

INDUSTRIAL TRAINING REPORT

At

All E Technologies

Submitted in partial fulfilment of the
Requirements for the award of
Degree of Bachelor of Technology in Information Technology



Submitted By

Name: Eshika Agarwal

University Roll No.: 20151203117

SUBMITTED TO:
Department of Information Technology
BHARATI VIDYAPEETH'S COLLEGE OF
ENGINEERING
NEW DELHI

CANDIDATE'S DECLARATION

I hereby declare that the Industrial Training Report is an authentic record of my work as requirements of Industrial Training during the period from 31st March to 17th June for the award of the degree of B.Tech. (Information Technology), Bharati Vidyapeeth's College of Engineering, New Delhi.

Eshika Agarwal
20151203117

Date: _____

CERTIFICATE



Dated: June 17, 2020

To Whomsoever It May Concern

This is to certify that Ms. Eshika Agarwal, D/O Navneet Kumar Agarwal, a student of Bharati Vidyapeeth College of Engineering underwent Internship Programme starting from March 31, 20 and ending on June 17, 2020 at All e Technologies Pvt. Ltd. in the IT Services department. During Internship she worked in technologies like Machine Learning with Python and performed the following task:

1. Developed modules for extracting features from images.
2. Developed APIs for demand forecast using features supplied by customer.
3. Developed basic understanding of Flask, Django and SQL primarily through self-learning.

During her tenure we found her conduct to be good and we wish her best of luck for her future endeavors.

This letter is being issued for the purpose of Industrial Experience for University purpose and need not to be used for any other purpose.

Sincerely
For All e Technologies Pvt. Ltd.


Authorized Signatory

All e Technologies (P) Ltd.

A-1, Sector 58, Noida - 201301, INDIA Tel.: + 91 120 3000 300 Fax : + 91 120 2588 660 www.alletec.com
Registered Office : UU-14, Vishakha Enclave, Pitampura, New Delhi-110034 CIN : U72200DL2000PTC106331



Scanned with
CamScanner

ACKNOWLEDGEMENT

I would like to extend my heartiest thanks with a deep sense of gratitude and respect to all those who provide me with immense help and guidance during my training period. First I would like to extend my heartiest thanks with a deep sense of gratitude to **Mr. Sanjeev Thukral** for allowing me to undergo the Training at All E technologies Noida.

I am grateful to my Trainers **Mr. Chetan Kargeti, and Mr. Maneesh Kumar**, for helping me to learn the various concepts and without the guidance of whom the completion of the Training would have been difficult.

Last but not least, I pay my sincere thanks and gratitude to all the staff members of the Training Institute for their support and for making my training valuable and fruitful.

Eshika Agarwal
20151203117

ORGANIZATION

All E Technologies

All E Technologies mainly works in the domain of Nav dynamics.

All e Technologies (Alletec) – a Microsoft Dynamics Gold Partner – has been engaged in providing Business Technology Solutions to Growth Companies since 2000. Over 600 customers, from new age e-Commerce businesses to large enterprises, diverse industries and geographies – are a testimony to the transformations our solutions and services have helped them bring around their businesses. Our ecosystem of customers, competitors and Microsoft see us as one of the most reliable providers of complex business applications including – ERP & CRM (Microsoft Dynamics), Mobility Applications, Custom applications on Microsoft stack, Azure Infrastructure & Platform, SharePoint, Power BI, and Power Apps. Managed by a leadership team that has always valued professional ethics over short-term business gains, Alletec has been recognized by Microsoft consistently since 2004 to be amongst their top Business Solutions partners in India, and part of their global elite ‘Inner Circle’ and ‘President's Club’ several

times. Alletec has been servicing customers in APAC, Europe, USA, Africa, and Middle-East, besides India. The company has also been engaged by Microsoft on several product engineering projects. Dozens of customers have been using our industry solutions for eCommerce, Travel, Healthcare, EPC, Dealer Management, Specialty Chemical Manufacturing, and Building Material industries. They have started a new branch as all e emerge which deals AI /ML

TABLE OF CONTENTS

Candidate's Declaration.....	1
Certificate.....	2
Acknowledgement.....	3
Organisation.....	4
Table of Contents.....	6
Table of Figures.....	7
Chapter 01 Internship	
1.1 Project.....	9
1.2 Programming Languages... ..	11
1.3 Dataset.....	12
1.4 Texture Extractor.....	13
1.5 Colour Extractor.....	20
1.6 Flask API.....	29
CONCLUSION.....,,.....	30
BIBLIOGRAPHY.....,,,,,.....	31

TABLE OF FIGURES

FIGURE1 (Page No: 11)	Demo
FIGURE 2 (Page No: 24)	Output
FIGURE 3 (Page No: 29)	Dataset

CHAPTER 01: Internship

Worked on a Demand forecasting model.

Accuracy of demand prediction can have significant impact on any business. It can help both – enhance revenues, and reduce costs.

Statistical forecasting methods merely present historical trends in demand but fail to conclusively predict future demand of a product with an ever changing external environment.

Any product's ability to sell depends on a variety of factors such as Product Characteristics including its non-visual & visual features, External Factors such as weather, events etc and Historical Demand.

Most demand prediction solutions fail to address Product's Visual characteristics and External Factors.

1.1PROJECT

- Visual Features are extracted using Image recognition techniques.
- User can select **External data** that influences demand.
- **Weather** features include temperature, rainfall, snowfall and humidity.
- **Events** related to festivals, sporting events, promotions & competitor events can be specified.
- **Social media sentiment** influencers can also be selected..
- Custom features, which are **specific to a customer's business** e.g. Sub-brand, can be added to demand features
- Finally, demand features related to Sales such as **Sales Volume, Price, Discounts** etc. can be selected.
- For example, in the case of shoes, shown features were extracted from product images of shoes. Features as shown in the image were extracted from the shoe image-
:



Figure 1 DEMO

OUTPUT FILE-:

Output generated was added to the excel sheet with meta data for prediction -:

A	B	C	D	E	F	G	H	I	J	K
1	ID SKU	Model	Product	Price	Polarized	Frame Shape	Lens Color	Has Nosepads	Frame Color	Has Top Bar
2	1 ORB3584N 90531U 58-13	ORB3584N_90531U	Blue/Silver Mirror Mirror	9790	No	aviators	lightsteelblue	True	grey	True
3	2 ORB3584N 001/19 58-13	Aviator Blaze	Light Blue Gradient Gradient	9290	No	aviators	burlywood	True	darkkhaki	True
4	3 ORB3584N 004/13 58-13	Aviator Blaze	Brown Gradient Gradient	9290	No	aviators	dimgrey	True	dimgrey	True
5	4 ORB3584N 153/11 58-13	Aviator Blaze	Grey Gradient Gradient	9290	No	aviators	darkslategrey	True	dimgrey	True
6	5 RB3025 001/33	Aviator Classic	Brown Classic B-15	7690	No	aviators	saddlebrown	True	silver	True
7	6 RB3025 001/41 58-14	Aviator Classic	Brown	7490	No	aviators	sienna	True	gainsboro	True
8	7 RB3025 001/57 58-14	Aviator Classic	Brown Polarized	7490	Yes	aviators	darkolivegreen	True	dimgrey	True
9	8 RB3025 001/57 62-14	Aviator Classic	Brown Polarized	9190	Yes	aviators	saddlebrown	True	silver	True

Figure 2 OUTPUT

1.2. Programming Languages

There are two programming languages we worked within this course:

Python: It is an high-level and general-purpose programming language. Its design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.[1] Python provides various libraries that saves us from writing redundant code. It is widely used language in the field of machine learning and data science. We made our project using jupyter.

Flask: It is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions . It help us to create restfull API using python .

MySQL- It is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for mySQL however, is for the purpose of a web database. We have connected our flaks app with mysql server to extract information using pymysql.

1.3DATASET USED-:

Nike Shoe images were used for extraction of features . We had 33 images in the dataset on which various extractors for eg texture, colour and dimension were applied on and the data extracted was then added to the csv file which the meta sales data.



Figure 3 DATASET

1.4TEXTURE EXTRACTOR

Haralick Texture is used to quantify an image based on texture. It was invented by Haralick in 1973 and you can read about it in detail [here](#). The fundamental concept involved in computing Haralick Texture features is the Gray Level Co-occurrence Matrix or GLCM.

Gray Level Co-occurrence matrix (GLCM) uses adjacency concept in images. The basic idea is that it looks for pairs of adjacent pixel values that occur in an image and keeps recording it over the entire image. Below figure explains how a GLCM is constructed.

As you can see from the above image, gray-level pixel value 1 and 2 occurs twice in the image and hence GLCM records it as two. But pixel value 1 and 3 occurs only once in the image and thus GLCM records it as one. Of course, I have assumed the adjacency calculation only from left-to-right. Actually, there are four types of adjacency and hence four GLCM matrices are constructed for a single image. Four types of adjacency are as follows.

- Left-to-Right
- Top-to-Bottom
- Top Left-to-Bottom Right
- Top Right-to-Bottom Left

From the four GLCM matrices, 14 textural features are computed that are based on some statistical theory.[4] All these 14 statistical features needs a separate blog post. So, you can

read in detail about those here. Normally, the feature vector is taken to be of 13-dim as computing 14th dim might increase the computational time.

STEPS TO EXTRACT TEXTURE (LEATHER OR NET CLOTH)

- Import libraries.
- Give the path to training dataset.
- Declare variable as empty lists to hold feature vectors and labels.
- Define a function that takes an input image to compute haralick texture.
- It extracts the haralick features for all 2 types of adjacency .(Leather and net cloth)
- finds the mean of all 4 types of GLCM.
- Return the resulting feature vector for that image which describes the texture.
- Loop over the training labels we have just included from training directory.
- Give the path to current image class directory.
- holds the current image class label.
- takes all the files with .jpg as the extension and loops through each file one by one.
- reads the input image that corresponds to a file.
- converts the image to grayscale.
- extracts haralick features for the grayscale image.
- appends the 13-dim feature vector to the training features list.
- appends the class label to training classes list.

CODE-:

```
import cv2
import numpy as np
import os
import glob
import mahotas as mt
from sklearn.svm import LinearSVC

TRAIN_PATH = "C:\\tmp\\visual_model\\texture\\train"
TEST_PATH = "C:\\tmp\\visual_model\\shoes"
ALLOWED_EXTENSION = [".jpg", ".png"]

TEXTURE_KEY = "Texture"

class HaralickTextureExtractor:

    def __init__(self,
allowedExtensions=ALLOWED_EXTENSION,
trainPath=TRAIN_PATH, testPath=TEST_PATH ):

        self.allowedExtensions = allowedExtensions
        self.trainPath = trainPath
        self.testPath = testPath
        self.showImages = False
        # function to extract haralick textures from an image

    def extract_features(self, image):
```



```
# calculate haralick texture features for 4 types of  
adjacency
```

```
textures = mt.features.haralick(image)
```

```
# take the mean of it and return it
```

```
ht_mean = textures.mean(axis=0)
```

```
return ht_mean
```

```
def startExtaction(self,container):
```

```
# load the training dataset
```

```
train_names = os.listdir(self.trainPath)
```

```
# empty list to hold feature vectors and train labels
```

```
train_features = []
```

```
train_labels = []
```

```
# loop over the training dataset
```

```
i = 1
```

```
print ("[STATUS] Started extracting haralick textures..")
```

```
for train_name in train_names:
```

```
    cur_path = self.trainPath + "/" + train_name
```

```
    cur_label = train_name
```

```
    i = 1
```

```
    for extension in self.allowedExtensions:
```

```
        for file in glob.glob(cur_path + "/*"+extension):
```

```
            print ("Processing Image - {} in {}".format(i,  
cur_label))
```

```
            # read the training image
```

```

image = cv2.imread(file)

# convert the image to grayscale
gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)

# extract haralick texture from the image
features = self.extract_features(gray)

# append the feature vector and label
train_features.append(features)
train_labels.append(cur_label)

# show loop update
i += 1

# have a look at the size of our feature vector and labels
print ("Training features:
{ }".format(np.array(train_features).shape))
print ("Training labels:
{ }".format(np.array(train_labels).shape))

# create the classifier
print ("[STATUS] Creating the classifier..")
clf_svm = LinearSVC(random_state=9)

# fit the training data and labels
print ("[STATUS] Fitting data/label to model..")
clf_svm.fit(train_features, train_labels)

```

```

# loop over the test images
for extension in self.allowedExtensions:
    for file in glob.glob(self.testPath + "/*"+extension):
        # read the input image
        image = cv2.imread(file)
        fileName = os.path.basename(file)

        textureData = None
        if fileName in container.keys():
            textureData =container[fileName]
        else:
            textureData = {}
            container[fileName]= textureData
        # convert to grayscale
        gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)

        # extract haralick texture from the image
        features = self.extract_features(gray)

        # evaluate the model and predict label
        prediction = clf_svm.predict(features.reshape(1, -
1))[0]

        # show the label
        cv2.putText(image, prediction, (20,30),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)
        print ("Prediction - {}".format(prediction))

        textureData[TEXTURE_KEY] = prediction

```

```
# display the output image
if self.showImages:
    cv2.imshow("Test_Image", image)
    cv2.waitKey(0) n
```

1.5COLOR EXTRACTOR

We'll start by importing our necessary packages. We'll use NumPy for numerical processing, argparse to parse our command line arguments, and cv2 for our OpenCV bindings. Then we handle parsing our command line arguments. We'll need just a single switch to load our image off disk. Then we define this list of colors: All we are doing here is defining a list of boundaries in the RGB color space (or rather, BGR, since OpenCV represents images as NumPy arrays in reverse order), where each entry in the list is a tuple with two values: a list of lower limits and a list of upper limits.[5] For example, let's take a look at the tuple ([17, 15, 100], [50, 56, 200]) Here, we are saying that all pixels in our image that have a R \geq 100, B \geq 15, and G \geq 17 along with R \leq 200, B \leq 56, and G \leq 50 will be considered red. Now that we have our list of boundaries, we can use the cv2.inRange function to perform the actual color detection. To create the output image, we apply our mask. This line simply makes a call to cv2.bitwise_and showing only pixels in the image that have a corresponding white (255) value in the mask.

CODE-:

```
import cv2;
import os
import numpy as np;
import csv
from collections import OrderedDict
from sqlalchemy.sql.expression import false
```

```

VALID_EXTENSIONS
=[".jpg",".jpeg",".gif",".bmp",".png"]
OUT_COLUMNS    =  {'File Name':1,  'Color
Detected':2,'Color Proportion (%)':3}

```

```

class ColorExtractor:

```

```

    def __init__(self, imagePath):
        self.imagePath = imagePath
        self.showImage = False

```

```

    def
extractColorForAllImagesToFile(self,filePath,fileName='outputColorInfo.csv'):
        imageFileList = os.listdir(self.imagePath)
        print("Image list found ",imageFileList)

        #ordered_fieldnames    =  OrderedDict([("File
Name",None),("Color      Detected",None),("Color
Proportion (%)",None)])
        outFile = os.path.join(filePath, fileName)
        fWrite = open(outFile,'w')
        #writer = csv.writer(fWrite)
        #writer.writerow(OUT_COLUMNS)

```

```

        writer =
csv.DictWriter(fWrite,fieldnames=OUT_COLUMN
S.keys())
        writer.writeheader()

        imageInfo = None
        imageInfoArr = []
        for imageFile in imageFileList:

            imageInfo = { }
            print("Processing ", imageFile)
            imageInfo["File Name"]=imageFile

        fileName,extension=os.path.splitext(imageFile)
        if extension in VALID_EXTENSIONS:
            absFile    =    os.path.join(self.imagePath,
imageFile)
            imgGray = im_in = cv2.imread(absFile,
cv2.IMREAD_GRAYSCALE);
            imgColor = cv2.imread(absFile);
            th, im_th = cv2.threshold(imgGray, 220,
255, cv2.THRESH_BINARY_INV);

            # Copy the thresholded image.
            im_floodfill = im_th.copy()

            # Mask used to flood filling.
            # Notice the size needs to be 2 pixels than
the image.

```

```

h, w = im_th.shape[:2]
mask = np.zeros((h+2, w+2), np.uint8)

# Floodfill from point (0, 0)
cv2.floodFill(im_floodfill, mask, (0,0),
255);

# Invert floodfilled image
im_floodfill_inv =
cv2.bitwise_not(im_floodfill)

# Combine the two images to get the
foreground.
im_out = im_th | im_floodfill_inv

mask_inv = cv2.bitwise_not(im_out)
img2_fg =
cv2.bitwise_and(imgColor,imgColor,mask =
im_out)

text=imageFile+"      Image      After
Mask"+str(img2_fg.shape)

self.detectColor(img2_fg,imageInfo)
writer.writerow(imageInfo)
if c.showImage:
    cv2.imshow(text, img2_fg)
    cv2.waitKey(0)
#imageInfoArr.append(imageInfo)

```



```

        #cv2.waitKey(0)
    else:
        print ("File "+imageFile+" has invalid
extension,",extension," so cannot be processed.")
        #writer.writerow(imageInfoArr)
        print("File ",outFile," created with color
information")
        fWrite.close()

```

```

#,
def addColorData(self, oDict, fileInfoDict):
    colorCount=0
    #comment ="Shoe is of"
    stats = "Shoe: "
    colors=[]
    proportions=[]
    fileInfoDict["Color Detected"] = colors
    fileInfoDict["Color Proportion (%)"] =
proportions

```

```

for key in oDict.keys():

```

```

    if (oDict[key]>1):

```

```

        stats
    stats+=str(key)+"("+str(oDict[key])+"%), "
    colors.append(str(key))
    proportions.append(str(oDict[key]))

    print (" Color info: ",fileInfoDict)
    return stats

```

```

def detectColor(self,image, fileInfoDict):
    boundaries=[ ([17, 15, 100], [50, 56,
200],["red",1.7]),
    ( [-10, -10, 214], [ 10, 10, 294],["light
red",1]),
    ( [ 80, 41, 214], [100, 61, 294], ["light-pink",1]),
    ( [141 , 41, 215], [161 , 61, 295],["pink",1]),
    ( [ 20 ,240, 10], [ 40 ,260, 90],["light-green",1]),
    ( [ 89 ,206 ,117], [109 ,226, 197],["light blue",1]),
    ( [ 74, 245 ,214], [ 94, 265, 294],["light-
yellow",20]),
    ([25, 146, 190], [62, 174, 250],["darkish
yellow",5]),
    ( [ 1, 238, 214], [ 21, 258, 294],["yellow",20]),
    ([ 10 ,197, 123], [ 30, 217, 203],["dark-
yellow",2]),
    ( [ 20 ,116, 61], [ 40 ,136, 141],["military-
green",10]),
    ([ 20 , 74, 112], [ 40, 94, 192],["brown",1]),
    ( [ 5, 159, 112], [ 25, 179, 192],["khaki",1]),
    ([110, 74, 112], [130 , 94, 192],["purple",10]),

```

```

( [-10, 166, 214], [ 10, 186, 294],["orange",1]),
([ 0, 143, 214], [ 20, 163, 294],["light-orange",1]),
([140, 240, 62], [160, 260, 142],["sea-green",1]),
([ 84, 168, 69], [104, 188, 149],["bluish-
green",1]),
    ([0, 0, 0], [20, 20, 25],["black",1.5]),
#    ([17, 15, 100], [50, 56, 200],["red",1.7]),

([86, 31, 4], [220, 100, 70],["blue",.8]),

    ([103, 86, 65], [145, 133, 128],["grey",1.5]),
    ([195, 195, 170], [240, 240,
240],["white",0.2])

]

```

```

i =0
# loop over the boundaries
colorDict = {}
cntTotal=1
for (lower, upper,color) in boundaries:
    # create NumPy arrays from the boundaries

    lower = np.array(lower, dtype = "uint8")
    upper = np.array(upper, dtype = "uint8")
    i=i+1

```

```

        # find the colors within the specified
boundaries and apply
        # the mask
        mask = cv2.inRange(image, lower, upper)
        output = cv2.bitwise_and(image, image,
mask = mask)

        cnt=np.count_nonzero(output)
        #print ("Color is ",color[0]," count is
",cnt*color[1])
        colorDict[color[0]]=cnt*color[1]
        cntTotal = cntTotal + cnt*color[1]
        # show the images

colorArr=[]
newColorDict={ }
for key in colorDict.keys():
    percentScore=colorDict[key]/cntTotal
    newScale = round(100*percentScore,2)
    colorDict[key]=newScale
    od = OrderedDict(sorted(colorDict.items(),
key=lambda x: x[1], reverse=True))
    #sorted(a_dict.items(), key=lambda item:
item[1][1])
    #for          key,          value          in
sorted(colorDict.iteritems(), key=lambda(k,v):(v,k)):
        # colorArr.append(key)
self.addColorData(od, fileInfoDict)

```

```
# USAGE EXAMPLE
```

```
PATH_OF_IMAGES= 'C:\\tmp\\nike'  
c = ColorExtractor(PATH_OF_IMAGES)  
#c.showImage = True  
c.extractColorForAllImagesToFile('C:\\tmp\\nike','color.csv')
```

```
##### END USAGE EXAMPLE
```

1.6 FLASK API

A restful flask Api was created which loaded all the extractors and loaded all the images and other data from the mysql database. On these images the extractor was applied to and the result obtained was then sent as an json file to be added to the output file .This file was then trained on and Regressor like XGBOOST and LASO were applied on the output file for prediction of the sales data . further, RMSE(Root Mean square Value) was also calculated in orderto check the accuracy and the value obtained was approx. 0.99 that is 99% accuracy.

LASO[2]-: Lasso regression is a type of **linear regression** that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of muticollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

XGBOOST[3]-: **XGBoost** is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way

CHAPTER 02: CONCLUSION

This Training that I undertook was truly a very rewarding experience for me in more than one way. It has given a big thrust to my technical knowledge and public speaking as a prospective Software professional. I have learned about the concepts in great detail and how to interact with the crowd. It has also helped me enhance my skills on the personal front.

And I feel extremely satisfied by the fact that I have managed to submit the report about my internship project.

I think I have exploited the opportunity that came my way to the fullest extent by increasing my technical knowledge and also gaining valuable work experience and also the art of public speaking apart from studying the other subjects in our curriculum.

BIBLIOGRAPHY

- [1] <https://opencv.org/>
- [2] <https://www.statisticshowto.com/lasso-regression/>
- [3] <https://xgboost.readthedocs.io/en/latest/>
- [4] <https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/>
- [5] <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>

