Saimun Islam
CIS 4130
Saimun.Islam@baruchmail.cuny.edu

Milestone 1 Proposal:

130k Images (512x512) - Universal Image Embeddings

https://www.kaggle.com/datasets/rhtsingh/130k-images-512x512-universal-image-embeddings

Data Source

1. Apparel - Deep Fashion Dataset
2. Artwork - Google Scrapped
3. Cars - Stanford Cars Dataset
4. Dishes - Google Scrapped
5. Furniture - Google Scrapped
6. Illustrations - Google Scrapped
7. Landmark - Google Landmark Dataset
8. Meme - Google Scrapped
9. Packaged - Holosecta, Grozi 3.2k, Freiburg Groceries, SKU110K
10. Storefronts - Google Scrapped
11. Toys - Google Scrapped

I want to be able to predict how the future car would look like based on the dataset

I want to be able to predict how future apparel would look like based on the data set

I want to be able to predict meme and rate them based on their content from 1-10.

I want to be able to see how the landmark would look like with low and high resolution.

Milestone2 Data Acquisition:

1. **To begin with I created a bucket to Amazon S3 with these code:**

   ```
   aws s3api create-bucket --bucket project-data- --region us-east-2 --create-bucket-configuration LocationConstraint=us-east-2
   ```

2. Install the Kaggle Command Line Interface (CLI) for Python 3.

   ```
   $ pip3 install Kaggle
   ```

3. Create a new API Token in Kaggle. Go to Kaggle.com and log in. Go to the Account page.

   A file kaggle.json will be downloaded.
   Open up this file in Notepad or TextEdit and copy the contents to the clipboard.

4. EC2 instance:
   a. Make a directory for Kaggle: $ mkdir .kaggle
   The leading "." is important.
   b. Create a new kaggle.json file using nano editor: $ nano .kaggle/kaggle.json
   c. Paste in the text from your kaggle.json file that you downloaded from Kaggle.com. It will user name:
   d. {"username":"saimunislam1","key":"d45e468509b8f1908ead5e4b0465b4de"}
   e. Save the kaggle.json file and exit nano.

f. Secure the file: `$ chmod 600 .kaggle/kaggle.json`
g. Try out a kaggle cli command: `$ kaggle datasets list`

Edit the kaggle_api_extended.py and make two changes (shown below) to accommodate writing the file to standard output.

h. Used the nano text editor to edit the kaggle_api_extended.py file:

```
$ nano  ~/.local/lib/python3.7/site-packages/kaggle/api/kaggle_api_extended.py
```

i. Use the "Go To" key in Nano: ^_ to go to line 1582 in the file.
j. Change line 1582 from:

```
    if not os.path.exists(outpath):
  to
    if not os.path.exists(outpath) and outpath != "-":
```

k. Change line 1594 From:

```
with open(outfile, 'wb') as out:
        to

with open(outfile, 'wb') if outpath != "-" else os.fdopen(sys.stdout.fileno(), 'wb', closefd=False) as out:
```

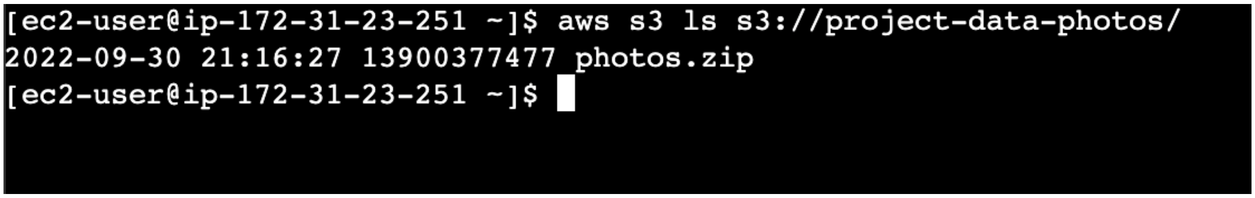
l. Save the file and exit Nano

5. Named the data set

```
kaggle datasets download -d rhtsingh/130k-images-512x512-universal-image-embeddings -p -  | aws s3 cp - s3://project-data-photos/photos.zip
```

6. S3 bucket to see if the file was downloaded

```
aws s3 ls s3://project-data-photos/
```



```
[ec2-user@ip-172-31-23-251 ~]$ aws s3 ls s3://project-data-photos/
2022-09-30 21:16:27 13900377477 photos.zip
[ec2-user@ip-172-31-23-251 ~]$ 
```

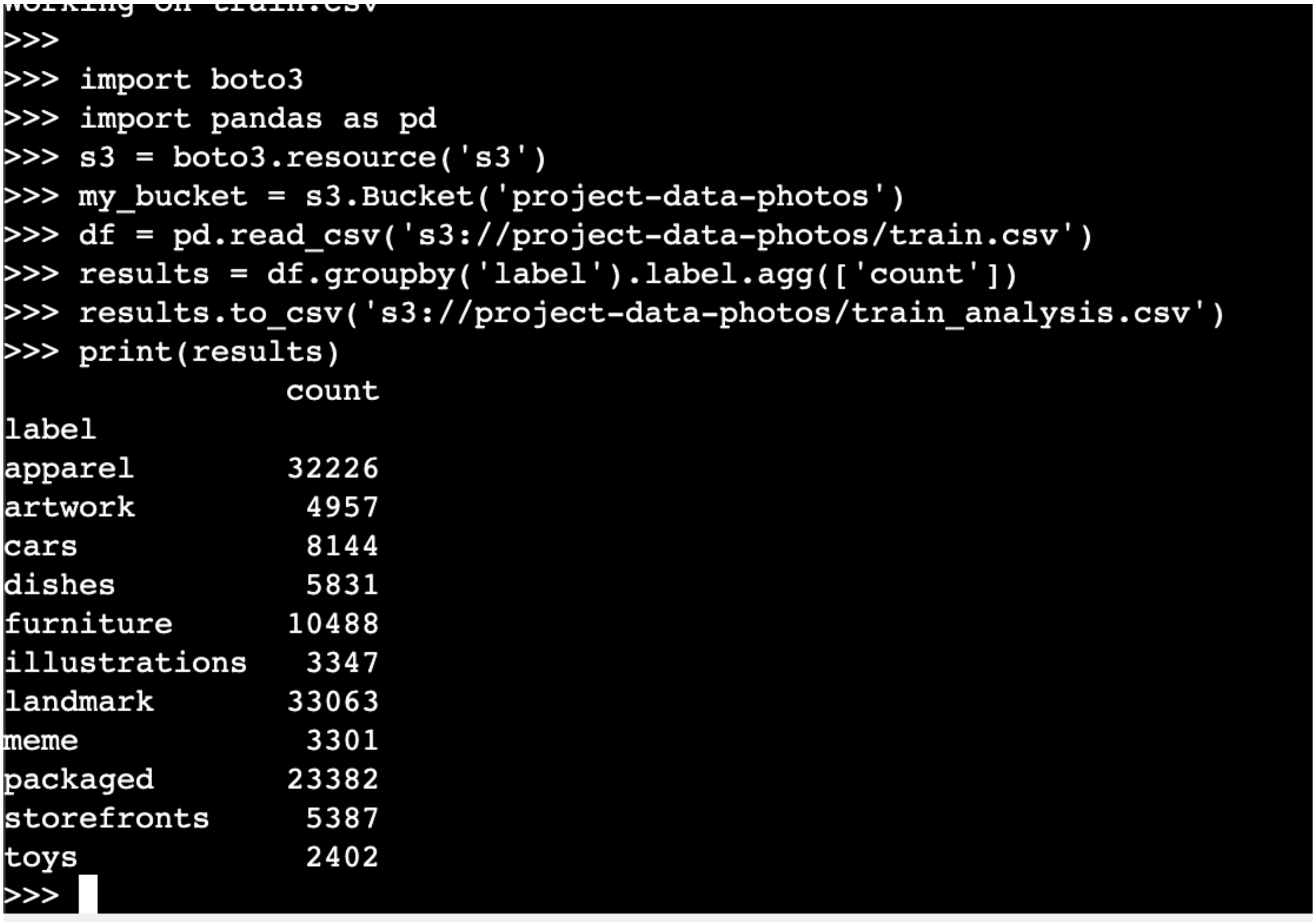
Milestone 3 Descriptive Statistics:

Step 1: I used this code to download all the data to S3

```
import zipfile
import boto3
from io import BytesIO
bucket="project-data-photos"
zipfile_to_unzip="photos.zip"
s3_client = boto3.client('s3', use_ssl=False)
s3_resource = boto3.resource('s3')
# Create a zip object that represents the zip file
zip_obj = s3_resource.Object(bucket_name=bucket,
key=zipfile_to_unzip)
buffer = BytesIO(zip_obj.get()["Body"].read())
z = zipfile.ZipFile(buffer)
# Loop through all of the files contained in the Zip archive
for filename in z.namelist():
     print('Working on ' + filename)
     # Unzip the file and write it back to S3 in the same bucket
          s3_resource.meta.client.upload_fileobj(z.open(filename
), Bucket=bucket, Key=f'{filename}')
```

step 2: I used this code to get total count of files and image

```
import boto3
import pandas as pd
s3 = boto3.resource('s3')
my_bucket = s3.Bucket('project-data-photos')
df = pd.read_csv('s3://project-data-photos/train.csv')
results = df.groupby('label').label.agg(['count'])
results.to_csv('s3://project-data-photos/train_analysis.csv')
     print(results)
```



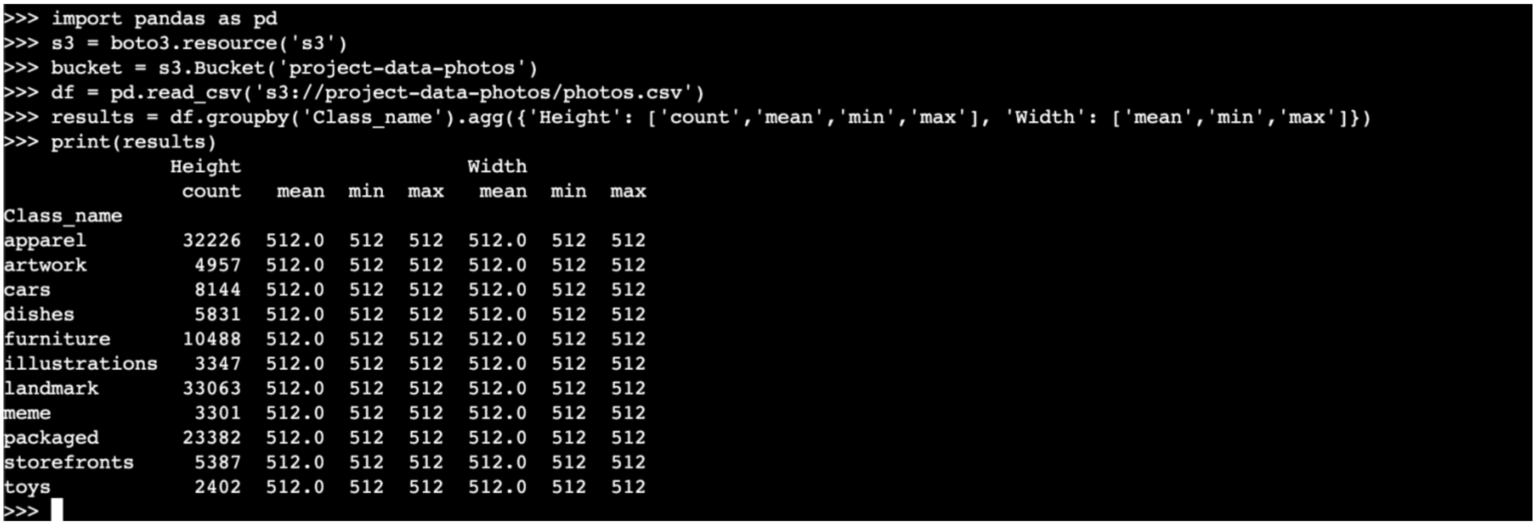
```
working on train.csv
>>>
>>> import boto3
>>> import pandas as pd
>>> s3 = boto3.resource('s3')
>>> my_bucket = s3.Bucket('project-data-photos')
>>> df = pd.read_csv('s3://project-data-photos/train.csv')
>>> results = df.groupby('label').label.agg(['count'])
>>> results.to_csv('s3://project-data-photos/train_analysis.csv')
>>> print(results)
               count
label
apparel        32226
artwork         4957
cars            8144
dishes          5831
furniture      10488
illustrations   3347
landmark       33063
meme            3301
packaged       23382
storefronts     5387
toys            2402
>>>
```

Step3: Made a CSV file with the image details

```
     from PIL import Image
     import boto3
     import botocore.exceptions
     import csv
     import pandas as pd
     s3 = boto3.resource('s3')
     bucket = s3.Bucket('project-data-photos')
     fields = ['Image_name', 'Width', 'Height', 'Class_name',
'Dimension']
     rows = []
     for object in bucket.objects.all():
             if 'jpg' or 'jpeg' or 'png' in object.key:
                     if 'zip' in object.key:
                             continue
                     print(f"Working with: {object.key}")
                     name = object.key.split('/')[-1]
                     slash = object.key.index('/')
                     class_name = object.key[:slash]
                     try:
                             bucket.download_file(object.key,
name)
                             img = Image.open(name)
                             width, height = img.size
                             dimension = f"{width}x{height}"
                             entry = [name, width, height,
class_name, dimension]
                             rows.append(entry)
                     except
botocore.exceptions.DataNotFoundError as e:
                             print(e)

     df=pd.DataFrame(data = rows, columns = fields)
     df.to_csv('s3://project-data-photos/photos.csv',
index=False)
```

Step 4: I used these code to extract information about the data

```
import boto3
import pandas as pd
s3 = boto3.resource('s3')
bucket = s3.Bucket('project-data-photos')
df = pd.read_csv('s3://project-data-photos/photos.csv')
results = df.groupby('Class_name').agg({'Height':
['count','mean','min','max'], 'Width': ['mean','min','max']})
print(results)
```

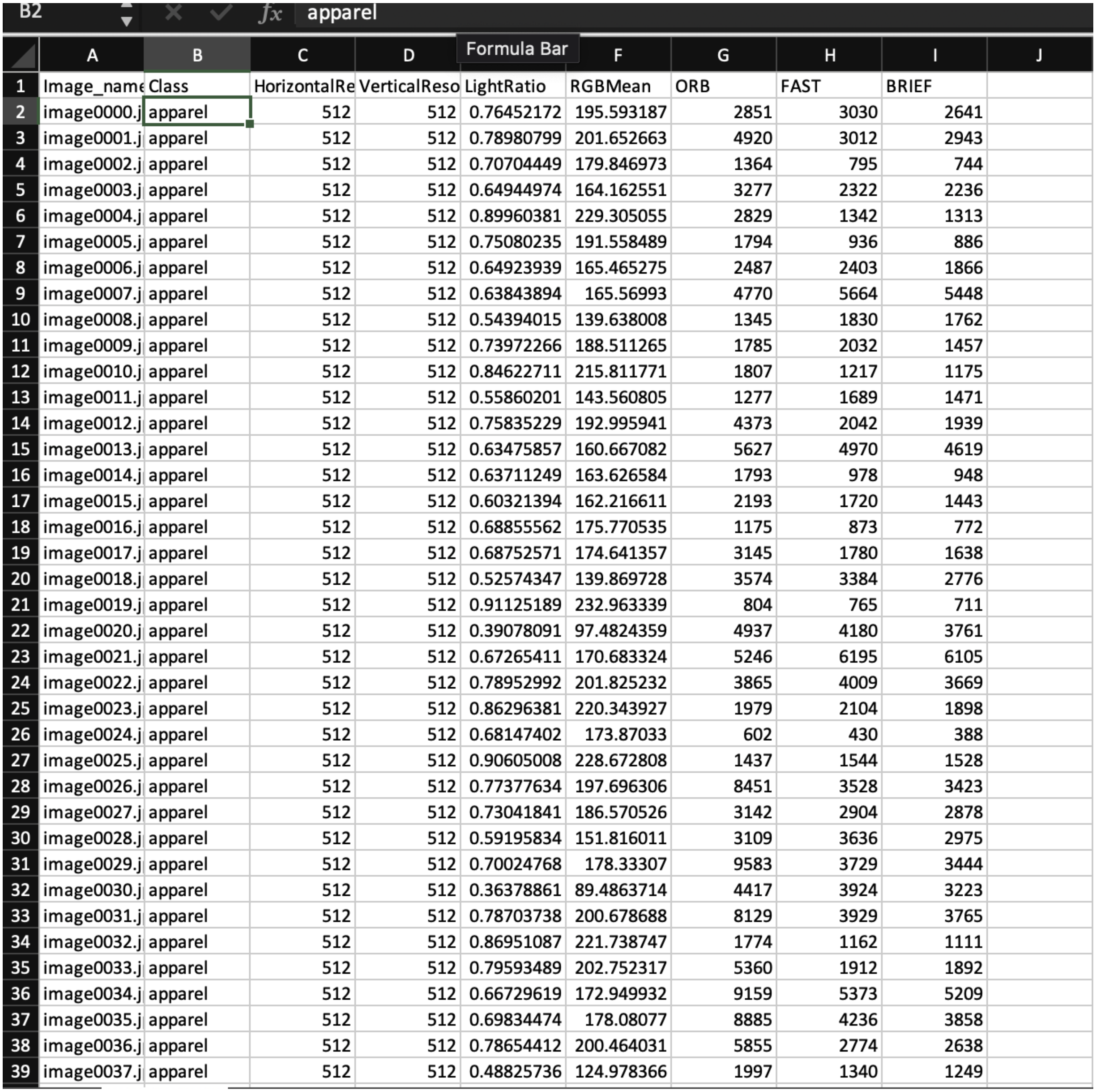


```
>>> import pandas as pd
>>> s3 = boto3.resource('s3')
>>> bucket = s3.Bucket('project-data-photos')
>>> df = pd.read_csv('s3://project-data-photos/photos.csv')
>>> results = df.groupby('Class_name').agg({'Height': ['count','mean','min','max'], 'Width': ['mean','min','max']})
>>> print(results)
              Height                    Width
               count   mean  min  max   mean  min  max
Class_name
apparel        32226  512.0  512  512  512.0  512  512
artwork         4957  512.0  512  512  512.0  512  512
cars            8144  512.0  512  512  512.0  512  512
dishes          5831  512.0  512  512  512.0  512  512
furniture      10488  512.0  512  512  512.0  512  512
illustrations   3347  512.0  512  512  512.0  512  512
landmark       33063  512.0  512  512  512.0  512  512
meme            3301  512.0  512  512  512.0  512  512
packaged       23382  512.0  512  512  512.0  512  512
storefronts     5387  512.0  512  512  512.0  512  512
toys            2402  512.0  512  512  512.0  512  512
>>>
```

Milestone 4 : Coding and Modeling:

```
     from skimage.io import imread, imshow
     import boto3
     import botocore.exceptions
     import csv
     import numpy as np
     import pandas as pd
     import cv2 as cv
     from PIL import ImageFile

     ImageFile.LOAD_TRUNCATED_IMAGES = True
     s3 = boto3.resource('s3')
     bucket = s3.Bucket('project-data-photos')

     fields = ['Image_name', 'Label', 'HorizontalResolution',
'VerticalResolution',\
              'LightRatio', 'RGBMean','ORB', 'FAST', 'BRIEF'] #,
'SURF', 'SIFT']
     rows = []
     for object in bucket.objects.all():
             if 'jpg' or 'jpeg' or 'png' in object.key:
                     if 'zip' in object.key:
                             continue
                     print(f"Working with: {object.key}")
                     name = object.key.split('/')[-1]
                     slash = object.key.index('/')
                     class_name = object.key[:slash]
                     try:
                             bucket.download_file(object.key,
name)
                             # read image in RGB
                             try:
                                     img = imread(name)
                             except:
                                     print("Couldn't read file")
                                     continue

     try:
                                     width, height = 
img.shape[:2]
                             except:
                                     width, height = img.shape
                             pixel = np.sum(width*height)
                             rgbmean = np.average(img)
                             # Computing ORB
                             orb = 
cv.ORB_create(nfeatures=50000)
                             orbkeypoints, orbdescriptor = 
orb.detectAndCompute(img, None)
                             # computing FAST
                             fast = 
cv.FastFeatureDetector_create()
                             fastkeypoints = fast.detect(img, 
None)
                             # computing BRIEF
                             brief = 
cv.xfeatures2d.BriefDescriptorExtractor_create()
                             briefkeypoints, briefdescriptor = 
brief.compute(img, fastkeypoints)
                             # computing Light ratio
                             img = imread(name, as_gray=True)
                             lightratio = np.sum(img)/pixel

     # creating entries
                             entry = [name, class_name, width, 
height,lightratio, rgbmean,len(orbkeypoints),\

len(fastkeypoints),len(briefkeypoints)]
                             rows.append(entry)
                     except 
botocore.exceptions.DataNotFoundError as e:
                             print(e)

     # create .csv from dataframe
     df=pd.DataFrame(data = rows, columns = fields)
     df.to_csv("my_file.csv",  index=False)ed
```

I use this Pyspark to the csv file and the all the images which



	A	B	C	D	E	F	G	H	I	J
1	Image_name	Class	HorizontalRe	VerticalReso	LightRatio	RGBMean	ORB	FAST	BRIEF	
2	image0000.j	apparel	512	512	0.76452172	195.593187	2851	3030	2641	
3	image0001.j	apparel	512	512	0.78980799	201.652663	4920	3012	2943	
4	image0002.j	apparel	512	512	0.70704449	179.846973	1364	795	744	
5	image0003.j	apparel	512	512	0.64944974	164.162551	3277	2322	2236	
6	image0004.j	apparel	512	512	0.89960381	229.305055	2829	1342	1313	
7	image0005.j	apparel	512	512	0.75080235	191.558489	1794	936	886	
8	image0006.j	apparel	512	512	0.64923939	165.465275	2487	2403	1866	
9	image0007.j	apparel	512	512	0.63843894	165.56993	4770	5664	5448	
10	image0008.j	apparel	512	512	0.54394015	139.638008	1345	1830	1762	
11	image0009.j	apparel	512	512	0.73972266	188.511265	1785	2032	1457	
12	image0010.j	apparel	512	512	0.84622711	215.811771	1807	1217	1175	
13	image0011.j	apparel	512	512	0.55860201	143.560805	1277	1689	1471	
14	image0012.j	apparel	512	512	0.75835229	192.995941	4373	2042	1939	
15	image0013.j	apparel	512	512	0.63475857	160.667082	5627	4970	4619	
16	image0014.j	apparel	512	512	0.63711249	163.626584	1793	978	948	
17	image0015.j	apparel	512	512	0.60321394	162.216611	2193	1720	1443	
18	image0016.j	apparel	512	512	0.68855562	175.770535	1175	873	772	
19	image0017.j	apparel	512	512	0.68752571	174.641357	3145	1780	1638	
20	image0018.j	apparel	512	512	0.52574347	139.869728	3574	3384	2776	
21	image0019.j	apparel	512	512	0.91125189	232.963339	804	765	711	
22	image0020.j	apparel	512	512	0.39078091	97.4824359	4937	4180	3761	
23	image0021.j	apparel	512	512	0.67265411	170.683324	5246	6195	6105	
24	image0022.j	apparel	512	512	0.78952992	201.825232	3865	4009	3669	
25	image0023.j	apparel	512	512	0.86296381	220.343927	1979	2104	1898	
26	image0024.j	apparel	512	512	0.68147402	173.87033	602	430	388	
27	image0025.j	apparel	512	512	0.90605008	228.672808	1437	1544	1528	
28	image0026.j	apparel	512	512	0.77377634	197.696306	8451	3528	3423	
29	image0027.j	apparel	512	512	0.73041841	186.570526	3142	2904	2878	
30	image0028.j	apparel	512	512	0.59195834	151.816011	3109	3636	2975	
31	image0029.j	apparel	512	512	0.70024768	178.33307	9583	3729	3444	
32	image0030.j	apparel	512	512	0.36378861	89.4863714	4417	3924	3223	
33	image0031.j	apparel	512	512	0.78703738	200.678688	8129	3929	3765	
34	image0032.j	apparel	512	512	0.86951087	221.738747	1774	1162	1111	
35	image0033.j	apparel	512	512	0.79593489	202.752317	5360	1912	1892	
36	image0034.j	apparel	512	512	0.66729619	172.949932	9159	5373	5209	
37	image0035.j	apparel	512	512	0.69834474	178.08077	8885	4236	3858	
38	image0036.j	apparel	512	512	0.78654412	200.464031	5855	2774	2638	
39	image0037.j	apparel	512	512	0.48825736	124.978366	1997	1340	1249	

includes

Which include the file name , class,Horizontal Resolution, Vertical Resolution, RGBMean, ORB, FAST, BRIEF. After finding details about all the I wanted to predict 2 models.

The confusion matrix and Precision and Recall Curve.

To find the datasets first I created EMR 6.9.0 **Applications** Spark: Spark 3.3.0 on Hadoop 3.3.3 HDFS and Zeppelin 0.10.1. then on the EMR I downloaded this command,

```
pip3 install matplotlib
```

the code I used for modeling :

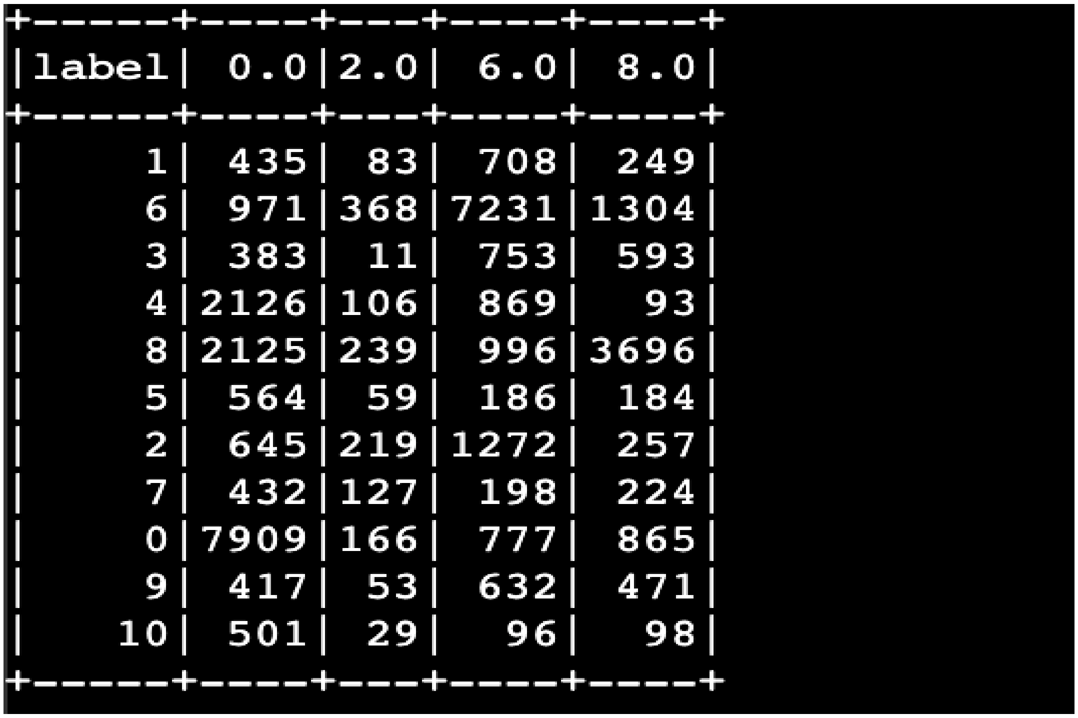
```
from pyspark.sql.functions import *
from pyspark.ml.feature import StringIndexer, OneHotEncoder,
VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression,
LogisticRegressionModel
from pyspark.ml.evaluation import *
from pyspark.ml.tuning import *
from itertools import chain
from pyspark.sql.types import *
import numpy as np

spark = SparkSession.builder.getOrCreate()

photosdf = spark.read.csv("s3://project-data-
photos/my_file.csv/", header=True, inferSchema=True)


photosdf.groupby("Class").count().show()
labels={'apparel':0,
'artwork':1,'cars':2,'dishes':3,'furniture':4,
'illustrations':5, 'landmark':6, 'meme':7, 'packaged':8,
'storefronts':9, 'toys':10}
labels_expr = create_map([lit(x) for x in
chain(*labels.items())])
photosdf = photosdf.withColumn("label",
labels_expr[col("Class")])
```

```
assembler = VectorAssembler(inputCols=["HorizontalResolution",
"VerticalResolution", "LightRatio", "RGBMean", "ORB", "FAST",
"BRIEF"], outputCol="features")

photospipe = Pipeline(stages=[assembler])
transformed_photosdf =
photospipe.fit(photosdf).transform(photosdf)
x = transformed_photosdf.sort("LightRatio").sample(0.25, 1234)
x.show()
trainingData, testData =
transformed_photosdf.randomSplit([0.7,0.3])
lr = LogisticRegression(maxIter=10, regParam=0.3,
elasticNetParam=0.8)

grid = ParamGridBuilder()
grid = grid.addGrid(lr.regParam, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0])
grid = grid.addGrid(lr.elasticNetParam, [0, 1])
grid = grid.build()

evaluator = MulticlassClassificationEvaluator()
cv = CrossValidator(estimator=lr, estimatorParamMaps=grid,
evaluator=evaluator)
all_models = cv.fit(trainingData)




bestModel = all_models.bestModel

test_results = bestModel.transform(testData)

test_results.select('Image_name','Class','probability',
'prediction',
'label').show(truncate=False)

print(evaluator.evaluate(test_results))

test_results.groupby('label').pivot('prediction').count().fillna
(0).show()
```

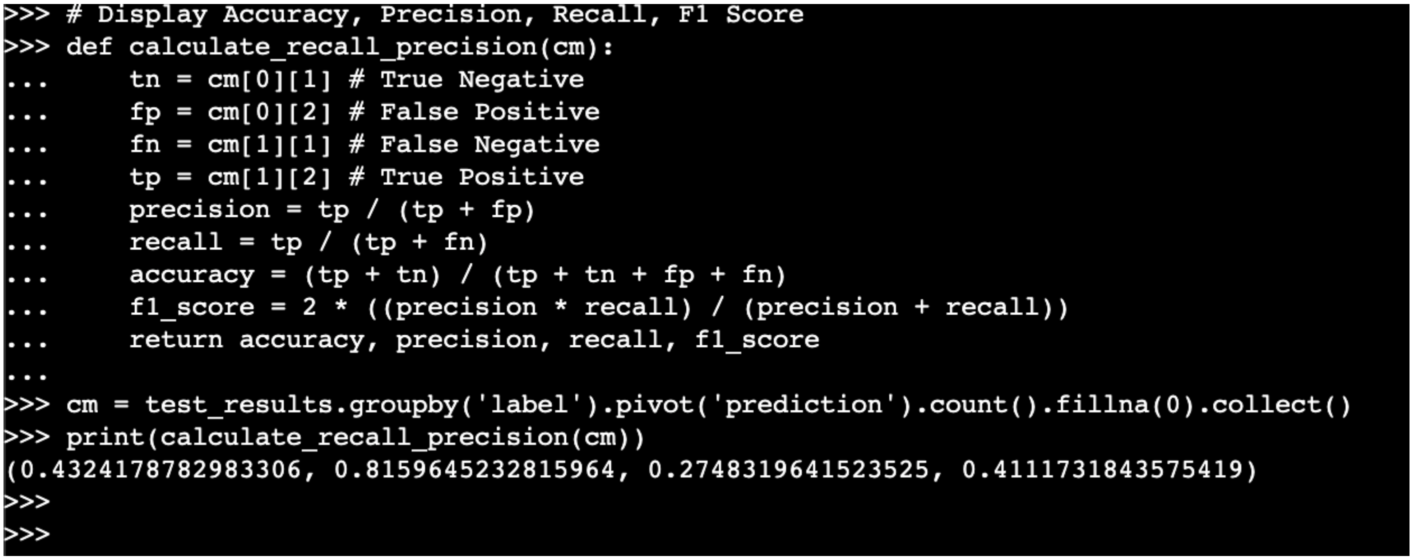
```
model_path = "s3://project-data-
photos/photos_logistic_regression"
bestModel.write().overwrite().save(model_path)




# Display Accuracy, Precision, Recall, F1 Score
def calculate_recall_precision(cm):
     tn = cm[0][1] # True Negative
     fp = cm[0][2] # False Positive
     fn = cm[1][1] # False Negative
     tp = cm[1][2] # True Positive
     precision = tp / (tp + fp)
     recall = tp / (tp + fn)
     accuracy = (tp + tn) / (tp + tn + fp + fn)
     f1_score = 2 * ((precision * recall) / (precision +
recall))
     return accuracy, precision, recall, f1_score

cm =
test_results.groupby('label').pivot('prediction').count().fillna
(0).collect()
print(calculate_recall_precision(cm))


# ROC Curve

import matplotlib.pyplot as plt

plt.figure(figsize=(5,5))

plt.plot(bestModel.summary.precisionByLabel,
bestModel.summary.recallByLabel)
plt.xlabel(‘precision’)

plt.ylabel('Recall')

plt.title("Precision and Recall Curve")

plt.savefig("PRC1.png")
```

Milestone 5 Visualizing Results:

The confusion matrix :

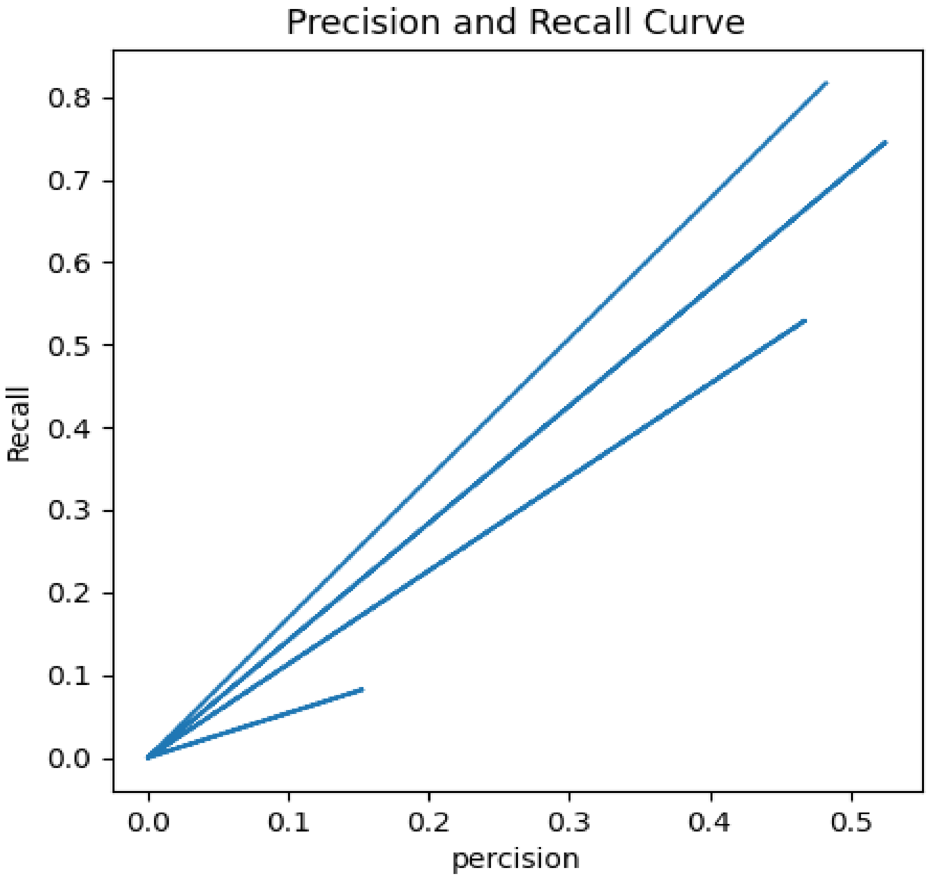


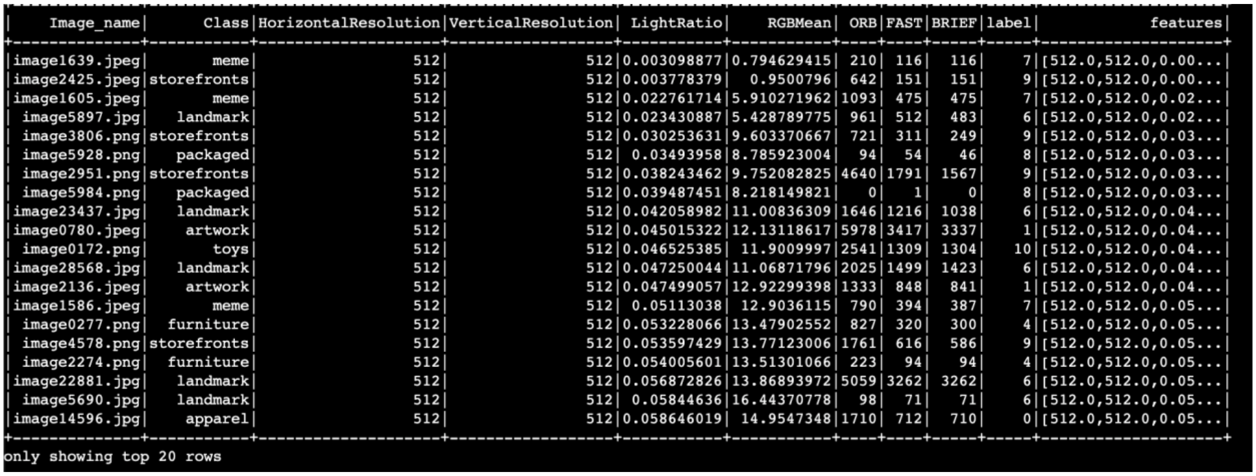
```
+-----+----+---+----+----+
|label| 0.0|2.0| 6.0| 8.0|
+-----+----+---+----+----+
|    1| 435| 83| 708| 249|
|    6| 971|368|7231|1304|
|    3| 383| 11| 753| 593|
|    4|2126|106| 869|  93|
|    8|2125|239| 996|3696|
|    5| 564| 59| 186| 184|
|    2| 645|219|1272| 257|
|    7| 432|127| 198| 224|
|    0|7909|166| 777| 865|
|    9| 417| 53| 632| 471|
|   10| 501| 29|  96|  98|
+-----+----+---+----+----+
```



```
>>> # Display Accuracy, Precision, Recall, F1 Score
>>> def calculate_recall_precision(cm):
...     tn = cm[0][1] # True Negative
...     fp = cm[0][2] # False Positive
...     fn = cm[1][1] # False Negative
...     tp = cm[1][2] # True Positive
...     precision = tp / (tp + fp)
...     recall = tp / (tp + fn)
...     accuracy = (tp + tn) / (tp + tn + fp + fn)
...     f1_score = 2 * ((precision * recall) / (precision + recall))
...     return accuracy, precision, recall, f1_score
...
>>> cm = test_results.groupby('label').pivot('prediction').count().fillna(0).collect()
>>> print(calculate_recall_precision(cm))
(0.4324178782983306, 0.8159645232815964, 0.2748319641523525, 0.4111731843575419)
>>>
>>>
```

```
Precision and Recall Curve:
```





```
+--------------+-----------+--------------------+------------------+-----------+-----------+----+----+-----+-----+--------------------+
|    Image_name|      Class|HorizontalResolution|VerticalResolution| LightRatio|    RGBMean| ORB|FAST|BRIEF|label|            features|
+--------------+-----------+--------------------+------------------+-----------+-----------+----+----+-----+-----+--------------------+
|image1639.jpeg|       meme|                 512|               512|0.003098877|0.794629415| 210| 116|  116|    7|[512.0,512.0,0.00...|
|image2425.jpeg|storefronts|                 512|               512|0.003778379|  0.9500796| 642| 151|  151|    9|[512.0,512.0,0.00...|
|image1605.jpeg|       meme|                 512|               512|0.022761714|5.910271962|1093| 475|  475|    7|[512.0,512.0,0.02...|
| image5897.jpg|   landmark|                 512|               512|0.023430887|5.428789775| 961| 512|  483|    6|[512.0,512.0,0.02...|
| image3806.png|storefronts|                 512|               512|0.030253631|9.603370667| 721| 311|  249|    9|[512.0,512.0,0.03...|
| image5928.png|   packaged|                 512|               512| 0.03493958|8.785923004|  94|  54|   46|    8|[512.0,512.0,0.03...|
| image2951.png|storefronts|                 512|               512|0.038243462|9.752082825|4640|1791| 1567|    9|[512.0,512.0,0.03...|
| image5984.png|   packaged|                 512|               512|0.039487451|8.218149821|   0|   1|    0|    8|[512.0,512.0,0.03...|
|image23437.jpg|   landmark|                 512|               512|0.042058982|11.00836309|1646|1216| 1038|    6|[512.0,512.0,0.04...|
|image0780.jpeg|    artwork|                 512|               512|0.045015322|12.13118617|5978|3417| 3337|    1|[512.0,512.0,0.04...|
| image0172.png|       toys|                 512|               512|0.046525385| 11.9009997|2541|1309| 1304|   10|[512.0,512.0,0.04...|
|image28568.jpg|   landmark|                 512|               512|0.047250044|11.06871796|2025|1499| 1423|    6|[512.0,512.0,0.04...|
|image2136.jpeg|    artwork|                 512|               512|0.047499057|12.92299398|1333| 848|  841|    1|[512.0,512.0,0.04...|
|image1586.jpeg|       meme|                 512|               512| 0.05113038| 12.9036115| 790| 394|  387|    7|[512.0,512.0,0.05...|
| image0277.png|  furniture|                 512|               512|0.053228066|13.47902552| 827| 320|  300|    4|[512.0,512.0,0.05...|
| image4578.png|storefronts|                 512|               512|0.053597429|13.77123006|1761| 616|  586|    9|[512.0,512.0,0.05...|
| image2274.png|  furniture|                 512|               512|0.054005601|13.51301066| 223|  94|   94|    4|[512.0,512.0,0.05...|
|image22881.jpg|   landmark|                 512|               512|0.056872826|13.86893972|5059|3262| 3262|    6|[512.0,512.0,0.05...|
| image5690.jpg|   landmark|                 512|               512| 0.05844636|16.44370778|  98|  71|   71|    6|[512.0,512.0,0.05...|
|image14596.jpg|    apparel|                 512|               512|0.058646019| 14.9547348|1710| 712|  710|    0|[512.0,512.0,0.05...|
+--------------+-----------+--------------------+------------------+-----------+-----------+----+----+-----+-----+--------------------+
only showing top 20 rows
```