Machine Learning Project

Team-15

CS20B1001-A.LAHARI

CS20B1126-G.KAVYASRI

EC20B1116-P.SAHITHI

Face Recognition using PCA and KNN

Collecting data

Pre processing

Feature extraction

PCA

KNN Classification

Data Collection

Collected images of 13 south Indian actors with 13 different variations(expression, angles, color gradient , lighting)

Modules we got familiarized:

Glob module : used to return all file paths that match a specific pattern

Imutils : to make basic image processing functions

Zipfile : manipulate zip files

os path: contains useful functions on pathnames

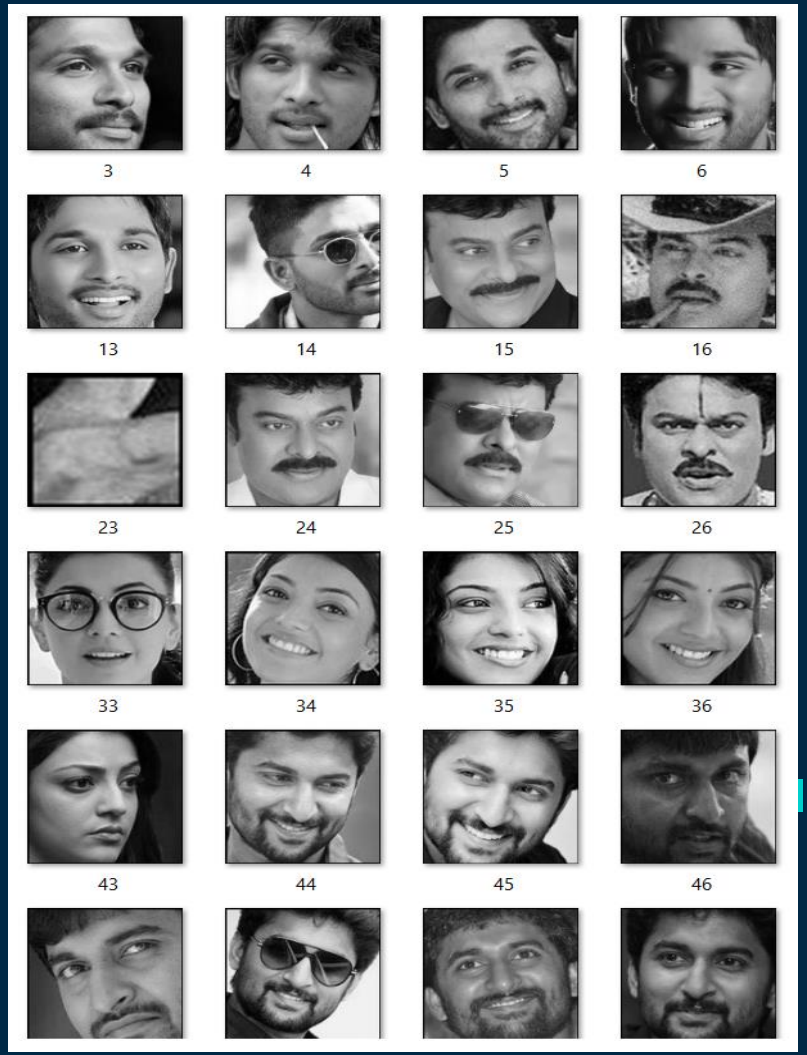
- We finally proceeded to use imutils module to resize,covert the images into greyscale

Data Pre-processing

- Extraction of faces from images :
  Haarcascading algorithm
- Converted into grayscale and resized(255x255) the images
- Stored the face images in the desired folder

```
r=1
for k in ip:
    i=cv2.imread(k)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +'haarcascade_frontalface_default.xml')
    faces=face_cascade.detectMultiScale(i, 1.1, 3);
    for (x, y, w, h) in faces:
            i1= cv2.cvtColor(i,cv2.COLOR_BGR2GRAY)
            cv2.rectangle(i1, (x,y), (x+w, y+h), (0,255,0), 2)
            roi_color=i1[y:y+h, x:x+w]
            cv2.imwrite(str(r)+'.jpg',roi_color)
            r=r+1
    cv2.imshow('img', i1)
    cv2.waitKey(10)
cv2.destroyAllWindows()
```

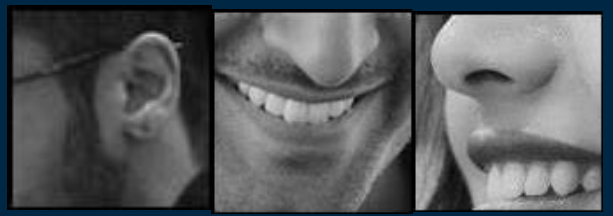
```
if not "ml" in os.listdir():
    os.mkdir("ml")
for i in li:
    img=PIL.Image.open(i)
    img=img.resize((255,255))
    img.save(i[:-4]+".jpg")
```



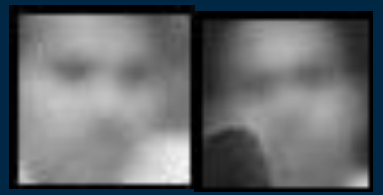
Issues Faced in pre-processing

- Failed to detect faces in some images instead some other parts of bodies were detected during Haarcascading.

- Only fraction of face is detected in some images.



- Some extra background faces were also detected and Consequently we got number images more than the actual images



Feature Extraction

Extracted pixels of image to numpy array.

Flattened the array to 1-D now these represent the feature vectors of each image.

Reduced the data to less features through PCA

Labelled our data to respective classes

```
from PIL import Image
l=[]
for k in v:
    i = Image.open(k)
    n = np.asarray(i)
    l.append(n)
arr = np.asarray(l)
arr
```

```
array([[[121, 116, 113, ..., 141, 141, 140],
        [120, 116, 113, ..., 142, 142, 142],
        [122, 118, 114, ..., 143, 143, 143],
        ...,
        [100,  94,  88, ..., 126, 121, 117],
        [ 98,  84,  88, ..., 128, 123, 122],
        [ 92,  75,  95, ..., 128, 121, 122]]

       [[123, 123, 123, ...,  86,  88,  92],
        [126, 127, 127, ...,  86,  87,  91],
        [129, 129, 129, ...,  87,  87,  90],
        ...,
        [ 90,  87,  85, ...,  52,  65,  73],
        [ 82,  90,  93, ...,  65,  65,  74],
```

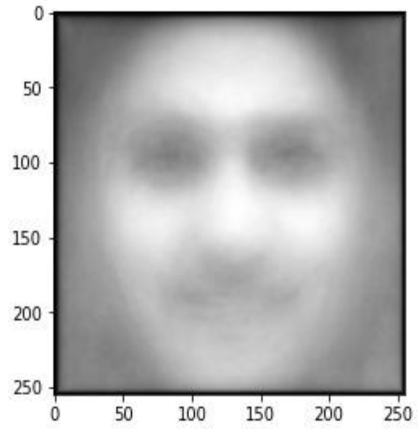
```
#actual data set
list1 = []
for j in arr:
    p = j.flatten()
    list1.append(p)

data = np.asarray(list1)
print(data.shape)
print(data)
```

```
(206, 65025)
[[ 1  1  1 ...  3  5  6]
 [ 0  0  0 ...  0  0  0]
 [ 2  0  9 ... 11  0  8]
 ...
 [ 0  0  0 ...  3  3  4]
 [ 0  2  1 ...  4 14  0]
 [ 1  4  2 ...  3 49  5]]
```

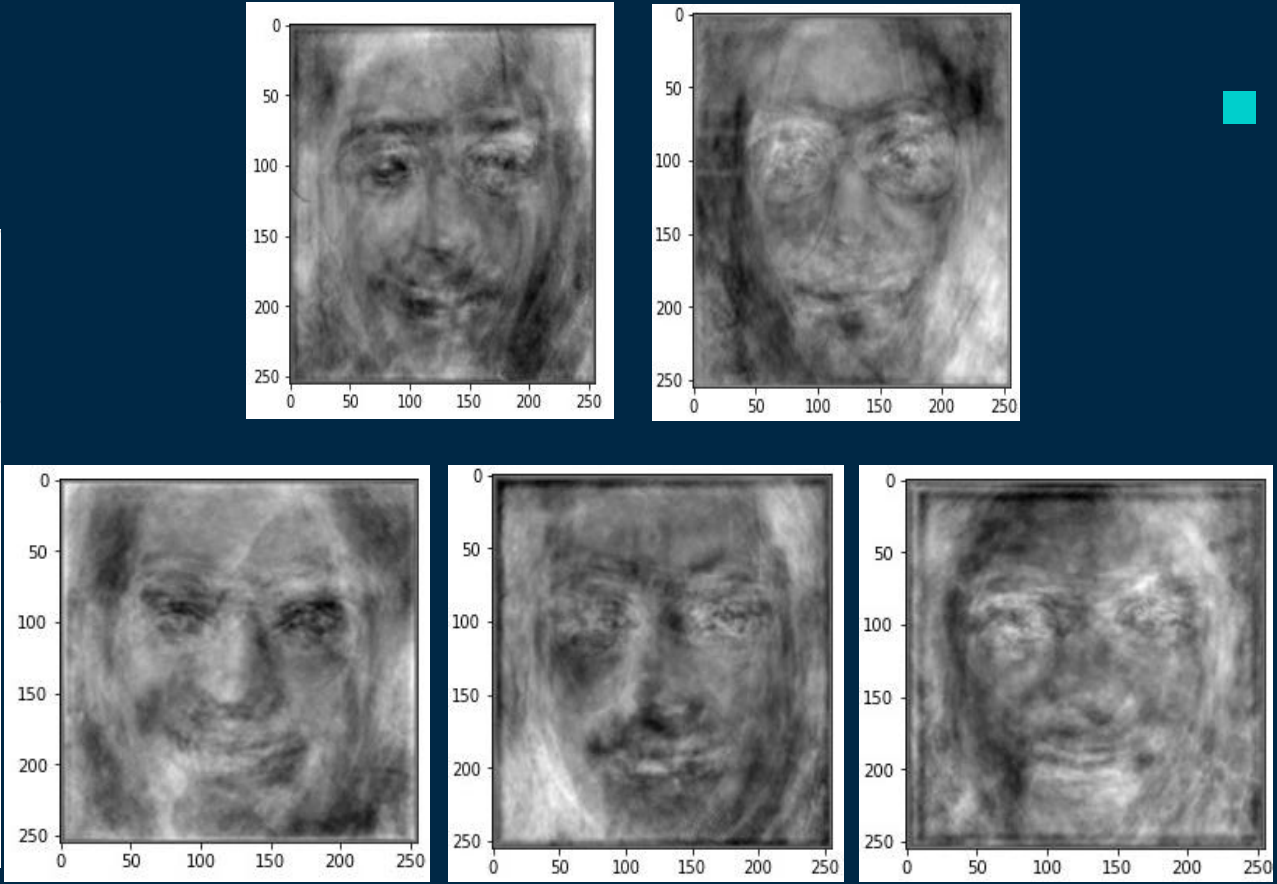
PCA

Mean Face

```python
# mean face
arr = np.array(mi)
arr_2d = np.reshape(arr, (255,255))
pixel_plot = plt.imshow(arr_2d,cmap="gray")
plt.show(pixel_plot)
```



Egien Faces



Issues Faced in PCA followed by labelling

- We initially performed pca without normalizing the pixel dataset and the accuracy was not satisfying. So we have to mean center the pixel value dataset.

- Ratio splitting
- Class labelling

K vs Accuracy

- KNN classification by feeding weighted distribution of eigen faces

```
accuracy for k=1 is 32.765748498987987
accuracy for k=3 is 21.354689625364656
accuracy for k=5 is 4.5687423283773748
accuracy for k=7 is 20.387638094678266
accuracy for k=9 is 7.8768726548748374
```

- KNN classification by feeding reduced dataset using pca

```
accuracy for k=1 is 33.33333333333333
accuracy for k=3 is 19.047619047619047
accuracy for k=5 is 14.285714285714285
accuracy for k=7 is 14.285714285714285
accuracy for k=9 is 14.285714285714285
```

KNN Classification

We performed K nearest neighbours Classifier for k=1

	y_Test	y_Pred
78	6.0	6.0
97	7.0	7.0
152	11.0	4.0
44	4.0	1.0
40	3.0	3.0
67	5.0	11.0
98	7.0	7.0
18	2.0	10.0
153	11.0	2.0
62	5.0	5.0
4	1.0	4.0
173	12.0	1.0
145	10.0	9.0
38	3.0	1.0
29	3.0	3.0
171	12.0	9.0
191	12.0	7.0
33	3.0	3.0
180	12.0	5.0
190	12.0	3.0
174	12.0	9.0

accuracy1 == 33.33333333333333

	precision	recall	f1-score	support
1.0	0.00	0.00	0.00	1
2.0	0.00	0.00	0.00	1
3.0	0.75	0.75	0.75	4
4.0	0.00	0.00	0.00	1
5.0	0.50	0.50	0.50	2
6.0	1.00	1.00	1.00	1
7.0	0.67	1.00	0.80	2
9.0	0.00	0.00	0.00	0
10.0	0.00	0.00	0.00	1
11.0	0.00	0.00	0.00	2
12.0	0.00	0.00	0.00	6
accuracy			0.33	21
macro avg	0.27	0.30	0.28	21
weighted avg	0.30	0.33	0.31	21

Wild Life Detection using PCA Logistic Regression and KNN Classification

Collecting data

Pre processing

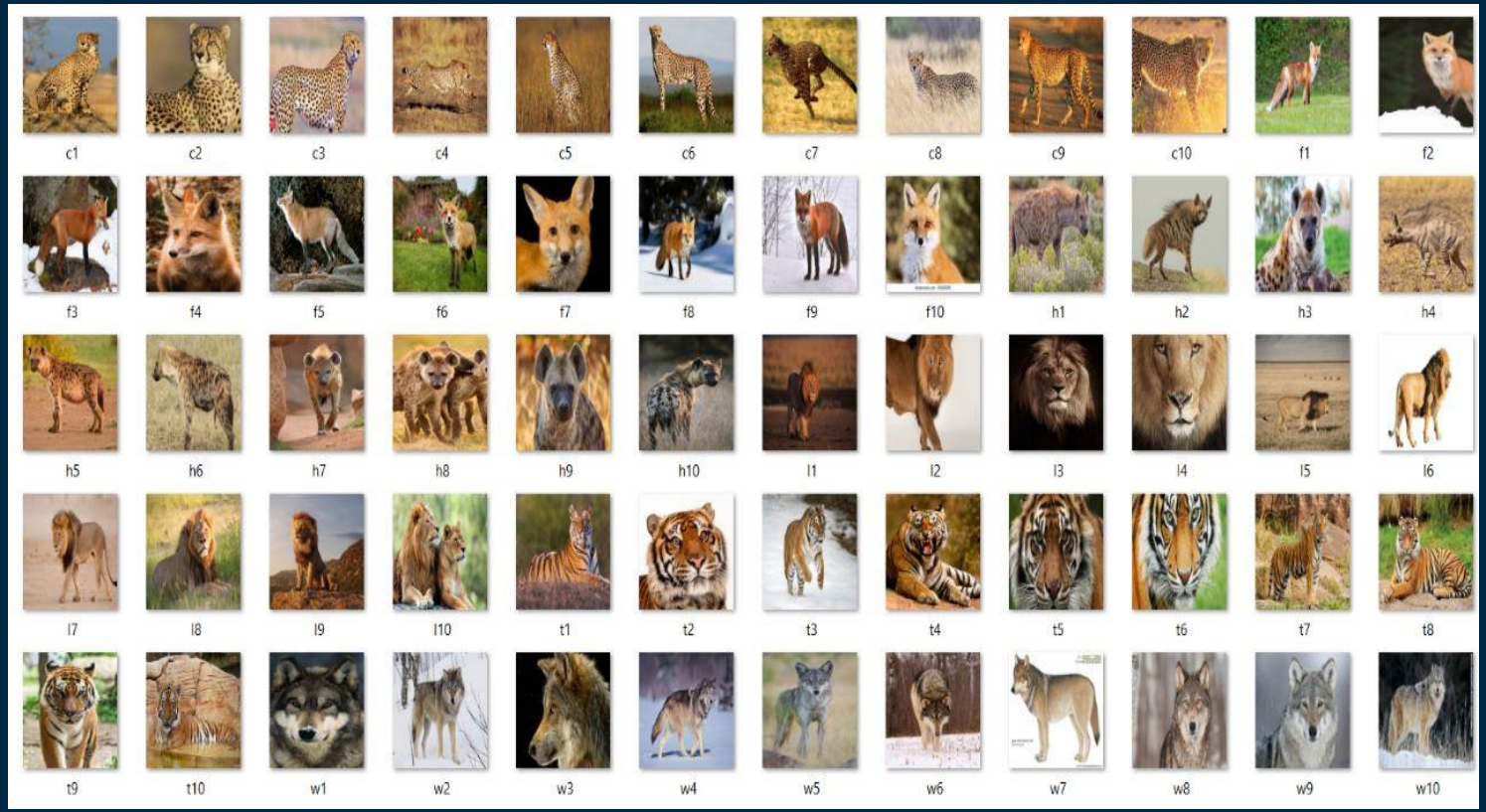
Feature extraction

PCA

Classification

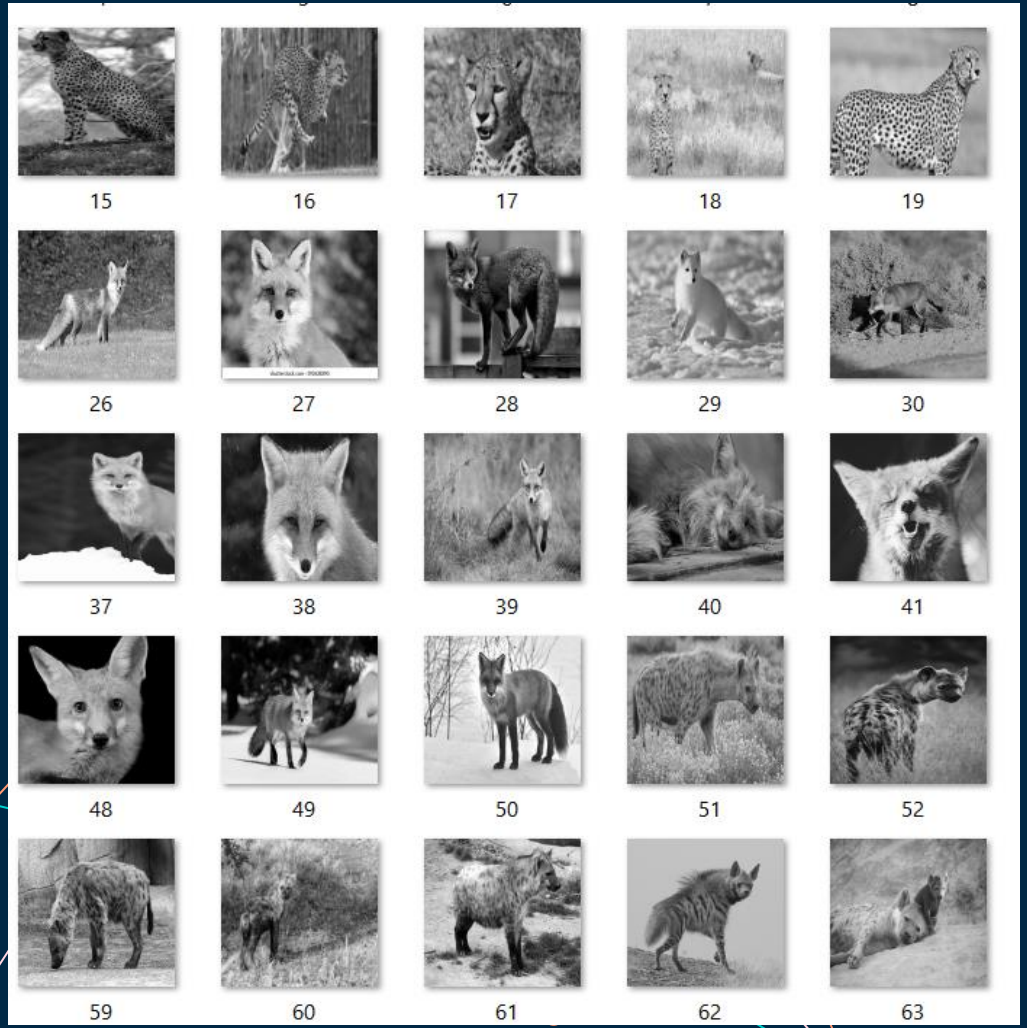
Data Collection

Dataset containing Images of Wild Animals from Kaggle



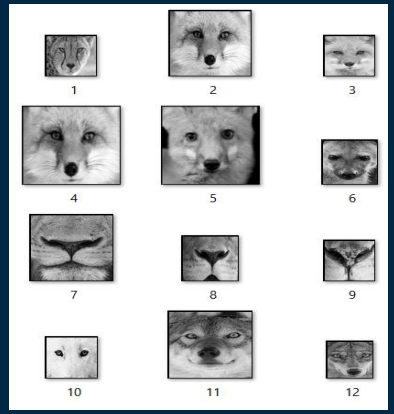
Data Pre-processing

- We collected a dataset from Kaggle which have 6 different animals of 25 each.
- We explored many face detection techniquies for the animals.
- Converted the images into grayscale and resized(224x224) the images.
- Stored the images in the desired folder.

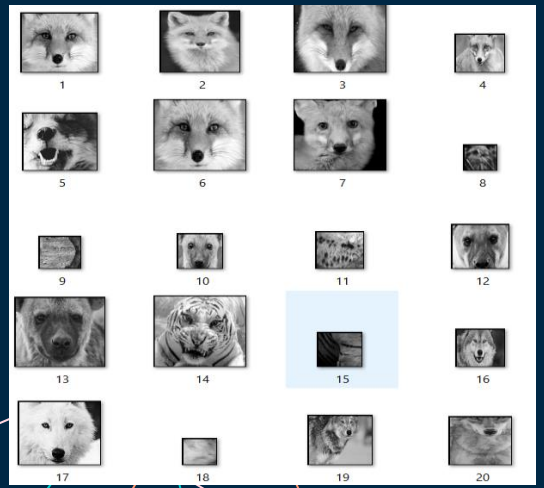


Issues in Data Pre-Processing

- We explored many algorithms for detecting animal faces like
  haarcascade_frontalcatface_extended.xml
  haarcascade_frontalcatface.xml



- But we were able to detect only few faces which belongs to specified species(say cat)
- Then we thought of doing haarcascading for human faces but the number of faces getting detected were still very less
- So we finally proceeded to process our images directly as animals entire body is a distinguishable factor



haarcascade_frontalface_default.xml

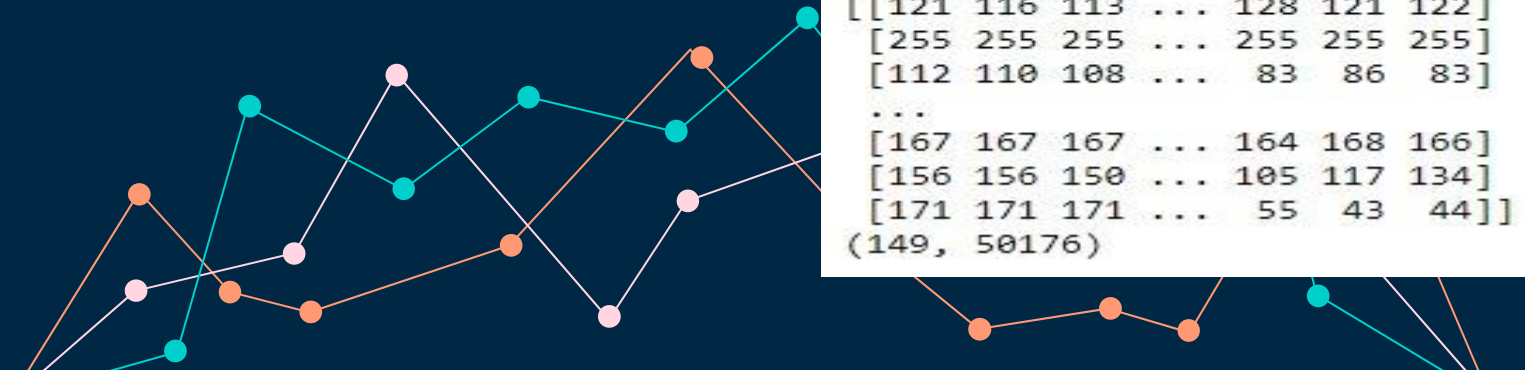
Feature Extraction

Extracted pixels of image to numpy array.

Flattened the array to 1-D now these represent the feature vectors of each image.

```
#actual data set
list1 = []
for j in arr:
    p = j.flatten()
    list1.append(p)

data = np.asarray(list1)
print(data)
print(data.shape)
```

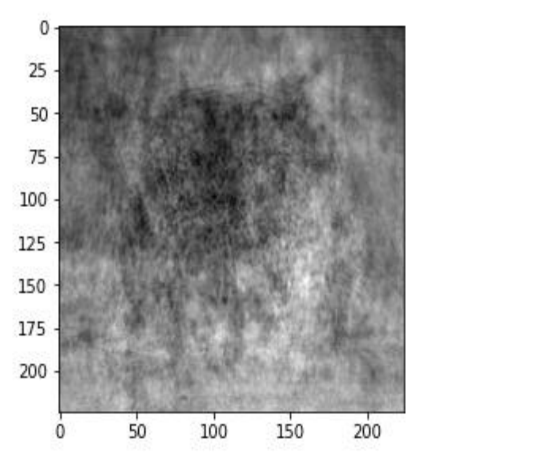
```
[[121 116 113 ... 128 121 122]
 [255 255 255 ... 255 255 255]
 [112 110 108 ...  83  86  83]
 ...
 [167 167 167 ... 164 168 166]
 [156 156 150 ... 105 117 134]
 [171 171 171 ...  55  43  44]]
(149, 50176)
```

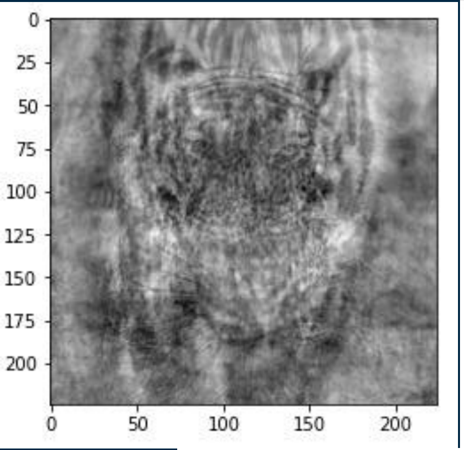


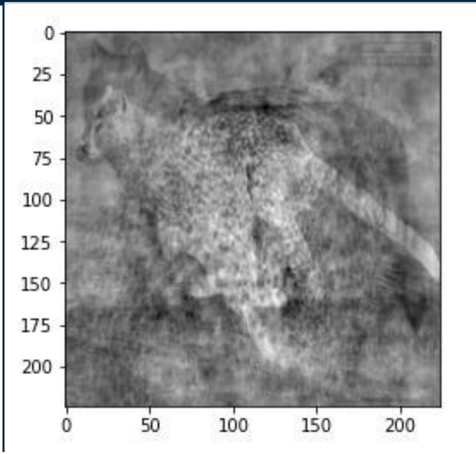
PCA

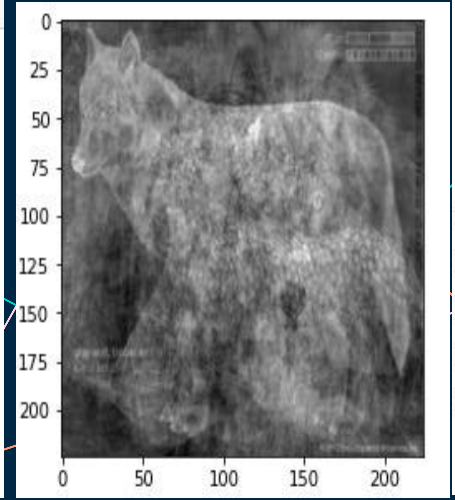
Mean Image and Eigen images

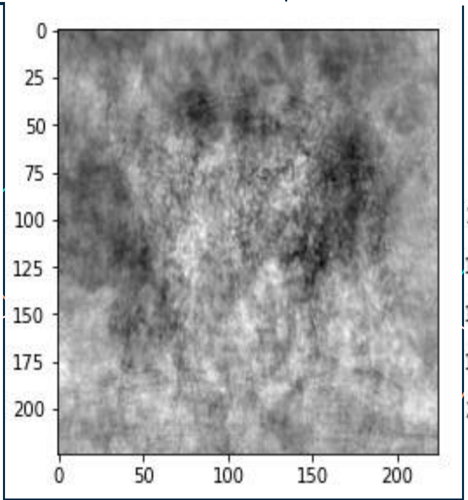
```
#mean vector

arr = np.array(mi)
# Convert 1D array to a 2D numpy array of 2
arr_2d = np.reshape(arr, (224,224))

pixel_plot = plt.imshow(arr_2d,cmap="gray")
plt.show(pixel_plot)
```

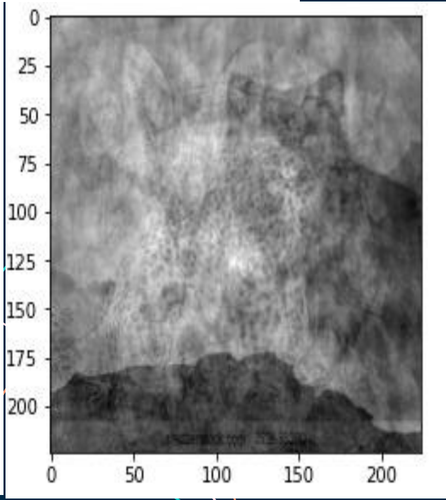












K vs Accuracy

- KNN via weighted distribution of eigen faces

```
accuracy for k=1 is 33.33333333333333
accuracy for k=3 is 33.33333333333333
accuracy for k=5 is 26.0
accuracy for k=7 is 22.876787978979887
accuracy for k=9 is 20.0
```

- KNN classification by feeding reduced dataset using pca

```
accuracy for k=1 is 40.0
accuracy for k=3 is 27.354689625364656
accuracy for k=5 is 19.5687423283773748
accuracy for k=7 is 10.387638094678266
accuracy for k=9 is 7.8768726548748374
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
reg = LogisticRegression()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)

print("Logistic Regression model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
```

```
Logistic Regression model accuracy(in %): 33.33333333333333
```

KNN Classification

```
accuracy1 == 40.0
```

	precision	recall	f1-score	support
1.0	0.50	1.00	0.67	2
2.0	0.50	0.25	0.33	4
3.0	0.00	0.00	0.00	2
4.0	0.00	0.00	0.00	2
5.0	0.33	0.50	0.40	2
6.0	0.67	0.67	0.67	3
accuracy			0.40	15
macro avg	0.33	0.40	0.34	15
weighted avg	0.38	0.40	0.36	15

	y_Test	y_Pred
146	6.0	6.0
34	2.0	6.0
125	6.0	6.0
118	5.0	5.0
24	1.0	1.0
28	2.0	5.0
83	4.0	1.0
68	3.0	5.0
38	2.0	3.0
121	5.0	2.0
131	6.0	3.0
73	3.0	4.0
44	2.0	2.0
4	1.0	1.0
74	4.0	1.0

```
accuracy1 == 40.0
```

Learning Experience

Scrapping data from web

Got familiarized to manipulate unstructured data

By extensive processing on data content information is extracted

Data Pre-processing consumed most of our effort and time

Learned about some new libraries and modules

THANKYOU