```
!pip install turicreate
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
import turicreate
```

Image retrieval using deep features

```
image_train = turicreate.SFrame('/content/gdrive/My Drive/Turicreate/Week 6/image_train_data/')
```

Computing summary statistics of the data

```
image_train['label'].summary()
```

```
+------------------+-------+----------+
|       item       | value | is exact |
+------------------+-------+----------+
|      Length      |  2005 |   Yes    |
| # Missing Values |   0   |   Yes    |
| # unique values  |   4   |    No    |
+------------------+-------+----------+

Most frequent items:
+------------+-------+
|   value    | count |
+------------+-------+
|    cat     |  509  |
|    dog     |  509  |
| automobile |  509  |
|    bird    |  478  |
+------------+-------+
```

Creating category-specific image retrieval models

```
dog_train_image=image_train.filter_by('dog','label')
```

```
dog_model = turicreate.nearest_neighbors.create(dog_train_image,features=['deep_features'],
                                                label='id')
```

```
Starting brute force nearest neighbors model training.
Validating distance components
```

```
model={}
for key in ('dog','cat','automobile','bird'):
    image=image_train.filter_by(key,'label')
    model[key] = turicreate.nearest_neighbors.create(image, features=['deep_features'],label='id')
```

```
def get_images_from_ids(query_result):
    return image_data.filter_by(query_result['reference_label'],'id')
```

```
image_test = turicreate.SFrame('/content/gdrive/My Drive/Turicreate/Week 6/image_test_data/')
cat1=image_test[0:1]

query=model['cat'].query(cat1)
get_images_from_ids(query[0])['image'].explore()
```

```
Starting pairwise querying.
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 0            | 1       | 0.196464    | 21.147ms     |
| Done         |         | 100         | 81.114ms     |
+--------------+---------+-------------+--------------+
```

	SArray
0	

```
query=model['dog'].query(cat1)
get_images_from_ids(query[0])['image'].explore()
```

```
Starting pairwise querying.
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 0            | 1       | 0.196464    | 13.465ms     |
| Done         |         | 100         | 87.124ms     |
+--------------+---------+-------------+--------------+
```

	SArray
0	

A simple example of nearest-neighbors classification

```
nearest_cats=model['cat'].query(cat1)
nearest_cats[0:5]['distance'].mean()
```

```
Starting pairwise querying.
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 0            | 1       | 0.196464    | 19.801ms     |
| Done         |         | 100         | 83.254ms     |
+--------------+---------+-------------+--------------+
36.15573070978294
```

```
nearest_dogs=model['dog'].query(cat1)
nearest_dogs[0:5]['distance'].mean()
```

```
Starting pairwise querying.
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 0            | 1       | 0.196464    | 22.972ms     |
| Done         |         | 100         | 81.816ms     |
+--------------+---------+-------------+--------------+
37.77071136184157
```

Computing nearest neighbors accuracy using SFrame operations

```
test_image={}
for key in ('dog','cat','automobile','bird'):
    test_image[key]=image_test.filter_by(key,'label')


dog_cat_neighbors = model['cat'].query(test_image['dog'], k=1)

dog_neighbors={}
for key in ('dog','cat','automobile','bird'):
    dog_neighbors[key] = model[key].query(test_image['dog'], k=1)
```

```
Starting blockwise querying.
max rows per data block: 4348
number of reference data blocks: 2
number of query data blocks: 1
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 1000         | 255000  | 50.0982     | 660.749ms    |
| Done         | 509000  | 100         | 696.836ms    |
+--------------+---------+-------------+--------------+
Starting blockwise querying.
max rows per data block: 4348
number of reference data blocks: 2
number of query data blocks: 1
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 1000         | 254000  | 49.9018     | 656.499ms    |
| Done         | 509000  | 100         | 677.334ms    |
+--------------+---------+-------------+--------------+
Starting blockwise querying.
max rows per data block: 4348
number of reference data blocks: 2
number of query data blocks: 1
+--------------+---------+-------------+--------------+
| Query points | # Pairs | % Complete. | Elapsed Time |
+--------------+---------+-------------+--------------+
| 1000         | 255000  | 50.0982     | 681.059ms    |
| Done         | 509000  | 100         | 691.814ms    |
```

```
dog_distances=turicreate.SFrame({
    'dog-dog':dog_neighbors['dog']['distance'],
    'dog-cat':dog_neighbors['cat']['distance'],
    'dog-automobile':dog_neighbors['automobile']['distance'],
    'dog-bird':dog_neighbors['bird']['distance'],
})
dog_distances.head()
```

dog-automobile	dog-bird	dog-cat	dog-dog
41.95797614571203	41.75386473035126	36.419607706754384	33.47735903726335
46.00213318067788	41.3382958924861	38.83532688735542	32.84584956840554
42.946229069238804	38.615759085289056	36.97634108541546	35.03970731890584
41.68660600484793	37.08922699538214	34.575007291446106	33.90103276968193
39.22696649347584	38.27228869398105	34.77882479101661	37.484925090925636
40.58451176980721	39.146208923590486	35.11715782924591	34.94516534398124
45.10673529610854	40.523040105962316	40.60958309132649	39.095727834463545
41.32211409739762	38.19479183926956	39.90368673062214	37.76961310322034
41.82446549950164	40.156713166131446	38.067470016821176	35.10891446032838
45.497692940110376	45.55979626027668	42.72587329506032	43.242283258453455

[10 rows x 4 columns]

Computing the number of correct predictions using 1-nearest neighbors for the dog class

```
def is_dog_correct(row):
    if row['dog-dog']==min(row.values()):
```

```
        return 1
    else:
        return 0


row=dog_distances[1]
row
```

```
{'dog-automobile': 46.00213318067788,
 'dog-bird': 41.3382958924861,
 'dog-cat': 38.83532688735542,
 'dog-dog': 32.84584956840554}
```

```
min(row.values())
```

```
32.84584956840554
```

```
correct=dog_distances.apply(is_dog_correct).sum()
total=len(dog_distances)
print(float(correct)/total)
```

```
0.678
```