Group members:

Cuiting Zhong(cz6425), Wenying Hu(wh7893), Xueru Rong(xr677), Lining Jiang(lj8823), Tianxin Huang(th29885), Rulan Pan(rp33973)

Task A

Use the Google cloud vision (they are better than Azure and AWS in image analytics for sure) to obtain image tags for each post.

In [ ]:

```
# collect image data and corresponding metadata
!instagram-scraper natgeo -u [redacted] -p [redacted] -m 1000 --media-types image --media-metadata
!pip install google-cloud-vision
```

In [ ]:

```
# data preprocessing - get #likes, #comments, caption
import pandas as pd
import re
df=pd.read_json('natgeo/natgeo.json')
df_sub=df[df.is_video==False]
df_sub=df_sub[['display_url','edge_media_preview_like','edge_media_to_caption','edge_media_to_comment']]


df_sub['imgid']=df_sub.display_url.map(lambda x:re.findall('\d+_\d+_\d+_n.jpg',x)[0])
df_sub['likes']=df_sub.edge_media_preview_like.map(lambda x:x['count'])
df_sub['comments']=df_sub.edge_media_to_comment.map(lambda x:x['count'])
df_sub['caption']=df_sub.edge_media_to_caption.map(lambda x:x['edges'][0]['node']['text'])

df_sub2=df_sub[['imgid','likes','comments','caption']]
df_sub2=df_sub2.reset_index(drop=True)
df_sub2.to_csv('metadata.csv')
```

```
In [ ]: # using google vision to get the labels (set the confidence score greater than 0.8)
from google.cloud import vision
from google.cloud.vision import types

result = []

# Instantiates a client
client = vision.ImageAnnotatorClient()

# The name of the image file to annotate
data_path = "natgeo"
data_dir_list = os.listdir(data_path)
i = 1
for img in data_dir_list:
    file_name = data_path + '/'+ img
    # Loads the image into memory
    with io.open(file_name, 'rb') as image_file:
        content = image_file.read()

    image = types.Image(content=content)
    # Performs label detection on the image file
    response = client.label_detection(image=image)
    labels = response.label_annotations
    temp_result = []
    for label in labels:
        if label.score>0.8:
            temp_result.append(label.description)
        else:
            break
    result.append((img,temp_result))
    print(i)
    i=i+1
```

```
In [ ]: final_result = pd.DataFrame(result, columns=('Image_ID', 'labels'))
final_result.to_csv('labels.csv')
df=pd.read_csv('metadata.csv')
df2=pd.merge(final_result, df,how='inner', on=None, left_on='Image_ID', right_on='imgi
d')
df2['len']=df2.labels.map(lambda x:len(x))
df2=df2[df2.len!=0]
final=df2[['Image_ID','labels','likes','comments','caption']]
final=final.reset_index(drop=True)
```

Task B

Create a metric (score) for engagement by using a weighted sum of #_likes and #_comments. Be sure to normalize #_likes and #_comments. Now create an engagement score = .4#*_likes (normalized)* + *.6*#_comments (normalized).

In [43]:
```
from sklearn import preprocessing

X=final[['likes','comments']]
min_max_scaler=preprocessing.MinMaxScaler()
X_norm=min_max_scaler.fit_transform(X)
X=pd.DataFrame(X_norm)
X['score']=0.4*X[0]+0.6*X[1]
final['engagement']=X['score']
final.to_csv('final.csv',index=False)
```

In [1]:
```
import pandas as pd
data=pd.read_csv('final.csv')
data.head()
```

Out[1]:

	Image_ID	labels	likes	comme
0	41951930_2114510558862016_5441544279503428984_...	['Land vehicle', 'Vehicle', 'Bus', 'Mode of tr...	176375	800
1	42004018_263483694309902_2093494549220855030_n...	['Vertebrate', 'Lion', 'Wildlife', 'Mammal', '...	480668	1319
2	42119673_2084322931898073_3502430754017885895_...	['Sky', 'Atmospheric phenomenon']	215272	621
3	42413773_130087801299936_6575466327402366190_n...	['Mountainous landforms', 'Mountain', 'Sky', '...	421620	972
4	42437700_329174901243947_4111914908916797926_n...	['Natural landscape', 'Aerial photography', 'W...	470052	1042

use caption to do prediction

In [2]:
```
data.engagement.describe()
```

Out[2]:
```
count    848.000000
mean       0.091717
std        0.084088
min        0.000052
25%        0.034769
50%        0.070550
75%        0.119171
max        0.943958
Name: engagement, dtype: float64
```

In [4]:
```
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

X_text, y = list(data['caption']), np.array(data['engagement'])

lemmatiser = WordNetLemmatizer()
documents = []
for i in range(0, len(X_text)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X_text[i]))
    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)
    # Converting to Lowercase
    document = document.lower()
    # Lemmatization
    document = document.split()[3:]
    document = [lemmatiser.lemmatize(word) for word in document]
    document = ' '.join(document)
    documents.append(document)
```

In [5]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=3, max_df=0.8,stop_words=stopwords.words('english'
))  #count word frequency
X_text = vectorizer.fit_transform(documents).toarray()
X_text.shape
```

Out[5]:
```
(848, 3347)
```

In [6]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X_text, y, test_size=0.33, random_s
tate=42)
regressor_caption = DecisionTreeRegressor(random_state = 0)
regressor_caption.fit(X_train, y_train)

print('MSE for test set:', mean_squared_error(y_test, regressor_caption.predict(X_test
)))
```

```
MSE for test set: 0.013866840939810188
```

In [7]:
```
importances = regressor_caption.feature_importances_
importances_idx = np.argsort(importances)[::-1]
feature_names = vectorizer.get_feature_names()
top10 = importances_idx[:10]
print('Top 10 important features:\n',[feature_names[j] for j in top10])
```

```
Top 10 important features:
 ['ban', 'sector', 'follower', 'cub', 'wildlife', 'antarctica', 'ice', 'patagonia',
'ladzinski', 'wild']
```

use label to do prediction

In [8]:
```
from sklearn.feature_extraction.text import CountVectorizer
wordcounter = CountVectorizer(stop_words = 'english')
X_label = wordcounter.fit_transform(data['labels']).todense()
X_label.shape
```

Out[8]:
```
(848, 859)
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X_label, y, test_size=0.33, random_
state=42)
regressor_label = DecisionTreeRegressor(random_state = 0)
regressor_label.fit(X_train, y_train)

print('MSE for test set:', mean_squared_error(y_test, regressor_label.predict(X_test)))
```

```
MSE for test set: 0.006398610814638869
```

In [10]:
```
importances = regressor_label.feature_importances_
importances_idx = np.argsort(importances)[::-1]
feature_names = wordcounter.get_feature_names()
top10 = importances_idx[:10]
print('Top 10 important features:\n',[feature_names[j] for j in top10])
```

```
Top 10 important features:
 ['rhinoceros', 'wildlife', 'snout', 'crocodilia', 'indian', 'water', 'nature', 'bir
d', 'bear', 'lavender']
```

use both label and caption to do prediction

In [11]:
```
df_text=pd.DataFrame(X_text)
df_text=df_text.reset_index()
df_label=pd.DataFrame(X_label)
df_label=df_label.reset_index()
df_text_label=pd.merge(df_text,df_label,on='index')
X_text_label=df_text_label.drop('index',axis=1).values
```

In [12]:
```
X_text_label
```

Out[12]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X_text_label, y, test_size=0.33, ra
ndom_state=42)
regressor_text_label = DecisionTreeRegressor(random_state = 0)
regressor_text_label.fit(X_train, y_train)

print('MSE for test set:', mean_squared_error(y_test, regressor_text_label.predict(X_te
st)))
```

```
MSE for test set: 0.007370647735884857
```

In [25]:
```
importances = regressor_text_label.feature_importances_
importances_idx = np.argsort(importances)[::-1]
feature_names = ['caption_'+s for s in vectorizer.get_feature_names()]+['label_'+s for
s in wordcounter.get_feature_names()]
top10 = importances_idx[:10]
print('Top 10 important features:\n',[feature_names[j] for j in top10])
```

```
Top 10 important features:
 ['caption_ban', 'label_black', 'label_wildlife', 'caption_follower', 'caption_ice',
'caption_wildlife', 'caption_antarctica', 'caption_worn', 'caption_facing', 'caption_
likely']
```

Build a model to predict engagement with image labels (text) as predictors. Is this model better than using captions to predict the same? What if you used both image labels and captions to predict engagement?

We use decision tree regression to predict engagement. The MSE with different predictors are:

MSE for test set of the model with only labels: 0.0064
MSE for test set of the model with only caption : 0.0139
MSE for test set of the model with both: 0.0074

By comparing the MSE,we find that the model with only labels as predictors perform the best.Then, the model with using both labels and captions. The model with only using captions is worst.

Task C

Perform topic modeling (LDA) on the original image labels. Choose an appropriate number of topics. You may want to start with 5 topics, but adjust the number up or down depending on the word distributions you get. Decide on a suitable name for each topic.

In [18]:
```
import pandas as pd
from pandas import DataFrame
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer

data = pd.read_csv("final.csv")

wordcounter = CountVectorizer(stop_words = 'english')
document_dtm = wordcounter.fit_transform(data['labels']).todense()
lda = LatentDirichletAllocation(n_components = 6, random_state = 30)
document_topic = lda.fit_transform(document_dtm)

def print_topic_word(x):
    topic = x.name
    top10_word = ' '.join(x.sort_values(ascending = False)[:10].index.tolist())
    print(topic + ': ' + top10_word)
    return

word_topic_df = DataFrame(lda.components_.T, index = wordcounter.get_feature_names(), c
olumns = ['Topic 1', 'Topic 2', 'Topic 3', 'Topic 4', 'Topic 5','Topic 6'])
result = word_topic_df.apply(print_topic_word)
```

```
C:\Users\cuiti\Anaconda3\lib\site-packages\sklearn\decomposition\online_lda.py:536: D
eprecationWarning: The default value for 'learning_method' will be changed from 'onli
ne' to 'batch' in the release 0.20. This warning was introduced in 0.18.
  DeprecationWarning)

Topic 1: wildlife animal terrestrial vertebrate mammal felidae carnivore cats bear bi
g
Topic 2: sky natural mountain landscape nature phenomenon tree landforms mountainous
environment
Topic 3: sky water black white monochrome cloud photography sea blue rock
Topic 4: marine ice sea water mammal penguin turtle seal biology arctic
Topic 5: bird beak history ancient elephant elephants mammoths site camel community
Topic 6: water people transport vehicle red adaptation mode resources child pink
```

Topics decided:

1. Wild mammal and vertebrate
2. Nature, mountain and landscape
3. Photography
4. Underwater mammal, penguin and turtle
5. Bird, elephant and history
6. Transportation and people

Now sort the data from high to low engagement score, and take the highest and the lowest quartiles (by engagement score). What are the main differences in the average topic weights of images across the two quartiles (e.g., greater weight of some topics in the highest versus lowest engagement quartiles)? Show the main results in a table.

In [28]:
```
#topic weights
document_topic = DataFrame(document_topic, columns = ['Wild mammal and vertebrate', 'Nature, mountain and landscape', 'Photography', 'Underwater mammal, penguin and turtle', 'Bird, elephant and history', 'Transportation and people'])
document_topic.head()
```

Out[28]:

	Wild mammal and vertebrate	Nature, mountain and landscape	Photography	Underwater mammal, penguin and turtle	Bird, elephant and history	Transportation and people
0	0.012821	0.012821	0.012838	0.012821	0.012821	0.935880
1	0.935851	0.012821	0.012849	0.012835	0.012824	0.012821
2	0.041667	0.791250	0.042082	0.041667	0.041667	0.041667
3	0.018519	0.907372	0.018553	0.018519	0.018519	0.018519
4	0.011155	0.312775	0.011212	0.011136	0.011111	0.642610

In [29]:
```
document_topic['engagement'] = data['engagement']
document_topic.head()
```

Out[29]:

	Wild mammal and vertebrate	Nature, mountain and landscape	Photography	Underwater mammal, penguin and turtle	Bird, elephant and history	Transportation and people	engageme
0	0.012821	0.012821	0.012838	0.012821	0.012821	0.935880	0.022630
1	0.935851	0.012821	0.012849	0.012835	0.012824	0.012821	0.098254
2	0.041667	0.791250	0.042082	0.041667	0.041667	0.041667	0.030883
3	0.018519	0.907372	0.018553	0.018519	0.018519	0.018519	0.082160
4	0.011155	0.312775	0.011212	0.011136	0.011111	0.642610	0.094124

In [31]:
```
document_topic.shape
```

Out[31]: (848, 7)

In [38]:
```
import numpy as np
top = document_topic.sort_values('engagement',ascending = False)[:212]
bottom = document_topic.sort_values('engagement',ascending = True)[:212]
```

In [43]:
```
top.mean()
```

Out[43]:
```
Wild mammal and vertebrate                 0.262829
Nature, mountain and landscape             0.198263
Photography                                0.165273
Underwater mammal, penguin and turtle      0.230206
Bird, elephant and history                 0.077241
Transportation and people                  0.066188
engagement                                 0.198921
dtype: float64
```

In [39]:
```
bottom.mean()
```

Out[39]:
```
Wild mammal and vertebrate                 0.062466
Nature, mountain and landscape             0.206479
Photography                                0.278094
Underwater mammal, penguin and turtle      0.069274
Bird, elephant and history                 0.114390
Transportation and people                  0.269297
engagement                                 0.022437
dtype: float64
```

In [54]:
```
# average weights for top quartile and bottom quartile
top_bottom=pd.DataFrame([top.mean(),bottom.mean()]).drop('engagement',axis=1).T
top_bottom.columns=['top','bottom']
top_bottom
```

Out[54]:

	top	bottom
Wild mammal and vertebrate	0.262829	0.062466
Nature, mountain and landscape	0.198263	0.206479
Photography	0.165273	0.278094
Underwater mammal, penguin and turtle	0.230206	0.069274
Bird, elephant and history	0.077241	0.114390
Transportation and people	0.066188	0.269297

In [55]:
```
topic_rank = pd.DataFrame({'Topics with top engagement':list(top_bottom.sort_values('to
p',ascending=False).index),'Topics with lowest engagement':list(top_bottom.sort_values
('bottom',ascending=False).index)})
topic_rank
```

Out[55]:

	Topics with top engagement	Topics with lowest engagement
0	Wild mammal and vertebrate	Photography
1	Underwater mammal, penguin and turtle	Transportation and people
2	Nature, mountain and landscape	Nature, mountain and landscape
3	Photography	Bird, elephant and history
4	Bird, elephant and history	Underwater mammal, penguin and turtle
5	Transportation and people	Wild mammal and vertebrate

In [58]:
```
top_bottom.sort_values('bottom',ascending=False).index
```

Out[58]:
```
Index(['Photography', 'Transportation and people',
       'Nature, mountain and landscape', 'Bird, elephant and history',
       'Underwater mammal, penguin and turtle', 'Wild mammal and vertebrate'],
      dtype='object')
```

For the top quartile engagement, the rank for topics based on the average weights:
['Wild mammal and vertebrate', 'Underwater mammal, penguin and turtle', 'Nature, mountain and landscape', 'Photography', 'Bird, elephant and history', 'Transportation and people']

For the bottom quartile engagement, the rank for topics based on the average weights:
['Photography', 'Transportation and people', 'Nature, mountain and landscape', 'Bird, elephant and history', 'Underwater mammal, penguin and turtle', 'Wild mammal and vertebrate']

Part D

What advice would you give Natgeo to increase engagement on its Instagram page based on your findings in Tasks B and C?

As shown in part b, the 10 important features of model with only labels are ['rhinoceros', 'wildlife', 'snout', 'crocodilia', 'indian', 'water', 'nature', 'bird', 'bear', 'lavender']. It implies that a pictue involving the element of these 10 features may have great influence on the engagement.

The top 10 important features from the model with only caption are: ['ban', 'sector', 'follower', 'cub', 'wildlife', 'antarctica', 'ice', 'patagonia', 'ladzinski', 'wild']. Therefore, mentioning some of those words in the capiton would have important impact on engagement.

From part c, we can find that among those instagram postings with high engagement score, on average, it involves the topic of "wild mammal and vertebrate" with highest weights and the topic of "underwater mammal, penguin and turtle" the second, and "transportation and people" the least. Among those with low engagement score, the topics are often "Photography" and "transportation and people" too.

Overall, Natgeo can post more pictures about wildlife and underwater mammal, penguin and turtle, and less picture with an abstract simple photography and transportation and people.