Department of E& TC Engineering
Vishwakarma Institute of Technology, Pune

COURSE PROJECT REPORT
COMPUTER VISION LAB (EL 3221)

TITLE

Detection of Stray Animals on Roads

BATCH

__ A3___

GROUP MEMBERS

Branch	Div.	Roll No	GR No	Name
E&tc	A	51	11811310	Rohan Juneja
E&tc	A	54	11810122	Om Kakde
E&tc	A	57	11810120	Kartikey Dwivedi
E&tc	A	59	11810402	Vedant Khandekar
E&tc	A	65	11810994	Mandar Kulkarni

Batch guide
Prof. Dr. Medha Wyawahare

Title: Detection of Stray Animals on Roads

Introduction: Stray animals like dogs, cats can be found on almost everywhere. Our Project aims at detection of these animals so as to properly locate them and provide them with shelter and food and also make sure that they don't cause any traffic havoc. One serious problem that all the developed nations are facing today is death and injuries due to road accidents.. Due to road accidents, every year 1 out of 20,000 persons lose their life and 12 out of 70,000 individuals face serious injuries in India. India is also known for the maximum number of road accidents in the world The collision of an animal with the vehicle on the highway is one such big issue, which leads to such road accidents.

Background/ Literature survey:

465 cat and dog images were taken to train the model . The data was taken from Kaggle[2] and standford university[1] . The report commissioned by World Health Organization in its Global Status Study on Road Safety 2013, revealed that the leading cause of death for young people (15-29 age) globally is due to road traffic collisions. Even though various countries have initiated and taken steps to reduce road traffic collisions and accidents, the total number of crashes and traffic accidents remain as high as 1.24 million per year. Road traffic accidents and injuries are expected to rise by almost 65% by the end of 2020 .[1] . Techniques like SIFT , K means clustering and SVM are used . The algorithms were studied using [4][5][6] articles respectively . Opencv official website[7] was referred for understanding syntax and inbuilt functions.

Algorithm / Methodology :

- **SIFT**

The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images. It was published by David Lowe in 1999.[1] Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving. SIFT keypoints of objects are first extracted from a set of reference images[1] and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalised Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

- **KNN**

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. A cluster refers to a collection of data points aggregated together because of certain similarities. You'll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

- **SVM**

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well

Creating animals dataset

Applying sift feature descriptor

Classification using K means clustering

Applying SVM to create model

Calulating classification report

Testing on different images

Python Code :

In [32]:

```
import numpy as np
import pandas as pd
import os
import csv
import cv2
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import sklearn.metrics as metrics
from sklearn.metrics import accuracy_score
```

In [33]:

```
path=r"C:\Users\user\CVcourseproject\dataset\cat\c5.jpg"
a=cv2.imread(path)

resize=(512,512)
img=cv2.resize(a,resize)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
sift = cv2.xfeatures2d.SIFT_create()

keypoints, descriptors = sift.detectAndCompute(gray, None)
out=pd.DataFrame(descriptors)

kmeans = KMeans(n_clusters=5)


kmeans.fit(out.values)
a=kmeans.labels_
hist=np.histogram(kmeans.labels_,bins=[0,1,2,3,4,5])
```

In [34]:

```
print("shape:",out.shape)
print("Feature vector")
print(out)
```

```
shape: (845, 128)
Feature vector
       0     1     2     3     4     5     6     7      8      9  ...   118  \
0      1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   86.0    0.0  ...   0.0
1      1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  137.0    4.0  ...   0.0
2      1.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   71.0    7.0  ...   5.0
3      3.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0  160.0   59.0  ...   1.0
4     25.0   3.0   2.0   0.0   0.0   0.0   5.0  22.0   72.0    4.0  ...   0.0
..     ...   ...   ...   ...   ...   ...   ...   ...    ...    ...  ...   ...
840  130.0   2.0   0.0   0.0   0.0   0.0   1.0  57.0   96.0   19.0  ...   2.0
841   40.0  20.0   1.0   3.0  28.0   2.0   0.0   0.0  154.0   55.0  ...  45.0
842   49.0  27.0   6.0  26.0   8.0   0.0   0.0   1.0  125.0   71.0  ...   3.0
843   30.0  96.0   5.0   4.0   3.0   0.0   0.0   0.0  119.0  131.0  ...   0.0
844   39.0   9.0   2.0  18.0  33.0   1.0   0.0   6.0  124.0  106.0  ...   0.0

       119    120   121   122   123   124   125   126   127
0     22.0   63.0  12.0   0.0   2.0   5.0   8.0  17.0  55.0
1     14.0   46.0  36.0   2.0   1.0   1.0   1.0  11.0  43.0
2    108.0   69.0   9.0  20.0  80.0  33.0   8.0   1.0  32.0
3      6.0   77.0  23.0   1.0   1.0   0.0   2.0  20.0  34.0
4     23.0  135.0  32.0   0.0   0.0   0.0   0.0   0.0  30.0
..     ...    ...   ...   ...   ...   ...   ...   ...   ...
840    8.0   10.0   8.0   2.0  23.0  76.0  54.0  23.0   4.0
841   34.0    0.0   1.0   6.0   8.0   6.0  11.0  44.0  18.0
842   56.0    0.0   0.0   4.0   4.0  42.0  63.0  10.0   0.0
843   40.0    5.0  10.0  29.0  26.0   6.0  15.0   3.0   2.0
844    9.0   32.0  94.0   8.0  12.0  11.0  18.0   1.0   2.0

[845 rows x 128 columns]
```

In [35]:

```
print("Kmeans")
print(kmeans)
```

```
Kmeans
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [36]:

```
print("Centers")
print(kmeans.cluster_centers_)
```

```
Centers
[[ 24.054234   16.144566    8.608421   10.61445    10.35543     5.5783157
    6.2409616  13.493975   94.27115    28.584343    6.4337196   4.8855286
    4.542178    3.632557    7.319272   37.57228   118.548294   26.81325
    3.6325226   3.3433943   5.1687126   6.2048216  13.186745   60.680782
   49.63855    13.596387    3.6988044   6.831311   18.93978    15.620475
   19.873493   41.475876   38.066254   15.969904    8.385552    9.71686
   11.7409525   7.198823    8.909622   21.945784  120.45181    32.596382
   10.463856    8.102406    4.548235    5.126498   10.174691   39.921684
  135.07828    39.795174    6.156618    5.204829    5.63859     5.3132725
   11.01807    51.8976     65.3554     18.61446     8.271078   11.951831
   26.361422   15.843363   15.385545   38.30121    40.32529    15.692776
    7.180731    8.981927   12.0060425   8.054202   10.006031   25.042168
  115.403656   28.885538    9.4518      8.524085    4.620514    5.066225
   15.030116   45.024105  136.84332    45.126488   10.692774    6.8494225
    5.903631    3.9156427   6.8313117  47.95784    61.686718   33.638565
   18.686747   20.506023   26.331308   12.47591     7.8614593  22.174707
   25.650593   13.0301075   7.8614483   7.921684   10.108456    9.180728
   10.566261   19.93976    94.03619    32.891567    9.192769    5.6746902
    4.1084194   3.9698925   7.1445894  31.8253    115.00003    53.451813
   10.927722    6.042178    5.8614273   4.0542154   5.1987796  29.246988
   45.915634   36.29518    15.746991   17.548185   25.897614    9.753
    4.5060306  12.524095 ]
 [ 32.78235    19.07058    11.694109   12.75882    19.87647    17.705872
   10.276473   16.688234   48.958878   28.241177   12.464711   20.31176
   35.194115   19.852938   10.7        17.258816   90.27058    30.65294
    6.7588334  12.3647     34.311756   20.364708   11.888233   29.005877
   18.782352    7.7705865   5.7058926  26.629396   88.952835   32.82942
    8.62943     9.170557   41.41176    22.976467   10.929415   14.111765
   23.31765    21.494114   16.45882    21.517647   62.11766    25.435284
   15.541182   32.676495   54.676514   32.899994   13.176473   14.935309
  124.27055    39.02941     8.441168   15.5647135  41.964703   17.982367
    7.0470567  25.382359   23.505877    7.811779    4.7999935  33.929436
  111.294205   39.688236    6.4117737   7.1940975  44.258804   27.141169
   13.564708   15.258822   22.058823   18.347061   14.788237   22.3
   58.947113   15.517651    8.247049   26.064697   57.588257   42.311733
   21.20588    24.92354   122.923485   27.335276    6.023533   18.570585
   43.723515   16.029406   10.44118    36.617645   23.611746    8.864721
    6.4412003  41.35882   109.60599    34.47648     4.6000185   6.3999834
   36.4        24.464699    9.47059    12.605882   19.970589   12.21765
   10.158825   17.494122   47.341225   22.129421    9.635293   16.723526
   39.329395   21.98236    13.788236   23.205904   83.57062    29.311771
    9.264706   21.07059    38.076477   14.929411   10.358828   26.48824
   20.376465    9.576478    5.9117765  31.847055   84.06465    26.617634
    6.3000045   6.558831 ]
 [ 23.066677   21.585712   18.309519   22.004755   27.83334    20.833332
   17.05238    18.314291   32.719013   31.104767   24.39522    27.266668
   29.07622    21.83811    19.228586   21.023802   57.93808    33.76192
   19.061909   16.80476    17.114275   19.704763   24.071417   39.13809
   40.22857    18.461906   14.157137   13.233325   15.876196   21.314287
   26.119083   36.714252   28.809513   29.176186   25.119055   31.752365
   30.057146   25.295242   21.79524    21.161905   36.35711    34.090477
   39.490498   53.266624   50.538136   30.514278   15.895234   14.428616
  117.03806    54.16193    21.714272   15.195248   10.800035   11.714301
   16.428568   43.5381     53.004757   20.890469   15.271422   20.504776
   20.65234    28.138086   31.680973   39.042847   28.94288    28.3
   25.304745   31.538094   32.92858    24.719048   19.18572    21.10476
   44.85245    18.800003   21.49049    38.819088   48.309555   41.266624
   29.495241   30.90001   116.95235    46.885696   19.585718   14.428569
   10.438118   12.952388   18.076176   52.74285    50.842834   35.58573
   32.980976   32.88095    18.985737   17.495243   17.285734   27.076181
```

```
 20.24282    20.647612   19.895214   23.309513   29.261913   21.857147
 18.452385   18.947622   36.233315   26.495241   23.928577   26.63809
 26.96189    22.28572    23.100006   31.333332   62.60477    44.719055
 26.947617   22.495235   15.2857275  12.233332   16.333332   36.14763
 38.023785   35.89049    31.471401   24.766666   13.352339    9.361895
 14.400009   24.733343 ]
[ 70.80268   38.129242   10.544211    5.8435307   7.2449055   3.4149723
  3.6394453 17.006802  117.57822    45.10882     8.945576    5.068016
  8.65308     8.197275    6.7278876  29.809517   46.965973   20.129263
 12.972788   21.54421    43.176914   25.81633    11.591835   16.34016
 20.707481   14.517009   13.081634   21.020407   36.353737   23.761906
 11.578232   12.047594   88.50343    31.578236    9.272116    7.469371
  9.632637    6.517025    5.523797   21.836735  139.46945    39.09523
  6.11563     6.5510178  10.380964    8.299312    5.7823067  34.95918
 61.06798    18.27213    10.129254   27.714296   66.6667     35.489815
 13.428571   17.700686   24.619057   19.809525   14.557822   21.721088
 44.54422    23.653063   14.068028   15.986408   88.163315   22.10204
  3.6530704   5.4897947   9.35376     8.374136    9.632659   30.38095
140.61911    37.544228    4.482991    7.1972656  10.714312    7.4421673
  4.3197117  36.789124   61.244843   21.884335   11.068031   32.70748
 68.33339    28.442165   10.068029   15.911554   23.666653   19.72788
 12.01362    23.999998   42.891155   24.952385   16.564625   14.877561
 70.63261    21.210875    3.2720952   4.034004    8.857154    7.496604
  8.482998   30.496586  121.99998    37.299328    6.1768713   8.448981
  9.380951    4.5578337   4.748311   41.993202   43.78914    21.639462
 14.027209   26.870749   47.006783   19.48299     7.9115605  17.44899
 19.17687    14.761893   13.863944   24.408161   35.13605    19.000002
  8.829932   13.204081 ]
[ 22.842117   24.059212   20.999989   30.072365   41.36185    18.592102
 11.598686   13.519734   66.55263    38.500008   19.940786   18.519735
 20.230263   15.078946   13.861841   32.10526    39.717125   27.26974
 21.63158    24.302639   27.815783   26.940794   20.06579    27.77631
 21.039473   22.335533   22.39475    28.631567   30.750004   21.092106
 15.986842   15.4605     32.802643   23.8421     21.973686   42.23682
 59.81581    30.697384    9.210522   10.210526  120.80263    37.131577
 12.493432   13.717129   17.54604    14.124993   15.013157   49.434242
 43.888115   20.072361   18.618418   37.190804   54.276352   46.0066
 25.592102   27.322374   25.309223   25.19079    27.217098   35.493443
 37.881577   24.743422   18.236845   18.559223   33.42761    13.789471
  9.532902   29.394735   59.151264   43.888145   21.875004   18.046051
121.250046   55.092064   19.960518   14.440784   16.072378   15.03948
 11.67104    32.940796   36.736835   32.875      37.78946    52.032936
 52.73025    32.76971    13.361843   14.0328865  27.624985   23.993423
 21.513157   30.499998   35.217113   30.131586   20.809227   23.539484
 22.256557   15.513151    9.625001   17.059217   38.934242   30.717096
 21.539476   21.269741   53.927628   33.921062   17.434216   14.44737
 19.92763    22.671059   21.111845   32.06579    33.86845    30.743427
 24.092125   24.440784   29.46052    24.559208   19.256577   21.710518
 22.90132    20.355257   17.789476   24.072367   26.578945   23.598682
 18.776316   22.046059 ]]
```

In [37]:

```
print("labels")
print(kmeans.labels_)
a=kmeans.labels_
print("Labels shapel / predicted values")
print(a.shape)
```

```
labels
[0 0 0 0 0 0 0 2 1 0 0 2 2 2 0 0 2 0 0 0 0 0 0 0 3 0 3 0 3 0 0 4 2 4 3 1 0
 0 0 3 0 0 0 3 3 3 3 0 0 0 3 3 0 0 1 3 1 0 3 3 3 1 2 0 2 3 3 3 0 2 3 3 3 2
 3 2 3 2 1 4 1 1 4 1 0 2 1 0 3 2 4 1 3 3 2 1 1 3 0 4 1 2 3 1 3 1 4 2 2 1 2
 0 2 1 2 0 1 0 3 2 2 2 3 3 2 2 1 2 1 0 2 4 1 0 4 4 4 1 2 3 3 0 1 2 1 1 1 0
 1 1 3 4 2 4 3 2 2 3 1 0 3 1 1 3 4 0 3 2 2 0 2 3 1 3 4 1 4 1 0 1 2 0 4 1 3
 4 4 4 4 2 2 2 4 2 1 2 0 2 4 1 4 4 3 1 0 1 1 1 0 1 0 0 2 4 1 2 1 1 3 4 4 3
 4 2 2 2 0 0 4 3 4 2 1 4 3 4 1 4 4 1 1 1 2 4 0 2 2 2 1 1 2 2 0 1 3 0 4 2 2
 0 0 1 4 2 4 2 2 1 2 0 0 1 2 2 1 1 2 0 0 0 1 2 0 4 4 2 0 4 4 0 0 3 3 4 2 4
 4 3 4 4 2 3 3 2 2 0 1 0 3 4 2 4 0 4 1 0 1 1 1 2 2 2 2 1 0 0 2 2 2 2 2 4 3
 3 3 4 1 2 2 0 0 4 2 1 2 4 2 3 4 0 4 4 2 1 2 2 4 4 1 1 1 2 1 2 0 2 0 2 2 1
 4 3 4 1 1 0 4 4 3 4 2 0 4 4 4 3 2 4 4 2 1 2 2 2 2 0 2 2 1 4 1 1 4 4 4 4 3
 2 3 3 1 4 4 0 4 1 1 3 4 4 4 3 0 3 4 1 4 4 4 3 2 0 0 4 1 1 1 3 2 2 0 0 1 2
 3 2 0 0 3 4 0 3 3 3 3 2 2 3 2 1 1 3 3 2 1 0 1 1 4 3 0 3 4 2 4 0 0 4 3 3 0
 0 3 4 0 1 2 2 2 2 0 4 0 3 3 3 2 0 1 2 2 2 2 2 2 4 2 0 4 4 1 2 4 2 2 2 2 4
 3 4 2 3 2 2 4 4 2 2 1 1 2 0 1 1 4 3 1 3 2 4 2 2 0 2 4 2 2 2 0 1 4 2 2 4 0
 1 2 2 2 2 3 1 4 2 1 1 4 1 2 0 0 1 2 3 0 4 1 0 4 4 3 4 4 4 4 1 2 2 0 2 2 2
 3 3 3 2 2 3 4 4 4 0 0 0 1 4 2 1 2 4 4 4 4 4 2 4 4 4 0 1 0 3 0 1 2 1 1 1 1
 4 0 3 2 2 4 0 2 2 0 0 2 3 2 1 1 3 4 4 3 2 2 3 4 2 4 1 1 1 4 2 2 1 1 2 2 2
 0 3 0 3 1 2 1 2 4 2 2 2 0 1 1 1 0 4 1 1 2 2 1 2 3 4 4 1 3 4 1 0 3 1 2 2 1
 2 1 3 2 3 4 4 0 1 2 2 2 2 2 3 2 4 3 3 2 4 1 4 2 2 2 0 1 1 0 2 0 4 2 0 1 2
 3 4 1 1 1 0 1 0 1 1 0 3 4 4 3 2 1 3 0 3 3 1 4 1 4 1 2 3 1 1 1 3 3 4 1 2 1
 3 1 3 1 3 1 0 1 3 1 3 3 0 0 3 3 3 0 3 0 3 0 0 0 2 2 0 2 0 0 3 0 0 0 1 1 4
 2 3 0 3 0 0 1 3 3 3 0 1 1 3 0 2 0 3 0 0 0 1 0 0 0 3 3 3 3 3 0]
Labels shapel / predicted values
(845,)
```

In [38]:

```
print("histogram[0,1,2,3,4]")
print(hist)
```

```
histogram[0,1,2,3,4]
(array([166, 170, 210, 147, 152], dtype=int64), array([0, 1, 2, 3, 4, 5]))
```

In [39]:

```
folder1=r"C:\Users\user\CVcourseproject\dataset\cat"

for filename in os.listdir(folder1):
    path=os.path.join(folder1,filename)
    a=cv2.imread(path)
    resize=(512,512)
    img=cv2.resize(a,resize)#resize image
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()#initialise sift detector
    keypoints, descriptors = sift.detectAndCompute(gray, None)#collection of detectors
 from image
    out=pd.DataFrame(descriptors)
    csv_data=out.to_csv('cat.csv', mode='a', header=False , index = False)
```

In [40]:

```
folder2=r"C:\Users\user\CVcourseproject\dataset\dog"

for filename in os.listdir(folder2):
    path=os.path.join(folder2,filename)
    a=cv2.imread(path)
    resize=(512,512)
    img=cv2.resize(a,resize)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(gray, None)
    out=pd.DataFrame(descriptors)
    csv_data=out.to_csv('dog.csv', mode='a', header=False , index = False)
```

In [41]:

```
data= pd.read_csv(r'C:\Users\user\CVcourseproject\cat.csv')
data
print(data)
```

```
          0.0  0.0.1  0.0.2  0.0.3    1.0  1.0.1  0.0.4  0.0.5    5.0  
4.0  \
0        44.0    2.0    1.0    1.0    0.0    0.0    0.0    3.0   22.0   
1.0   
1       128.0   74.0   17.0    7.0   23.0   12.0    1.0    7.0   42.0   4
7.0   
2         0.0    0.0    0.0    0.0    8.0   74.0  106.0    1.0  107.0   1
7.0   
3        23.0    4.0    0.0   14.0  101.0   29.0    2.0    7.0  120.0   6
9.0   
4         2.0   13.0   16.0   10.0    4.0   28.0   64.0   10.0    1.0   
4.0   
...       ...    ...    ...    ...    ...    ...    ...    ...    ...   
...   
108721  130.0    2.0    0.0    0.0    0.0    0.0    1.0   57.0   96.0   1
9.0   
108722   40.0   20.0    1.0    3.0   28.0    2.0    0.0    0.0  154.0   5
5.0   
108723   49.0   27.0    6.0   26.0    8.0    0.0    0.0    1.0  125.0   7
1.0   
108724   30.0   96.0    5.0    4.0    3.0    0.0    0.0    0.0  119.0  13
1.0   
108725   39.0    9.0    2.0   18.0   33.0    1.0    0.0    6.0  124.0  10
6.0   

        ...  34.0  52.0.1  18.0.2  47.0  42.0  64.0  36.0  4.0.4  1.0.12  
\
0       ...   6.0    94.0    78.0  27.0   3.0   4.0  14.0    3.0     2.0   
1       ...   0.0     1.0     0.0   0.0   0.0   0.0   1.0    0.0     0.0   
2       ...  16.0    57.0     1.0   0.0   0.0   0.0   1.0    3.0     0.0   
3       ...   1.0     1.0     6.0   6.0   9.0  17.0   1.0    0.0     0.0   
4       ...  23.0   104.0    10.0  28.0  52.0  15.0   0.0    1.0    81.0   
...     ...   ...     ...     ...   ...   ...   ...   ...    ...     ...   
108721  ...   2.0     8.0    10.0   8.0   2.0  23.0  76.0   54.0    23.0   
108722  ...  45.0    34.0     0.0   1.0   6.0   8.0   6.0   11.0    44.0   
108723  ...   3.0    56.0     0.0   0.0   4.0   4.0  42.0   63.0    10.0   
108724  ...   0.0    40.0     5.0  10.0  29.0  26.0   6.0   15.0     3.0   
108725  ...   0.0     9.0    32.0  94.0   8.0  12.0  11.0   18.0     1.0   

        4.0.5  
0        12.0  
1         0.0  
2         3.0  
3         0.0  
4        83.0  
...       ...  
108721    4.0  
108722   18.0  
108723    0.0  
108724    2.0  
108725    2.0  

[108726 rows x 128 columns]
```

In [42]:

```
data2= pd.read_csv(r'C:\Users\user\CVcourseproject\dog.csv')
print(data2)
```

```
         12.0   9.0  20.0  25.0  29.0   7.0  18.0  20.0.1   15.0   22.0
...  \
0        21.0   1.0  11.0  21.0  12.0  23.0  17.0    38.0   58.0   18.0
...
1         0.0   0.0   0.0   0.0   0.0   0.0   0.0     0.0    6.0    1.0
...
2         0.0   0.0   0.0   0.0   0.0   0.0   0.0     0.0    6.0    0.0
...
3        32.0  66.0  18.0   4.0   5.0  19.0   8.0     9.0   91.0  112.0
...
4        27.0   0.0   0.0   3.0  82.0  76.0   6.0    11.0  146.0    2.0
...
...       ...   ...   ...   ...   ...   ...   ...     ...    ...    ...
...
112011   18.0   0.0   0.0   0.0   1.0   1.0   9.0   121.0  132.0    5.0
...
112012  121.0  33.0   0.0   0.0   0.0   0.0   0.0    16.0  122.0   62.0
...
112013  144.0  24.0   2.0   0.0   0.0   1.0   3.0    10.0   42.0   29.0
...
112014  167.0  10.0   1.0   0.0   0.0   0.0   0.0    32.0  167.0   20.0
...
112015    5.0   2.0   4.0  39.0  57.0  13.0  14.0    23.0   67.0   11.0
...

        2.0.2  3.0.5  0.0.14  0.0.15  0.0.16  0.0.17  0.0.18  0.0.19  0.0.
20  \
0         0.0    0.0     0.0     0.0     0.0     0.0     0.0     0.0
0.0
1        16.0   26.0   119.0    30.0    23.0    11.0     3.0     4.0
6.0
2         8.0   34.0    30.0     3.0     2.0    11.0     4.0    18.0     5
8.0
3         8.0   28.0     0.0     0.0     0.0     0.0     1.0     1.0
0.0
4         0.0    1.0     0.0     0.0     0.0     0.0     0.0     0.0
0.0
...       ...    ...     ...     ...     ...     ...     ...     ...
...
112011    0.0    1.0     3.0     2.0     0.0     0.0     0.0     0.0
0.0
112012    0.0   58.0    32.0     5.0     3.0     7.0     8.0     1.0
0.0
112013    0.0    5.0     3.0     3.0     0.0     0.0     0.0     0.0
0.0
112014    0.0    3.0     0.0     0.0     0.0     0.0     0.0     0.0
0.0
112015    0.0    0.0     0.0     0.0     0.0     0.0     0.0     0.0
0.0

        0.0.21
0          0.0
1         33.0
2         50.0
3          0.0
4          0.0
...        ...
112011     0.0
112012     2.0
112013     0.0
112014     0.0
```

In [43]:

```
kmeans1 = KMeans(n_clusters=5)
kmeans1.fit(data)
```

Out[43]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [44]:

```
kmeans2 = KMeans(n_clusters=5)
kmeans2.fit(data2)
```

Out[44]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [45]:

```
hist1=np.histogram(kmeans1.labels_,bins=[0,1,2,3,4,5])
hist2=np.histogram(kmeans2.labels_,bins=[0,1,2,3,4,5])

print('histogram of Cats')
print(hist1)

print('histogram of Dogs')
print(hist2)
```

```
histogram of Cats
(array([25713, 17078, 26723, 19000, 20212], dtype=int64), array([0, 1, 2,
3, 4, 5]))
histogram of Dogs
(array([20995, 18629, 19282, 25233, 27877], dtype=int64), array([0, 1, 2,
3, 4, 5]))
```

In [46]:

```
folder1=r"C:\Users\user\CVcourseproject\dataset\cat"
i=0
data=[]

for filename in os.listdir(folder1):
    path=os.path.join(folder1,filename)
    a=cv2.imread(path)
    resize=(512,512)
    img=cv2.resize(a,resize)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(gray, None)
    out=pd.DataFrame(descriptors)
    #predict values of feature vector with pretrained kmeans
    array_double = np.array(out, dtype=np.double)
    a=kmeans1.predict(array_double)
    hist=np.histogram(a,bins=[0,1,2,3,4,5])
    #append the dataframe into the array in append mode, the array will only have 5 val
ues which will store the values in a row
    data.append(hist[0])

#convert Array to Dataframe and append to the list
Output = pd.DataFrame(data)
#add row class
Output["Class"] = i
csv_data=Output.to_csv('CatFinal.csv', mode='a', header=False , index = False)
```

In [47]:

```
folder2=r"C:\Users\user\CVcourseproject\dataset\dog"
i=1
data=[]

for filename in os.listdir(folder2):
    path=os.path.join(folder2,filename)
    a=cv2.imread(path)
    resize=(512,512)
    img=cv2.resize(a,resize)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(gray, None)
    out=pd.DataFrame(descriptors)
    array_double = np.array(out, dtype=np.double)
    a=kmeans2.predict(array_double)
    hist=np.histogram(a,bins=[0,1,2,3,4,5])
    data.append(hist[0])

Output = pd.DataFrame(data)
Output["Class"] = i
csv_data=Output.to_csv('DogFinal.csv', mode='a', header=False , index = False)
```

In [48]:

```
data1= pd.read_csv(r'C:\Users\user\CVcourseproject\CatFinal.csv' , header=None)
print(data1)

data2= pd.read_csv(r'C:\Users\user\CVcourseproject\DogFinal.csv' , header=None)
print(data2)
```

```
      0    1    2    3    4  5
0   206  304  229  340  341  0
1   218  159  229  205  201  0
2   595  404  487  548  586  0
3   556  364  602  499  390  0
4   199  150  190  176  287  0
..  ...  ...  ...  ...  ... ..
68  141  168  161  231  343  0
69  116  118  126  163  161  0
70  195  215  248  208  192  0
71  634  321  456  274  228  0
72  109  163  182  171  220  0

[73 rows x 6 columns]
      0    1    2    3    4  5
0   310  409  583  696  898  1
1    62   47   33   39   37  1
2   290  154  141  160  205  1
3   210   96   83   84   95  1
4   516  321  346  313  291  1
..  ...  ...  ...  ...  ... ..
60  340  279  214  237  243  1
61  456  271  233  250  276  1
62  266   62  111   70   92  1
63  477  142  173  108  113  1
64  199  238  278  307  326  1

[65 rows x 6 columns]
```

In [49]:

```
A=data1.append(data2)

csv_data=A.to_csv('Final.csv', mode='a', header=False , index = False)
```

In [50]:

```
data= pd.read_csv(r'C:\Users\user\CVcourseproject\Final.csv')

data.columns=['1','2','3','4','5','Class']

print(data)

print(data[0:5])
```

```
       1    2    3    4    5  Class
0    218  159  229  205  201      0
1    595  404  487  548  586      0
2    556  364  602  499  390      0
3    199  150  190  176  287      0
4     22   24   10   15   69      0
..   ...  ...  ...  ...  ...    ...
132  340  279  214  237  243      1
133  456  271  233  250  276      1
134  266   62  111   70   92      1
135  477  142  173  108  113      1
136  199  238  278  307  326      1

[137 rows x 6 columns]
     1    2    3    4    5  Class
0  218  159  229  205  201      0
1  595  404  487  548  586      0
2  556  364  602  499  390      0
3  199  150  190  176  287      0
4   22   24   10   15   69      0
```

In [51]:

```
x = data.iloc[:, 0:5]
print("X values")
print(x)

y = data['Class']
print("Y values")
print(y)
```

```
X values
       1    2    3    4    5
0    218  159  229  205  201
1    595  404  487  548  586
2    556  364  602  499  390
3    199  150  190  176  287
4     22   24   10   15   69
..   ...  ...  ...  ...  ...
132  340  279  214  237  243
133  456  271  233  250  276
134  266   62  111   70   92
135  477  142  173  108  113
136  199  238  278  307  326

[137 rows x 5 columns]
Y values
0      0
1      0
2      0
3      0
4      0
      ..
132    1
133    1
134    1
135    1
136    1
Name: Class, Length: 137, dtype: int64
```

In [52]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.20, random_state=0
)
```

In [53]:

```
from sklearn import svm

model1 = svm.SVC(kernel='linear')


model1.fit(x_train, y_train)

y_pred1 = model1.predict(x_test)
print("SVM Results")
print(classification_report(y_test, y_pred1))
print("SVM: ",accuracy_score(y_test, y_pred1))
```

```
SVM Results
              precision    recall  f1-score   support

           0       0.93      0.82      0.87        17
           1       0.77      0.91      0.83        11

    accuracy                           0.86        28
   macro avg       0.85      0.87      0.85        28
weighted avg       0.87      0.86      0.86        28

SVM:  0.8571428571428571
```

In [57]:

```
path=r"C:\Users\user\CVcourseproject\dataset\dog\n02106662_104.jpg"
from IPython.display import display
from PIL import Image
im = Image.open(path)
display(im)
data=[]

a=cv2.imread(path)
resize=(512,512)

img=cv2.resize(a,resize)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

sift = cv2.xfeatures2d.SIFT_create()
keypoints, descriptors = sift.detectAndCompute(gray, None)

out=pd.DataFrame(descriptors)

kmeans = KMeans(n_clusters=5)

kmeans.fit(out.values)

hist=np.histogram(kmeans.labels_,bins=[0,1,2,3,4,5])

#append the dataframe into the array in append mode, the array will only have 5 values
 which will store the values in a row
data.append(hist[0])

print("predicted kmeans:",data)
Output = pd.DataFrame(data)

print("Dataframe:")
print(Output)
```

```
predicted kmeans: [array([363, 170, 231, 205, 317], dtype=int64)]
Dataframe:
     0    1    2    3    4
0  363  170  231  205  317
```

In [98]:

```
#assigning the columns 1 to 128 of new image as training variables
x = Output.iloc[:, 0:5]

#prediction
y_pred1 = model1.predict(x)

#prints the prediction of the class
print(y_pred1)
```

```
[1]
```

In [99]:

```
#condition to check if its cat or Dog
if y_pred1==0:
    print("Cat")
elif y_pred1==1:
    print("Dog")
```

```
Dog
```

Applications

- Stray animals often falls prey to fast moving vehicles which is danger for animals as well as the rider so detection of these animals is important
- Lost Animals are really difficult to find . If the picture of the animal is known then detecting similar looking pet in the same area will make it easier to find the pet
- Stray Animals especially dogs may have serious diseases like rabies which can be fatal for humans and other animals. Locating such animals is necessary for everyone's safety. This can be done using the proposed algorithm.

Conclusion:

- After cropping the animals images we can apply feature descriptors like sift to extract the features . Further by the use of unsupervised algorithm like K means clustering we can label our target dataset . The labelled dataset can now be processed using a supervised algorithm like SVM . The dataset contained 465 cat images and 465 dog images . After training the model the accuracy was found to be 80 +- 5 %.

References:

1. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7792584
2. http://vision.stanford.edu/aditya86/ImageNetDogs/main.html
3. https://www.kaggle.com/crawford/cat-dataset
4. https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40
5. https://en.wikipedia.org/wiki/K-means_clustering
6. https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
7. https://opencv.org/