits programs to be written compactly and readably. Python programs for beginners are much easier and shorter than equivalent C++, C or java programs.[23]

## 5.2. Jupyter Notebook

Jupyter Notebooks have seen energetic selection in the data science network, to a degree where they are progressively replacing Microsoft Word as the default composing condition for research. Inside digital humanities literature, one can discover references to Jupyter notebooks (split off from iPython, or interactive Python, notebooks in 2014) dating to 2015.

Jupyter Notebooks include likewise traction inside computerized humanities as an instructive device. Different Programming Historian tutorials, for example, Text Mining in Python through the HTRC Feature Reader, and Extracting Illustrated Pages from Digital Libraries with Python, just as other educational materials for workshops, make reference to placing code in a Jupyter notebook or utilizing Jupyter notebooks to control learners while permitting them to freely remix and alter code. The notebook format is undeniably appropriate for educating, particularly when students have various degrees of specialized capability and comfort with composing and altering code.

The purpose for Jupyter notebooks is to give a progressively available interface to code utilized in digitally supported research or teaching method. Instruments like Jupyter notebook are less significant to learn or educate about in a vacuum, in light of the fact that Jupyter notebooks themselves do not do anything which may be helpful for further research or teaching method.[24]

## 5.3. Libraries

Some of the libraries which are used in this work are mentioned here. Some of the libraries are Numpy, Pandas, scikit-learn, Pickle etc.

### 5.3.1. scikit-learn

It is a Python library which is related with NumPy and SciPy. It is considered as probably the best library for working with complex information. There is a great deal of changes being made in this library. One change is the cross-validation feature, giving the capacity to utilize more than one measurement. Heaps of training strategies like Logistic regression and nearest neighbors have received some little upgrades. It contains a various number of calculations for actualizing standard machine learning and data mining tasks like diminishing dimensionality, classification, regression, clustering, and model choice.

Features:

1. Cross-validation : There are different strategies to check the precision of managed models on unseen data.
2. Unsupervised learning algorithms : Again, there is a huge spread of calculations in the contribution–beginning from clustering, factor examination, head segment analysis to unsupervised neural systems.
3. Feature extraction : Useful in classification for feature extraction from text and images.

### 5.3.2. NUMPY

Numpy is considered as one of the most mainstream ML library in Python. TensorFlow and different libraries utilizes Numpy inside for playing out various procedure on Tensors. Array interface is the best and the most significant element of Numpy. This interface can be used for expressing pictures, sound waves, and other binary raw streams as a variety of genuine numbers in N-dimensional. For executing this library for ML, knowledge about Numpy is significant for full stack developers.

Features:

1. Very interactive and easy to use.

2. Makes complex numerical executions easy.
3. Makes coding  simple and getting a handle on the concepts is simple.
4. Broadly utilized, henceforth a great deal of open source commitment.

### 5.3.3. PANDAS

Pandas is an ML library in Python that gives data structures of elevated level and a wide assortment of tools for analysis. One of the incredible features of this library is the capacity to decipher complex activities with data utilizing a couple of orders. Pandas have such a large number of inbuilt strategies for gathering, joining data, and filtering, as well as time-series functionality. Pandas ensure that the whole procedure of manipulating data will be simpler.

Backing for activities, for example, Re-ordering, Iteration, Sorting, Aggregations, Concatenations and Visualizations are among the component features of Pandas. Presently, there are less releases of pandas library which incorporates hundreds of new features, bug fixes, upgrades, and changes in API. The upgrades in pandas respects its capacity to gathering and sorting data, selects most appropriate yield for the apply method, and offers help for performing custom sort tasks.

Data Analysis among everything else takes the highlight with regards to use of Pandas. Be that as it may, Pandas when utilized with different libraries and devices, guarantees high usefulness and great measure of adaptability.[25]

### 5.3.4. MATPLOTLIB

Matplotlib is a Python library that utilizes Python Script to compose 2-dimensional charts and plots. Regularly numerical or logical applications require more than single signals in a description. This library encourages to construct different plots one after another which be shown in Figure 34. One can, nonetheless, use Matplotlib to control various characteristics of figures too.

Features:

1. Matplotlib can make such quality figures that are great for distribution. Figures made with Matplotlib are accessible in printed version groups across various interactive stages.
2. One can utilize MatPlotlib with various toolkits, for example, Python Scripts, IPython Shells, Jupyter Notebook, and numerous other four graphical UIs.

3. Various third-parties libraries can be coordinated with Matplotlib applications. For example, seaborn, ggplot, other projections and mapping toolkits such as base map.
4. A functioning network of developers is devoted to assist with any of the requests with Matplotlib. Their commitment to Matplotlib is profoundly appreciable.
5. Beneficial thing is that one can track any bugs, new fixes, and feature requests on the issue tracker page from Github. It is an official page for featuring various issues identified with Matplotlib.[26]

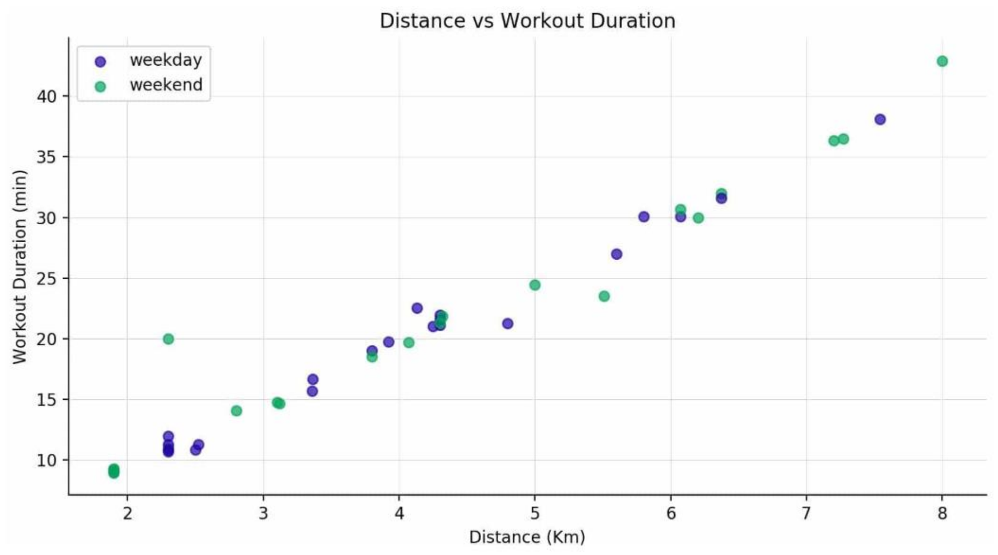


Figure 34 - graph using matplotlib

## 5.3.5. openCV

OpenCV, also known as Open Source Computer Vision, is a python package for image processing. It screens overall functions that are centered around instant PC vision. In spite of the fact that OpenCV has no appropriate documentation, as per numerous developers, it is perhaps the hardest library to learn. However, it gives numerous inbuilt capacities through which one can learn Computer vision without any problem.

Features:

1. OpenCV is a perfect image processing bundle that permits one to both study and compose images simultaneously.
2. Computer Vision permits one to remake, interfere, and fathom a 3D domain from its particular 2D condition.
3. This bundle permits one to analyze exceptional items in any recordings or pictures. Items, for example, faces, eyes, trees, and so forth.
4. Ones can likewise spare and catch any snapshot of a video and furthermore break down its various properties like motion, foundation, and so forth.
5. OpenCV is great with many working frameworks, for example, Windows, OS-X, Open BSD, and numerous others.

### 5.3.6. PICKLE

Pickle is utilized for serializing and de-serializing Python object structures, likewise, called marshaling or straightening. Serialization alludes to the way towards changing over an item in memory to a byte stream that can be put away on disk or sent over a system. Later on, this character stream would then be able to be recovered and deserialized back to a Python object. The former is the transformation of an item from one portrayal (data in RAM) to another (text on disk), while the latter is the way towards encoding information with less bits, so as to spare disk space.

Pickling is helpful for applications where there is a need of some level of persistency in the information. The program's state information can be saved to disk, so that one can keep working at it later on. It can likewise be utilized to send information over a Transmission Control Protocol (TCP) or socket connection, or to store python objects in a database. Pickle is helpful for when one is working with ML algorithms, where there is a need to save them to have the option to make new predictions sometime in the future, without revamping everything or training the model once again.[27]

# 6. Implementation

## 6.1. hyper-parameters tuning for different classifiers

While making an ML model, plan decisions will be given with regards to how to characterize the model engineering. Generally, one does not quickly have the optimal idea what the ideal model engineering ought to be for a given model, and in this manner, one prefers to have the option to explore a range of outcomes. In true ML style, machine is requested to perform the investigation and select the ideal model engineering consequently. Parameters which characterize the model architecture are kown as hyperparameters and hence this procedure of searching for the perfect model design is referred to as hyperparameter tuning.

***Grid search*** is seemingly the most essential hyperparameter tuning technique. With this procedure, one essentially constructs a model for every possible combination of the values of the hyperparameter values given, assessing each model, and choosing the design which delivers the best outcomes. Here I will be clarifying about the grid search parameters used for different classifiers.[28]

**Parameter selection**

Basically, Grid search takes a range of parameters from which it suggests the best suitable parameter. In order to decide which range to be given, a detailed search is required. From [29] range of C for SVC can be determined. Similarly, from [30], [31], [33] and [34], range suitable for classifier can be averaged and hence C = [1, 10, 100, 1000] was considered. If we the above-mentioned references, then they have considered values of C up to 0.001. Previously model was trained including these values and gave the best parameter in positive values. Hence to increase the scope, positive values has been considered. Similarly, references have been considered in the same procedure for all the classifiers in order to get the best range of parameters for the hyper tuning.

**MLP Classifier parameters:**

*hidden_layer_sizes* allow the user to set the number of nodes and number of layers in the neural network classifier. Every component in the tuple represents the quantity of nodes at the ith position where i is the index of the tuple. Accordingly, the length of tuple means the absolute number of hidden layers in the network..

*activation* function includes ‘identity’, ‘logistic’, ‘tanh’, ‘relu’. The activation function is a numerical "gate" in the middle of the input feeding the current neuron and its output setting off to the following layer. It tends to be as basic as a step function that turns the neuron output on and off, depending upon a standard or threshold.

*learning _rate* is a tuning parameter in an optimization algorithm that decides the progression size at every iteration while advancing toward a least of loss function. ‘constant’ keeps the learning rate constant at a given value, ‘invscaling’ gradually decreases the learning rate while ‘adaptive’ keeps the learning rate constant as long as training loss keeps decreasing.[35]

*alpha* is a parameter for regularization term, otherwise known as penalty term, that combats overfitting by compelling the size of the weights. Increasing alpha may fix high fluctuation (an indication of overfitting) by encouraging smaller weights, bringing about a decision boundary plot that shows up with lesser curvatures.[36]

Learning rate : constant, invscaling, adaptive

Alpha values : [0.1, 0.01, 0.001, 0.0001, 0.00001]

*Best parameters for Precision* : {learning_rate = 0.1, alpha = constant}

*Best parameters for Recall* : {learning_rate = 0.1, alpha = invscaling}

**Decision Tree Classifier parameters:**

*Criterion* is about understanding what a decent split point for root nodes on classification trees is. Decision trees split on the feature and corresponding split point that outcomes in the biggest data gain (IG) for a given criterion. “Gini” and “entropy” are the supported criteria.

*max_depth* is a measure of how many splits a tree can make before coming to a prediction. This procedure could be proceeded further with more splitting until the tree is as pure as possible under the circumstances. Default is None which will expand until all the leaves are pure.

*min_samples_split* is the least number of samples required to split an internal node.[37][38]

min samples split : range(2, 22, 2)

max depth : [4, 8, 12]

*Best parameters for Precision & Recall* : {max_depth = 12, min_samples_split = 4}

**Logistic Regression parameters:**

*Penalty* is used to specify the method of penalization of the coefficients of noncontributing variables. L1 (Lasso) is related to feature selection as it neglects or shrinks the coefficient of less important features to zero. L2 (Ridge) considers all the features, only few are shrunk. Less computationally serious than Lasso.

*Solvers* refers to the optimization method used to find the optimum of the objective function. It includes “liblinear”, “lbfgs”, “newton-cg”, “saga” and “sag”. Both penalties restrict the choice of solver. Liblinear uses L1 penalty while lbfgs, newton-cg and sag support only L2. Saga is a variant of sag, which support L1, L2 and also “Elastic-Net”.

*C* is the trade-off parameter of logistic regression that decides the quality of the regularization, and higher estimations of C relate to less regularization (where one can indicate the regularization function).

C : [0.01, 0.1, 1, 10, 100]

Penalty : [l1, l2]

*Best parameters for Precision & Recall* : {C = 0.1, penalty = l1}

**K-nearest Neighbors Classifier parameters:**

*n_neighbors* determines the notation of neighbors utilized while figuring the algorithm. The number of neighbors is the core deciding factor.

*P* is a power parameter in which if the value used is 1, it calculates Manhattan distance and Euclidean distance when considered 2. Distance metric uses distance function which gives a relationship metric between each component in the datasets.

No. of neighbors : range(1, 31)

*Best parameters for Precision & Recall* : {n_neighbors = 2}

**Support Vector Classifier parameters:**

*C* is the inverse of the regularization term which follows the same principal as suggested in Logistic Regression. Overfit can be caused to high value of C.

*Kernel's* function is to accept data as input and change it into the necessary structure. Diverse SVM algorithms utilize various kinds of kernel functions. It must be one of the 'linear', 'rbf', 'poly', 'sigmoid', 'precomputed' or any callable.

*gamma* parameter characterizes how far the impact of a single training example ranges, with low values signifying 'far' and high values, 'close'. The gamma parameters can be viewed as the reverse of the radius of influence samples chose by the model as support vectors. It represents the coefficients of the above given kernels.[39]

C : [1, 10, 100, 1000]

Gamma : [0.001, 0.0001]

*Best parameters for Precision & Recall* : {C = 10, gamma = 0.0001}

## 6.2. Scores on different classifiers

Precision and recall are two critical model assessment metrics. While precision refers to the level of the outcomes which are significant, recall refers to the level of absolute applicable outcomes effectively grouped by the calculation which is portrayed in Figure 35. Unfortunately, it is impossible to expand both these measurements simultaneously, as one comes at the expense of another.

For simplicity, there is another metric accessible, called F-1 score, which is a consonant mean of precision and recall. For issues where both precision and recall are significant, one can choose a model which amplifies this F-1 score. For different issues, a trade-off is required, and a choice must be made whether to amplify precision or recall. Here I will be showing the results of precision, recall and F-1 score for different classifiers.[40]

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

Figure 35 - Precision and recall

A Classification report is utilized to measure the nature of predictions from a classification algorithm. What number of predictions are True and what number are False can be made out through the report. More explicitly, True Positives, False Positives, True negatives and False Negatives are utilized to foresee the metrics of a classification report as demonstrated below in the Tables 2, 3 and 4.[41]

**Decision Tree Classifier scores:**

Classification reports with the Precision, Recall and F-1 score has been mentioned in the below given Tables. Classification reports are generated on Resized data (300*300), Adaptive Equalized data and HOG data in Tables 2, 3 and 4, respectively.

Adaptive Histogram Equalization is a computer image processing technique which improves contrasts in images. It differs from conventional histogram equalization in the regard that the adaptive strategy computes a few histograms, each comparing to a particular segment of the picture, and uses them to redistribute the lightness estimations of the picture. It is therefore suitable for improving the neighborhood contrast and upgrading the definitions of edges in each area of a picture. Histogram of Oriented Gradients is used in image processing or computer vision mainly for object detection. This technique counts occurrences of gradient orientation in localized portions of an image. For resized data, images have been resized to (128*128) pixels.

| | Precision | Recall | F-1 score |
|---|---|---|---|
| Defective | 0.89 | 0.87 | 0.88 |
| ok | 0.87 | 0.90 | 0.89 |
| | | | |
| Accuracy | | | 0.88 |
| Macro avg. | 0.88 | 0.88 | 0.88 |
| Weighted avg. | 0.88 | 0.88 | 0.88 |

TABLE 2 - SCORES ON RESIZED IMAGES FOR DT CLASSIFIER

| | Precision | Recall | F-1 score |
|---|---|---|---|
| Defective | 0.87 | 0.76 | 0.81 |
| ok | 0.79 | 0.89 | 0.84 |
| | | | |
| Accuracy | | | 0.82 |
| Macro avg. | 0.83 | 0.82 | 0.82 |
| Weighted avg. | 0.83 | 0.82 | 0.82 |

TABLE 3 - SCORES ON ADAPTIVE EQUALIZATION FOR DT CLASSIFIER

| | Precision | Recall | F-1 score |
|---|---|---|---|
| Defective | 0.81 | 0.75 | 0.78 |
| ok | 0.77 | 0.82 | 0.80 |
| | | | |
| Accuracy | | | 0.79 |
| Macro avg. | 0.79 | 0.79 | 0.79 |
| Weighted avg. | 0.79 | 0.79 | 0.79 |

TABLE 4 - SCORES ON HOG FOR DT CLASSIFIER

Similarly, model was trained on three different training datasets mentioned above. Scores were generated on each model trained and hence total 18 models were trained which is explained in Table 5. Instead of showing all the tables of different classifiers for different datasets it can be summarized into one single table which is shown in Table 6 and 7.

| *CLASSIFIERS* | *NO. OF MODELS* |
|---|---|
| *DECISION TREE* | *3* |
| *K-NEAREST NEIGHBORS* | *3* |
| *LOGISTIC REGRESSION* | *3* |
| *MULTILAYER PERCEPTRON* | *2x3 = 6** |
| *SUPPORT VECTOR CLASSIFIER* | *3* |
| *TOTAL* | *18* |

TABLE 5 - MODELS GENERATED

*All the classifiers have Same Precision and Recall Scores Except MLP.

Training datasets:

A = Resized Images (128*128)

B = Resized Images with Adaptive Equalization

C = Resized Images with Histogram of Histogram of Oriented Gradients(HOG) descriptors.

**Decision Tree Classifier**

| TRAINING DATA | ACCURACY | MACRO AVERAGE | | | WEIGHTED AVERAGE | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| A | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 |
| B | 0.82 | 0.83 | 0.82 | 0.82 | 0.83 | 0.82 | 0.82 |
| C | 0.79 | 0.79 | 0.79 | 0.79 | 0.79 | 0.79 | 0.79 |
| | | | | | | | |

**KNeighbors Classifier**

| TRAINING DATA | ACCURACY | MACRO AVERAGE | | | WEIGHTED AVERAGE | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| A | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| B | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| C | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| | | | | | | | |

**SVC**

| TRAINING DATA | ACCURACY | MACRO AVERAGE | | | WEIGHTED AVERAGE | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| A | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| B | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| C | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| | | | | | | | |

Table 6 - classification report for DT, KNN, SVC

| Logistic Regression | | | | | | | |
|---|---|---|---|---|---|---|---|
| TRAINING DATA | ACCURACY | MACRO AVERAGE | | | WEIGHTED AVERAGE | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| A | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |
| B | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| C | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| | | | | | | | |
| MultiLayer Perceptron(Precision BEST) | | | | | | | |
| TRAINING DATA | ACCURACY | MACRO AVERAGE | | | WEIGHTED AVERAGE | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| A | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| B | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| C | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| | | | | | | | |
| MultiLayer Perceptron(RECALL BEST) | | | | | | | |
| TRAINING DATA | ACCURACY | MACRO AVERAGE | | | WEIGHTED AVERAGE | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| A | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| B | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| C | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| | | | | | | | |

Table 7 - classification report for LR and MLP

From Table 6 and 7 a broad idea of the model can be framed from its values generated. Macro avg. says the function to compute f1 for each label and returns the average without considering the proportion for each label in the dataset while Weighted avg. says the function to compute f1 for each label, and returns the average considering the proportion for each label in the dataset. Best training dataset in every classifier is highlighted.

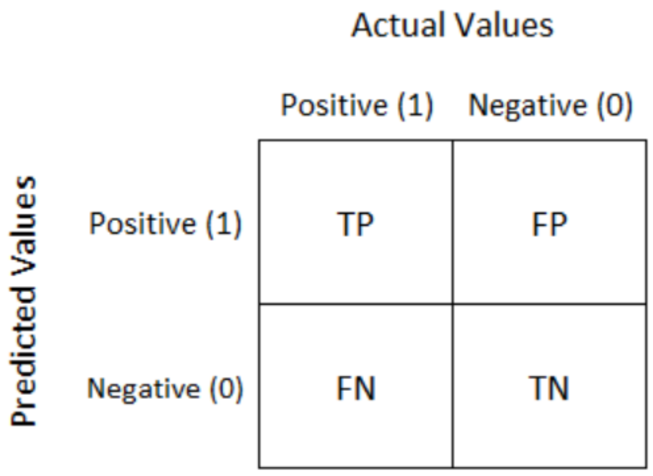
Comparing all the best results, SVC and MLP got the highest accuracies about 100%. Out of the three training datasets in SVC and MLP, A (Resized images) and B(adaptive Equalized images) performed the best. Logistic Regression and K-nearest Neighbors showed the accuracy on average 95 % and 97%, respectively when considering all three training datasets. On the other hand, Decision Tree performed the worst of all classifiers, producing on average 83%.

## 6.3. Confusion matrix

Confusion Matrix as shown in Figure 36, is a performance estimation for ML classification issue where yield can be at least two classes. It is a table with 4 unique combinations of anticipated and genuine values. It is very helpful for estimating Recall, Precision, Specificity, Accuracy and AUC-ROC Curve.

Taking an example of a pregnant woman and a man which can be seen in Figure 37.

Here predicted values are positive and negative actual values as true and false.



| | | Actual Values | |
|---|---|---|---|
| | | Positive (1) | Negative (0) |
| Predicted Values | Positive (1) | TP | FP |
| | Negative (0) | FN | TN |

Figure 36 - confusion matrix

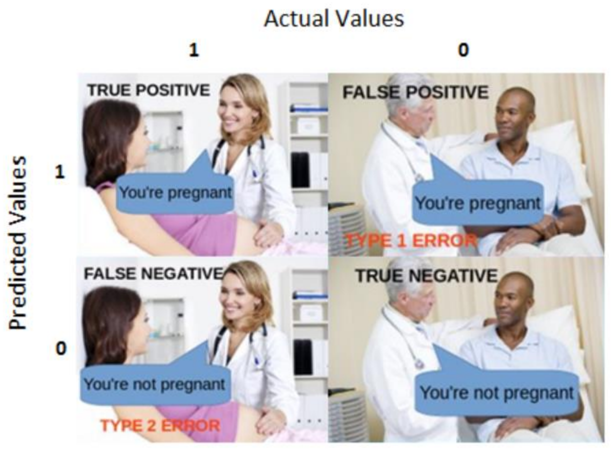


Figure 37 - example of confusion matrix

TP means True Positive which means one predicted positive and its true. Prediction is woman is pregnant and she is.

TN means True Negative which means one predicted negative and its true. Prediction is man is not pregnant and he is not.

FP means False Positive(Type 1 Error) which means one predicted positive and its false. Prediction is man is pregnant and he is not.

FN means False Negative(Type 2 Error) which means one predicted that women is not pregnant but she is.[42]

Some of the examples of confusion matrix of Resized, AE and HOG Data and are as follows in Figures 38, 39 and 40, respectively.

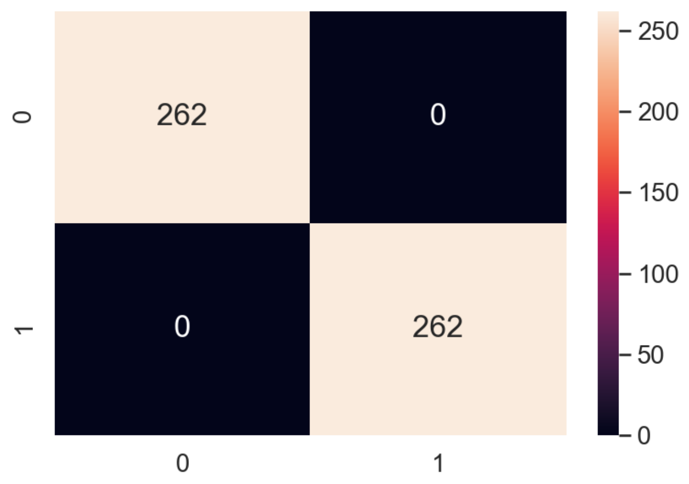


Figure 38 - confusion matrix of resized data for svc

Considering Figure 38, I got 262, 0, 262 and 0 as TP, FP, FN, TN values, respectively. The figure demonstrates the confusion matrix for SVC classifier which used the resized data as training data.

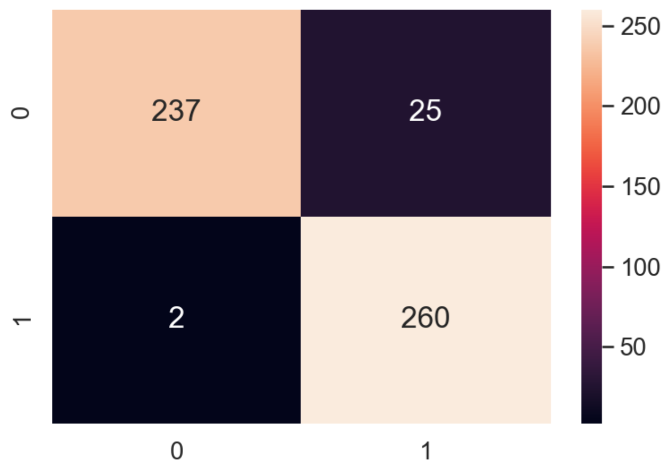


FIGURE 39 - CONFUSION MATRIX OF HOG FOR KNN

From Table 6, the accuracy of training data A i.e. resized data, is 1 which suggest that the model was perfectly trained predicting all the images correct.

Similarly, in Figure 39, 25 images were predicted as FP and 2 as FN. In other words, 25 images were predicted OK though being defective and 2 were predicted defective though being OK. Hence the accuracy of the model is less which can be made out from Table 6 in the C of KNN (95%).

Figure 40 demonstrates the confusion matrix of Decision Tree for Adaptive Equalized data. From the figure, a high number of FP and FN can be noted which suggest its low accuracy and can be made out from Table 6 in the B of DT(82%).

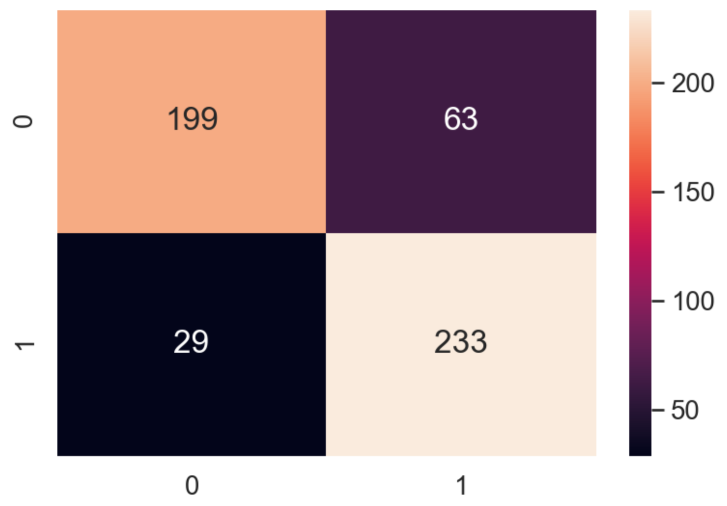


Figure 40 - confusion matrix of adaptive equalization for DT

# 7. Conclusion and future improvements

This work presents the solution to quality assessment of casting products which are discussed deeply in previous chapters. Defects are detected manually but it can incur a waste of time and also human error which may be reason behind the rejection of whole order. In order to avoid the wastage of time and rejection Machine learning algorithms were used to forecast the state of the final product. The goal was to find the best parameters for the classifiers which were obtained using grid search. The best parameters found were used to train the models and classification report were generated to match the scores obtained. In table 7 the best accuracy scores on different training data sets are highlighted and from it a conclusion can be derived which suggest SVC and MLP can be the best suited classifiers for our data.

Convolutional Neural Network (CNN) these days is succeeded by MLP. MLP is presently esteemed inadequate for current propelled computer vision tasks and has the attribute of completely connected layers, where each perceptron is associated with  other perceptron. Disadvantage is that the quantity of complete parameters can develop to exceptionally high. CNN is the winner of multiple ImageNet competitions. In CNN, each filter is panned around the whole picture as per certain size and stride permits the filter to discover and coordinate patterns regardless of where the example is situated in a given picture. Weights considered are smaller and shared which means easier to train and less wasteful. The layers are sparsely connected or partially associated as opposed to completely connected.

# 8. References

[1] "Industry 4.0 - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Industry_4.0. [Accessed: 01-Jul-2020].

[2] A. Joshi and L. M. Jugulkar, *INVESTIGATION AND ANALYSIS OF METAL CASTING DEFECTS AND DEFECT REDUCTION BY USING QUALITY CONTROL TOOLS*. 2014.

[3] S. Chaudhari and H. Thakkar, "Review on Analysis of Foundry Defects for Quality Improvement of Sand Casting," 2014.

[4] B. Adaway, "Industrial applications of image processing.," *Comput. Graph. 83, (Online Publ. Pinner)*, pp. 555–568, 1983.

[5] S. Dey, "Hands-On Image Processing with Python: Expert techniques for advanced image ... - Sandipan Dey - Google Books." .

[6] N. Dalal, B. T. Histograms, and B. Triggs, "Histograms of Oriented Gradients for Human Detection," pp. 886–893, 2005.

[7] "Histogram Equalization — skimage v0.17.2 docs." [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_equalize.html. [Accessed: 22-Jun-2020].

[8] "Supervised and Unsupervised Machine Learning Algorithms." [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. [Accessed: 20-Jun-2020].

[9] M. R. M. Talabis, R. McPherson, I. Miyamoto, J. L. Martin, and D. Kaye, "Analytics Defined," in *Information Security Analytics*, Elsevier, 2015, pp. 1–12.

[10] I. Nitze, A. Wegener, U. Schulthess, I. Nitze, U. Schulthess, and H. Asche, "COMPARISON OF MACHINE LEARNING ALGORITHMS

RANDOM FOREST, ARTIFICIAL NEURAL NETWORK AND SUPPORT VECTOR MACHINE TO MAXIMUM LIKELIHOOD FOR SUPERVISED CROP TYPE CLASSIFICATION,” 2012.

[11] Z. Kozareva and A. Montoyo, “LNAI 4139 - Paraphrase Identification on the Basis of Supervised Machine Learning Techniques.”

[12] A. C. Lorena *et al.*, “Comparing machine learning classifiers in potential distribution modelling,” *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5268–5275, May 2011.

[13] “Casting (metalworking) - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Casting_(metalworking). [Accessed: 22-Jun-2020].

[14] “How to Handle Imbalanced Classes in Machine Learning.” [Online]. Available: https://elitedatascience.com/imbalanced-classes. [Accessed: 22-Jun-2020].

[15] “casting product image data for quality inspection | Kaggle.” [Online]. Available: https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product. [Accessed: 22-Jun-2020].

[16] “3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.23.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html. [Accessed: 24-Jun-2020].

[17] “Classification In Machine Learning | Classification Algorithms | Edureka.” [Online]. Available: https://www.edureka.co/blog/classification-in-machine-learning/. [Accessed: 24-Jun-2020].

[18] “Machine Learning Decision Tree Classification Algorithm - Javatpoint.” [Online]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. [Accessed: 25-Jun-2020].

[19] “Machine Learning Classifiers - Towards Data Science.” [Online].

Available: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623. [Accessed: 25-Jun-2020].

[20] "Classifying data using Support Vector Machines(SVMs) in Python - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/. [Accessed: 25-Jun-2020].

[21] "Introduction to Logistic Regression - Towards Data Science." [Online]. Available: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148. [Accessed: 25-Jun-2020].

[22] "Understanding of Multilayer perceptron (MLP) - Nitin Kumar Kain - Medium." [Online]. Available: https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f. [Accessed: 25-Jun-2020].

[23] "3.5.9 Documentation." [Online]. Available: https://docs.python.org/3.5/. [Accessed: 26-Jun-2020].

[24] Q. Dombrowski, T. Gniady, and D. Kloster, "Introduction to Jupyter Notebooks," *Program. Hist.*, no. 8, Dec. 2019.

[25] "Top 10 Python Libraries You Must Know In 2020 | Edureka." [Online]. Available: https://www.edureka.co/blog/python-libraries/. [Accessed: 26-Jun-2020].

[26] "The 30 Best Python Libraries and Packages for Beginners." [Online]. Available: https://www.ubuntupit.com/best-python-libraries-and-packages-for-beginners/. [Accessed: 26-Jun-2020].

[27] "Python Pickle Tutorial - DataCamp." [Online]. Available: https://www.datacamp.com/community/tutorials/pickle-python-tutorial. [Accessed: 27-Jun-2020].

[28] "Hyperparameter tuning for machine learning models." [Online]. Available: https://www.jeremyjordan.me/hyperparameter-tuning/. [Accessed: 27-Jun-2020].

[29] "Tune Hyperparameters for Classification Machine Learning

Algorithms." [Online]. Available: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/. [Accessed: 11-Jul-2020].

[30] "Seleting hyper-parameter C and gamma of a RBF-Kernel SVM — scikit-learn 0.11-git documentation." [Online]. Available: https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/auto_examples/svm/plot_svm_parameters_selection.html. [Accessed: 11-Jul-2020].

[31] "In Depth: Parameter tuning for SVC | by Mohtadi Ben Fraj | All things AI | Medium." [Online]. Available: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769. [Accessed: 11-Jul-2020].

[32] "3.2. Tuning the hyper-parameters of an estimator — scikit-learn 0.23.1 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html. [Accessed: 11-Jul-2020].

[33] "SVM Hyperparameter Tuning using GridSearchCV | ML - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/. [Accessed: 11-Jul-2020].

[34] "HyperParameter tuning an SVM — a Demonstration using HyperParameter tuning | by Rohit Madan | Analytics Vidhya | Medium." [Online]. Available: https://medium.com/analytics-vidhya/hyperparameter-tuning-an-svm-a-demonstration-using-hyperparameter-tuning-cross-validation-on-96b05db54e5b. [Accessed: 11-Jul-2020].

[35] "sklearn.neural_network.MLPClassifier — scikit-learn 0.23.1 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. [Accessed: 09-Jul-2020].

[36] "Varying regularization in Multi-layer Perceptron — scikit-learn 0.23.1 documentation." [Online]. Available: https://scikit-

learn.org/stable/auto_examples/neural_networks/plot_mlp_alpha.html. [Accessed: 10-Jul-2020].

[37] "Understanding Decision Trees for Classification (Python) | by Michael Galarnyk | Towards Data Science." [Online]. Available: https://towardsdatascience.com/understanding-decision-trees-for-classification-python-9663d683c952. [Accessed: 10-Jul-2020].

[38] "sklearn.tree.DecisionTreeClassifier — scikit-learn 0.23.1 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed: 09-Jul-2020].

[39] "Hyper-Parameter Tuning and Model Selection, Like a Movie Star | by Caleb Neale | Towards Data Science." [Online]. Available: https://towardsdatascience.com/hyper-parameter-tuning-and-model-selection-like-a-movie-star-a884b8ee8d68. [Accessed: 09-Jul-2020].

[40] "Precision vs Recall - Towards Data Science." [Online]. Available: https://towardsdatascience.com/precision-vs-recall-386cf9f89488. [Accessed: 27-Jun-2020].

[41] "Understanding the Classification report through sklearn - Muthukrishnan." [Online]. Available: https://muthu.co/understanding-the-classification-report-in-sklearn/. [Accessed: 27-Jun-2020].

[42] "Understanding Confusion Matrix - Towards Data Science." [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Accessed: 27-Jun-2020].

# 9. Appendix

## Appendix 1 - image processing with resizing.

```
DATADIR = './train/def'
DATADIR1 = './test/ok'

IMG_SIZE = 128

path = os.path.join(DATADIR)
for img in os.listdir(path):
    img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_
GRAYSCALE)
    new_array = cv2.resize(img_array,(IMG_SIZE, IMG_SIZE))
    plt.imshow(new_array, cmap='gray')
    plt.title('Defective Resized Image')
    plt.show()
    break

path1 = os.path.join(DATADIR1)
for img in os.listdir(path1):
    img_array1 = cv2.imread(os.path.join(path1,img), cv2.IMREA
D_GRAYSCALE)
    new_array1 = cv2.resize(img_array1,(IMG_SIZE, IMG_SIZE))
    plt.imshow(new_array1, cmap='gray')
    plt.title('Ok Resized Image')
    plt.show()
    break
```

# Appendix 2 - image processing with adaptive equalization.

```
path = os.path.join(DATADIR)
for img in os.listdir(path):
    img_array = cv2.imread(os.path.join(path,img))
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    fd, hog_image = hog(new_array, orientations=8, pixels_per_
cell=(8, 8),
                        cells_per_block=(1, 1), visualize=True
)
    fig, (axes1, axes2) = pylab.subplots(1, 2, figsize=(10, 5)
, sharex=True, sharey=True)
    axes1.axis('off'), axes1.imshow(new_array, cmap=pylab.cm.g
ray),
    axes1.set_title('Defective Resized image')
    hog_image_rescaled = exposure.rescale_intensity(hog_image,
in_range=(0, 10))
    axes2.axis('off'), axes2.imshow(hog_image_rescaled, cmap=p
ylab.cm.gray),
    axes2.set_title('Histogram of Oriented Gradients')
    pylab.show()
    break

path = os.path.join(DATADIR1)
for img in os.listdir(path):
    img_array = cv2.imread(os.path.join(path,img))
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    fd, hog_image = hog(new_array, orientations=8, pixels_per_
cell=(8, 8),
                        cells_per_block=(1, 1), visualize=True
)
    fig, (axes1, axes2) = pylab.subplots(1, 2, figsize=(10, 5)
, sharex=True, sharey=True)
    axes1.axis('off'), axes1.imshow(new_array, cmap=pylab.cm.g
ray),
    axes1.set_title('Ok Resized image')
    hog_image_rescaled = exposure.rescale_intensity(hog_image,
in_range=(0, 10))
    axes2.axis('off'), axes2.imshow(hog_image_rescaled, cmap=p
ylab.cm.gray),
    axes2.set_title('Histogram of Oriented Gradients')
    pylab.show()
    break
```

# Appendix 3 - image processing with histogram of oriented gradients

```
path = os.path.join(DATADIR)
for image in os.listdir(path):
    img_array = cv2.imread(os.path.join(path,image))
    img = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))

    img_adapteq = exposure.equalize_adapthist(img, clip_limit=
0.03)

    fig = plt.figure(figsize=(15, 8))
    axes = np.zeros((2, 4), dtype=np.object)
    axes[0, 0] = fig.add_subplot(2, 4, 1)
    for i in range(1, 2):
        axes[0, i] = fig.add_subplot(2, 4, 1+i, sharex=axes[0,
0], sharey=axes[0,0])
    for i in range(0, 2):
        axes[1, i] = fig.add_subplot(2, 4, 5+i)

    ax_img, ax_hist, ax_cdf = plot_img_and_hist(img, axes[:, 0
])
    ax_img.set_title('Defective Resized Image')

    y_min, y_max = ax_hist.get_ylim()
    ax_hist.set_ylabel('Number of pixels')
    ax_hist.set_yticks(np.linspace(0, y_max, 5))

    ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_adapteq, a
xes[:, 1])
    ax_img.set_title('Adaptive equalization')

    ax_cdf.set_ylabel('Fraction of total intensity')
    ax_cdf.set_yticks(np.linspace(0, 1, 5))

    fig.tight_layout()
    plt.show()
    break


path = os.path.join(DATADIR1)
for image in os.listdir(path):
    img_array = cv2.imread(os.path.join(path,image))
    img = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    img_adapteq = exposure.equalize_adapthist(img, clip_limit=
0.03)

    fig = plt.figure(figsize=(15, 8))
    axes = np.zeros((2, 4), dtype=np.object)
    axes[0, 0] = fig.add_subplot(2, 4, 1)
    for i in range(1, 2):
        axes[0, i] = fig.add_subplot(2, 4, 1+i, sharex=axes[0,
0], sharey=axes[0,0])
    for i in range(0, 2):
        axes[1, i] = fig.add_subplot(2, 4, 5+i)

    ax_img, ax_hist, ax_cdf = plot_img_and_hist(img, axes[:, 0
])
    ax_img.set_title('Ok Resized Image')

    y_min, y_max = ax_hist.get_ylim()
    ax_hist.set_ylabel('Number of pixels')
    ax_hist.set_yticks(np.linspace(0, y_max, 5))

    ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_adapteq, a
xes[:, 1])
    ax_img.set_title('Adaptive equalization')

    ax_cdf.set_ylabel('Fraction of total intensity')
    ax_cdf.set_yticks(np.linspace(0, 1, 5))

    fig.tight_layout()
    plt.show()
    break
```

## Appendix 4 - hyperparameter tuning.

```
#different parameters for required classifiers
param_grid = [{'C': [1, 10, 100, 1000],'gamma': [0.001, 0.0001
], 'kernel': ['rbf']}]

#classifiers
svc = SVC()

scores = ['precision', 'recall']

for score in scores:
    print("Tuning hyper-parameters for %s" % score)
    print()

    clf = GridSearchCV(svc, param_grid, verbose = 2, scoring='
%s_macro' % score)
    clf.fit(x_train_std, y_train)

    print("Best parameters set found:")
    print()
    print(clf.best_params_)
    print()
    print("Grid scores on development set:")
    print()
    means = clf.cv_results_['mean_test_score']
    stds = clf.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, clf.cv_results_[
'params']):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()

    print("Detailed classification report:")
    print()

    y_true, y_pred = y_test, clf.predict(x_test_std)
    print(classification_report(y_true, y_pred))
    print()[32]
```