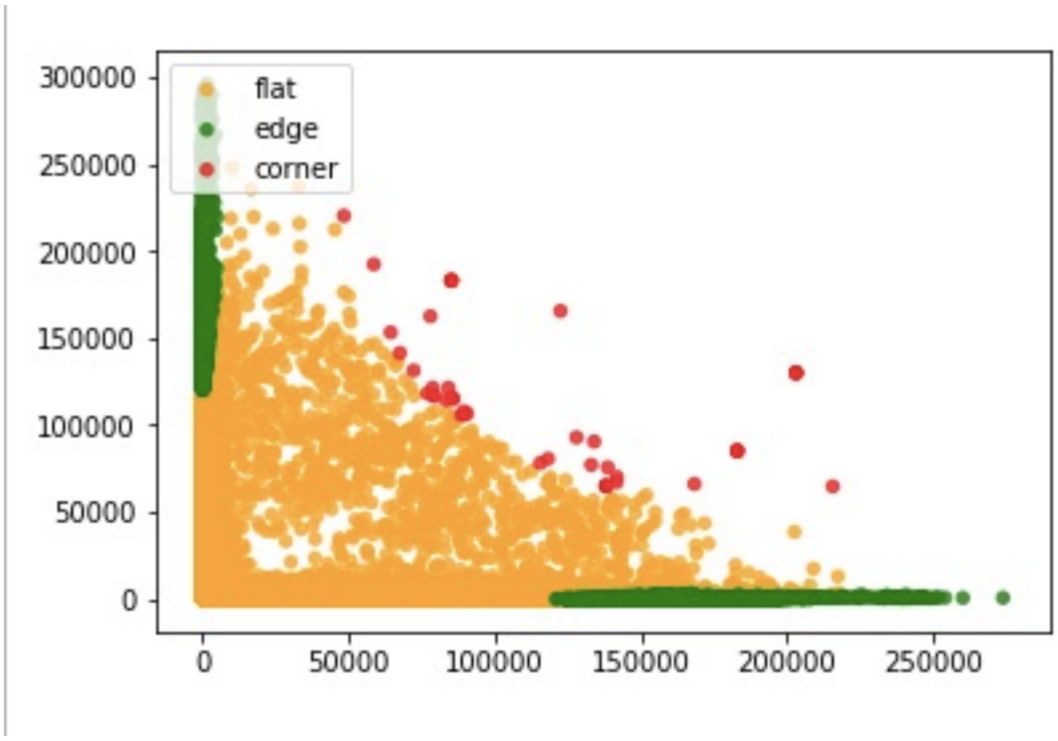
1. We find the Harris corner response initially. Later we find the regions of the Flat Corner and edge data. Running them on the given image we come to find that:
   Edges:12160
   Flat: 313995
   Corners: 45

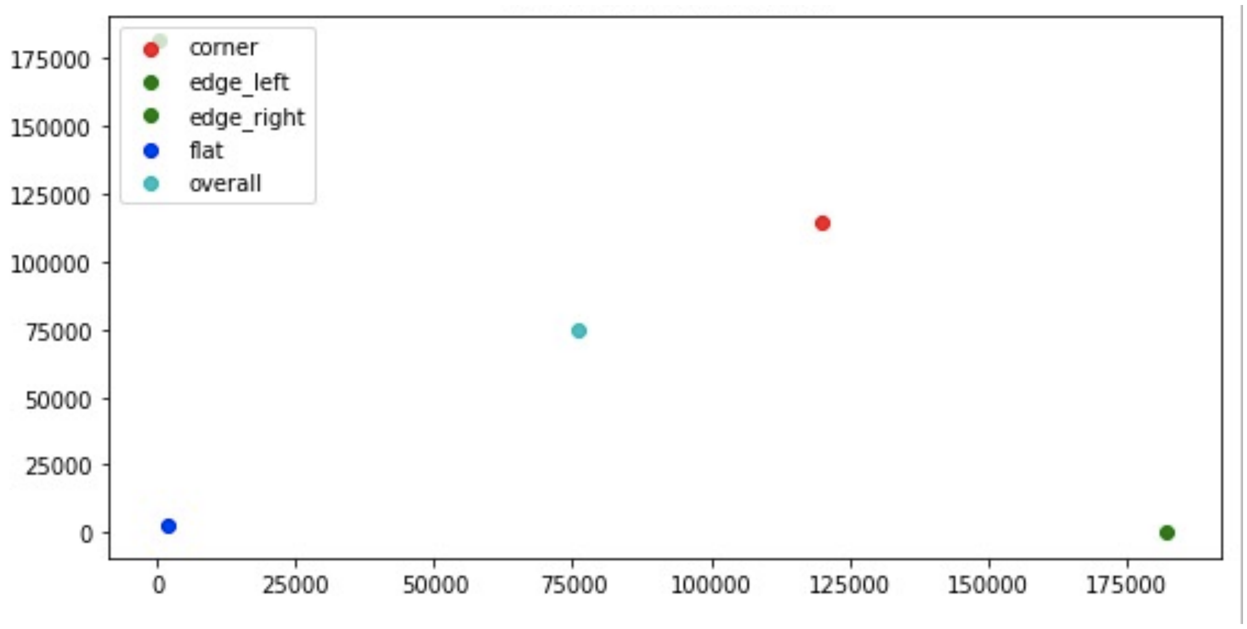


We now perform thresholding to get the region features values(Flat,Corner,Edges). This can be used to create the array for actual classifier results, and the distribution graph.

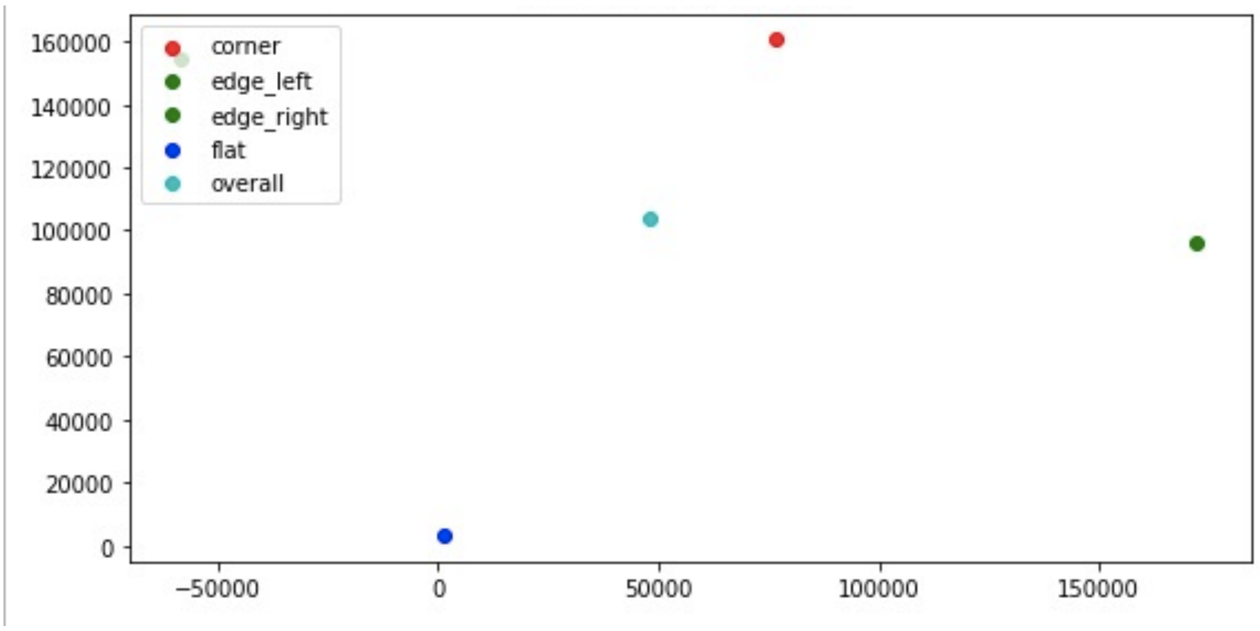
Best Threshold regions could be observed at corner : 8954090131.34 and edge : -714898662.639

Now we understand the Mean Variations. The following graphs help us understand the:

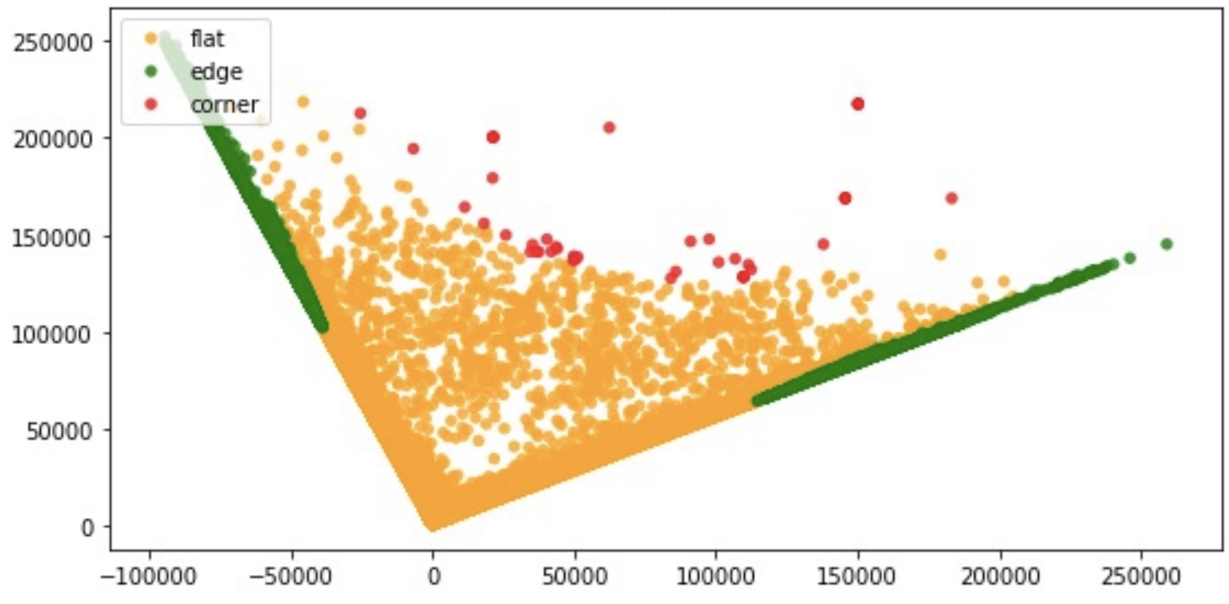
Fisher mean Distribution Plot:



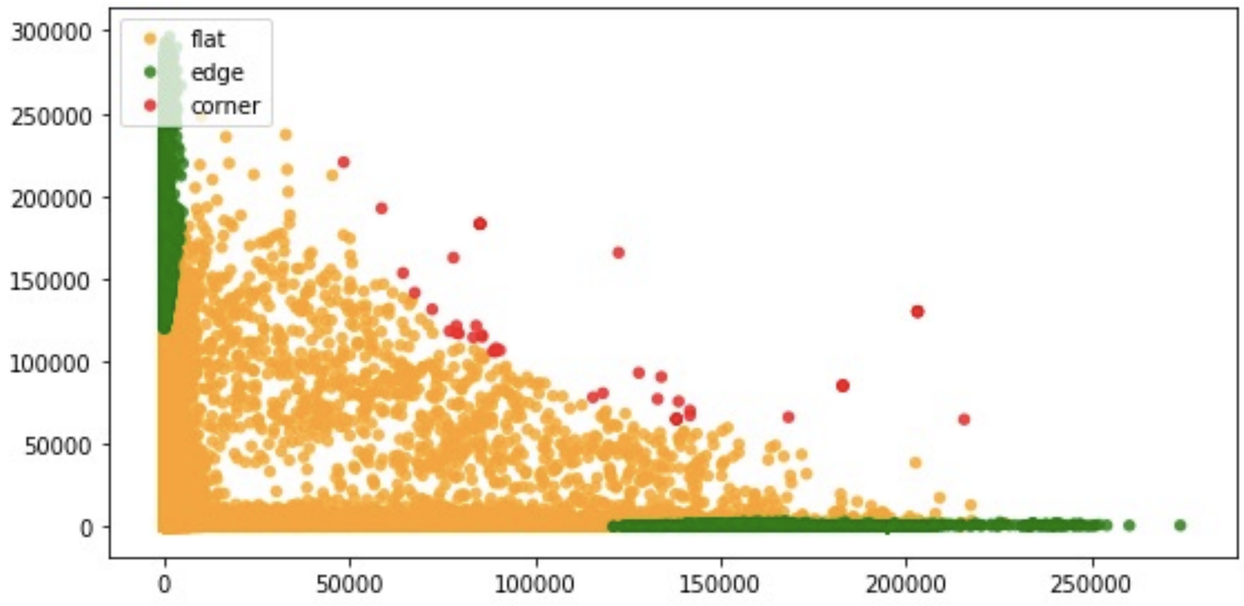
Actual mean Distribution plot:



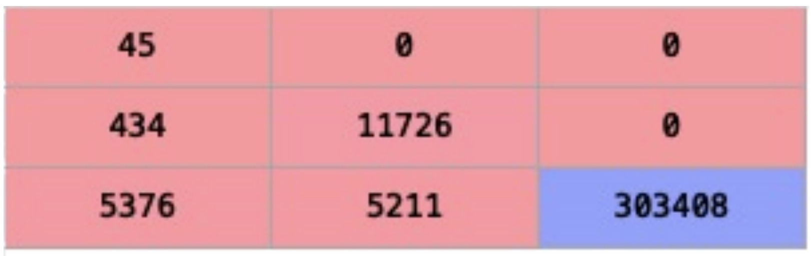
Fisher Space Distribution plot:



Actual Distribution plot:

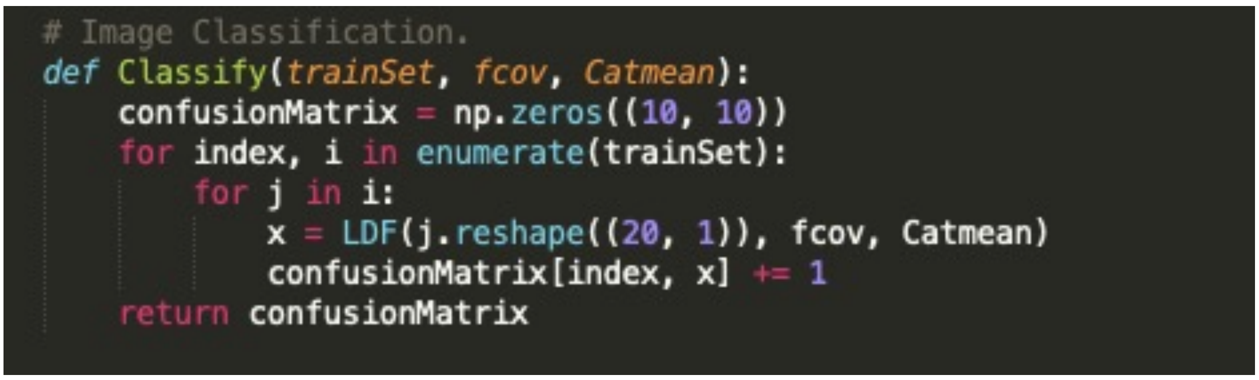


Observed Confusion matrix on classifying with Mahalanobis distance.

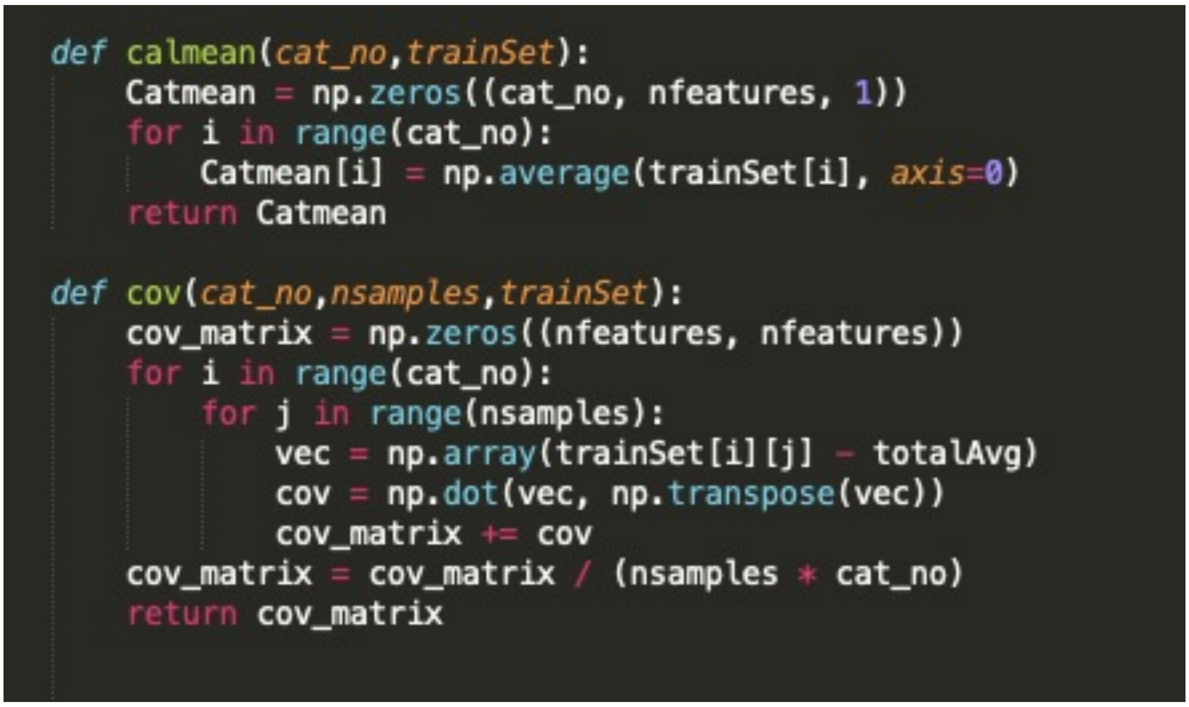


45	0	0
434	11726	0
5376	5211	303408

2.

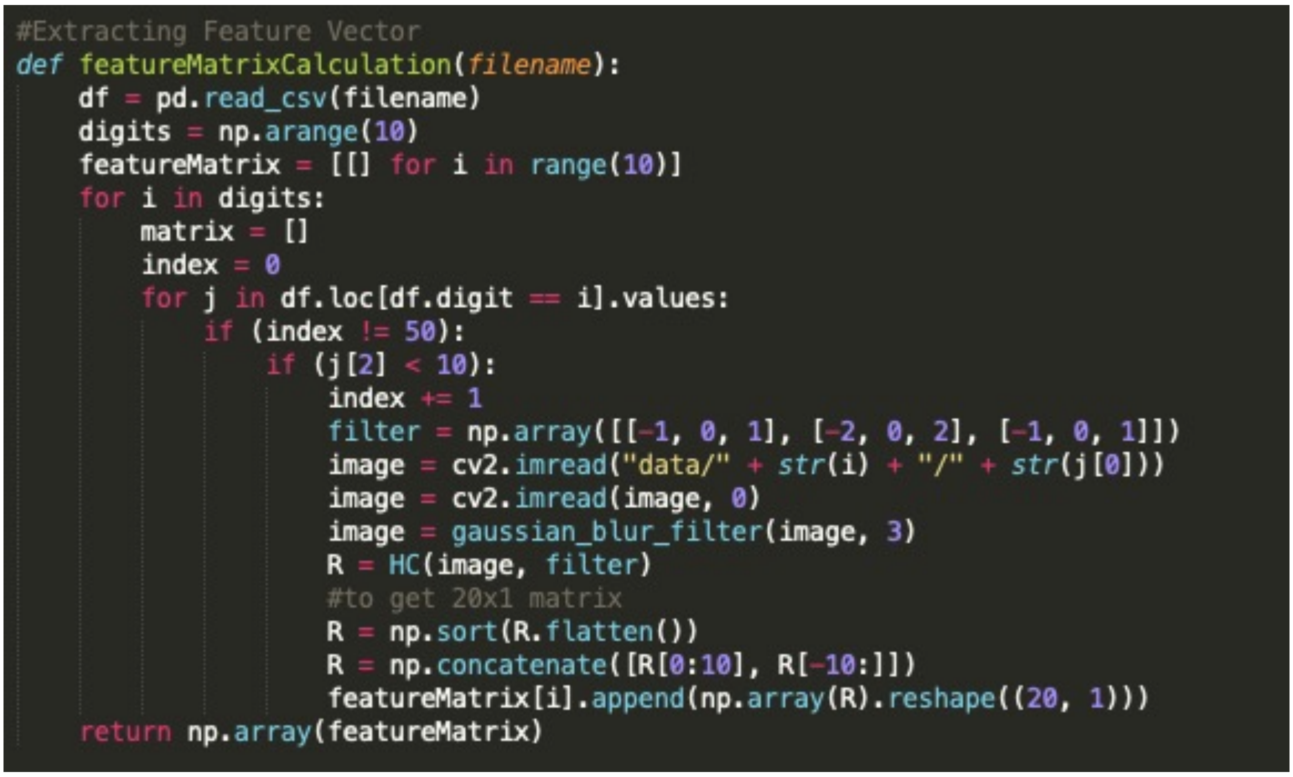


```
# Image Classification.
def Classify(trainSet, fcov, Catmean):
    confusionMatrix = np.zeros((10, 10))
    for index, i in enumerate(trainSet):
        for j in i:
            x = LDF(j.reshape((20, 1)), fcov, Catmean)
            confusionMatrix[index, x] += 1
    return confusionMatrix
```



```python
def calmean(cat_no,trainSet):
    Catmean = np.zeros((cat_no, nfeatures, 1))
    for i in range(cat_no):
        Catmean[i] = np.average(trainSet[i], axis=0)
    return Catmean

def cov(cat_no,nsamples,trainSet):
    cov_matrix = np.zeros((nfeatures, nfeatures))
    for i in range(cat_no):
        for j in range(nsamples):
            vec = np.array(trainSet[i][j] - totalAvg)
            cov = np.dot(vec, np.transpose(vec))
            cov_matrix += cov
    cov_matrix = cov_matrix / (nsamples * cat_no)
    return cov_matrix
```



```python
#Extracting Feature Vector
def featureMatrixCalculation(filename):
    df = pd.read_csv(filename)
    digits = np.arange(10)
    featureMatrix = [[] for i in range(10)]
    for i in digits:
        matrix = []
        index = 0
        for j in df.loc[df.digit == i].values:
            if (index != 50):
                if (j[2] < 10):
                    index += 1
                    filter = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
                    image = cv2.imread("data/" + str(i) + "/" + str(j[0]))
                    image = cv2.imread(image, 0)
                    image = gaussian_blur_filter(image, 3)
                    R = HC(image, filter)
                    #to get 20x1 matrix
                    R = np.sort(R.flatten())
                    R = np.concatenate([R[0:10], R[-10:]])
                    featureMatrix[i].append(np.array(R).reshape((20, 1)))
    return np.array(featureMatrix)
```