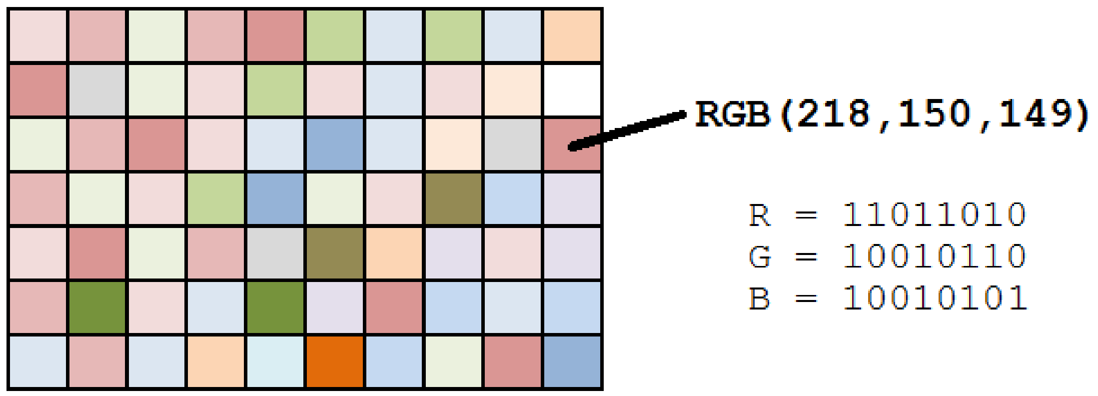
CLUSTERING METHODS WITH SCIPY

Dominant colors in images

Shaumik Daityari
Business Analyst

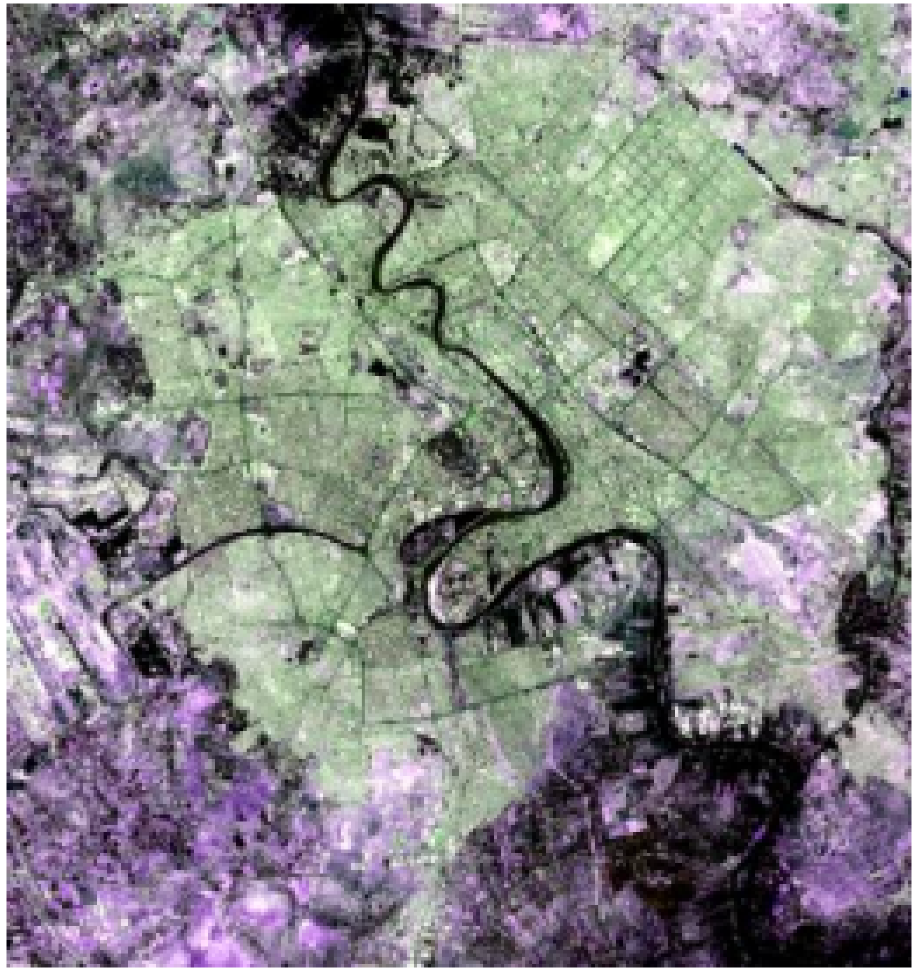
Dominant colors in images

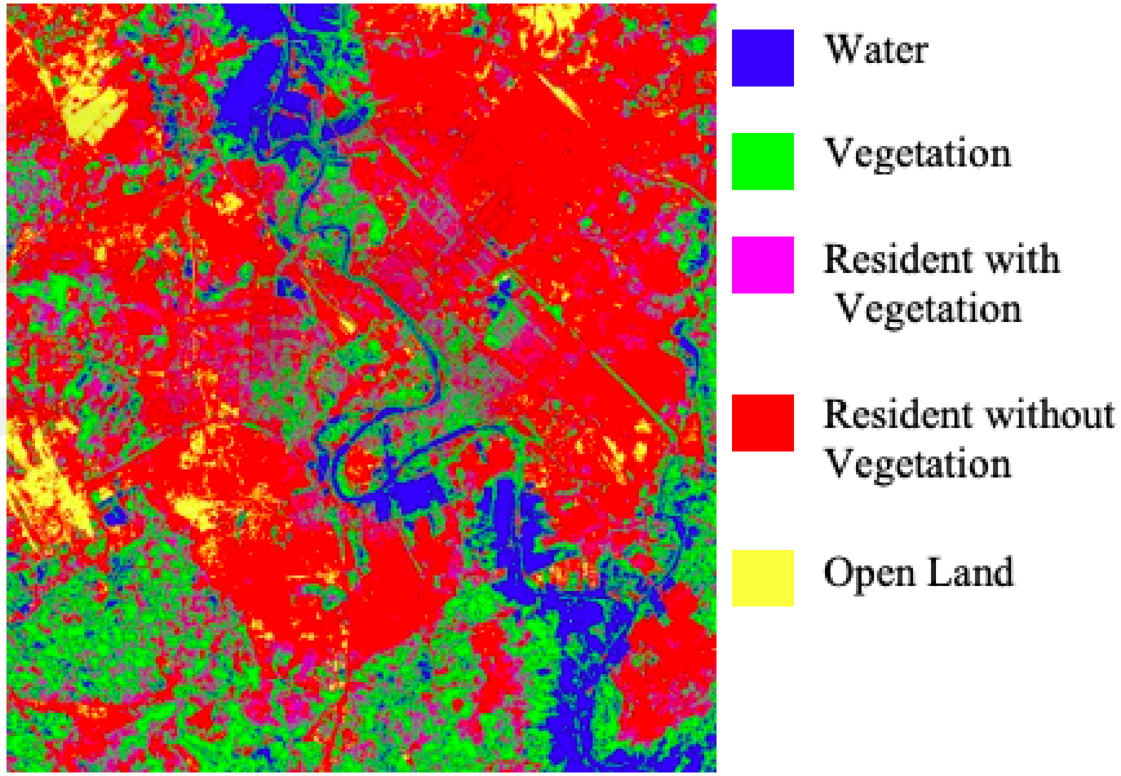
- All images consist of pixels
- Each pixel has three values: *Red*, *Green* and *Blue*
- Pixel color: combination of these RGB values
- Perform k-means on standardized RGB values to find cluster centers
- Uses: Identifying features in satellite images



Source

Feature identification in satellite images





Source

Tools to find dominant colors

- Convert image to pixels: `matplotlib.image.imread`
- Display colors of cluster centers: `matplotlib.pyplot.imshow`

Convert image to RGB matrix

```
import matplotlib.image as img
image = img.imread('sea.jpg')
image.shape
```

```
(475, 764, 3)
```

```
r = []
g = []
b = []

for row in image:
    for pixel in row:
        # A pixel contains RGB values
        temp_r, temp_g, temp_b = pixel
        r.append(temp_r)
        g.append(temp_g)
        b.append(temp_b)
```

Data frame with RGB values

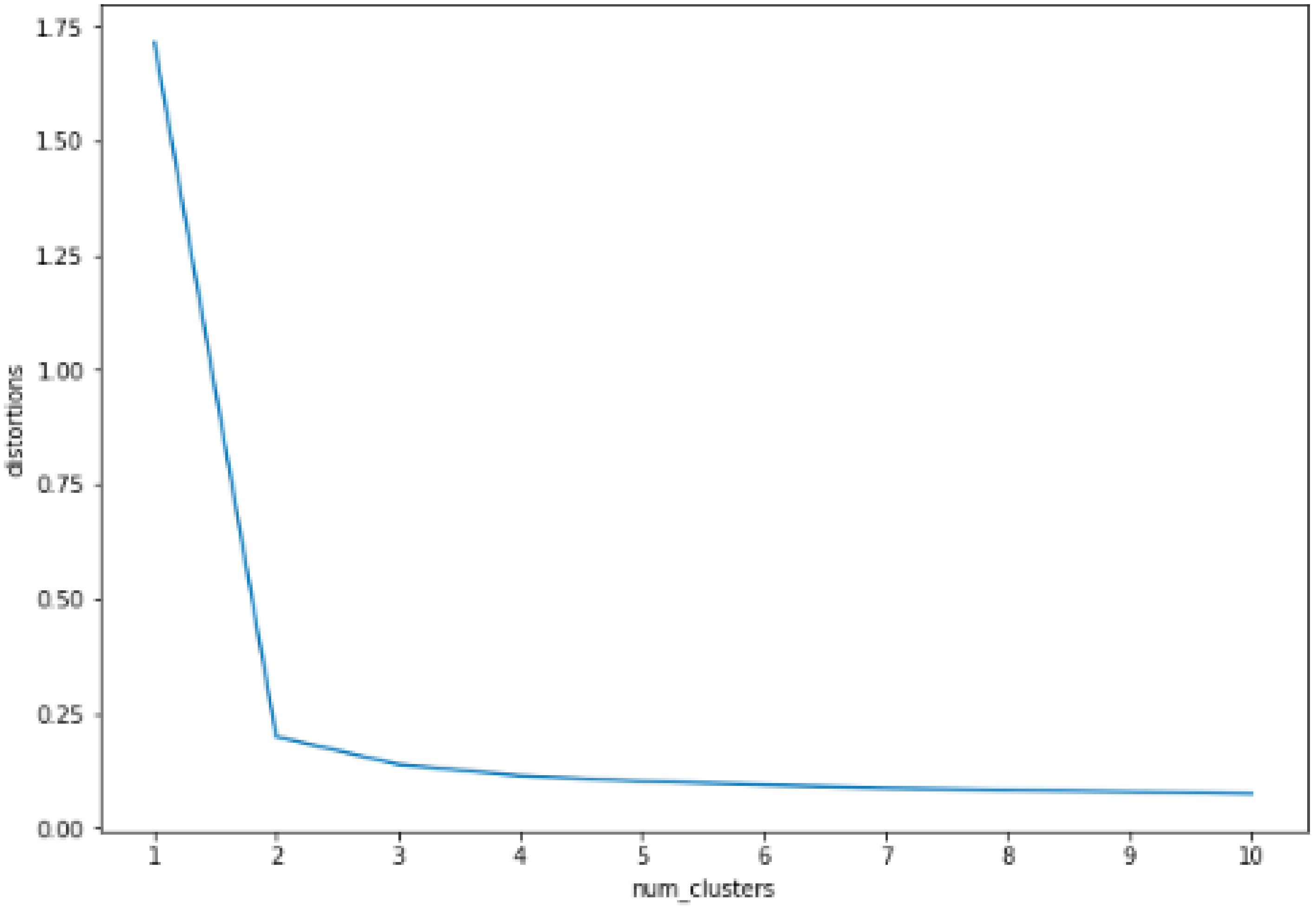
```
pixels = pd.DataFrame({'red': r,
                       'blue': b,
                       'green': g})
pixels.head()
```

red	blue	green
252	255	252
75	103	81
...	...	...

Create an elbow plot

```
distortions = []
num_clusters = range(1, 11)

# Create a list of distortions from the kmeans method
for i in num_clusters:
    cluster_centers, _ = kmeans(pixels[['scaled_red', 'scaled_blue',
                                        'scaled_green']], i)
    distortions.append(distortion)

# Create a data frame with two lists - number of clusters and distortions
elbow_plot = pd.DataFrame({'num_clusters': num_clusters,
                           'distortions': distortions})

# Creat a line plot of num_clusters and distortions
sns.lineplot(x='num_clusters', y='distortions', data = elbow_plot)
plt.xticks(num_clusters)
plt.show()
```

Elbow plot


1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
distortions
1
2
3
4
5
6
7
8
9
10
num_clusters

Find dominant colors

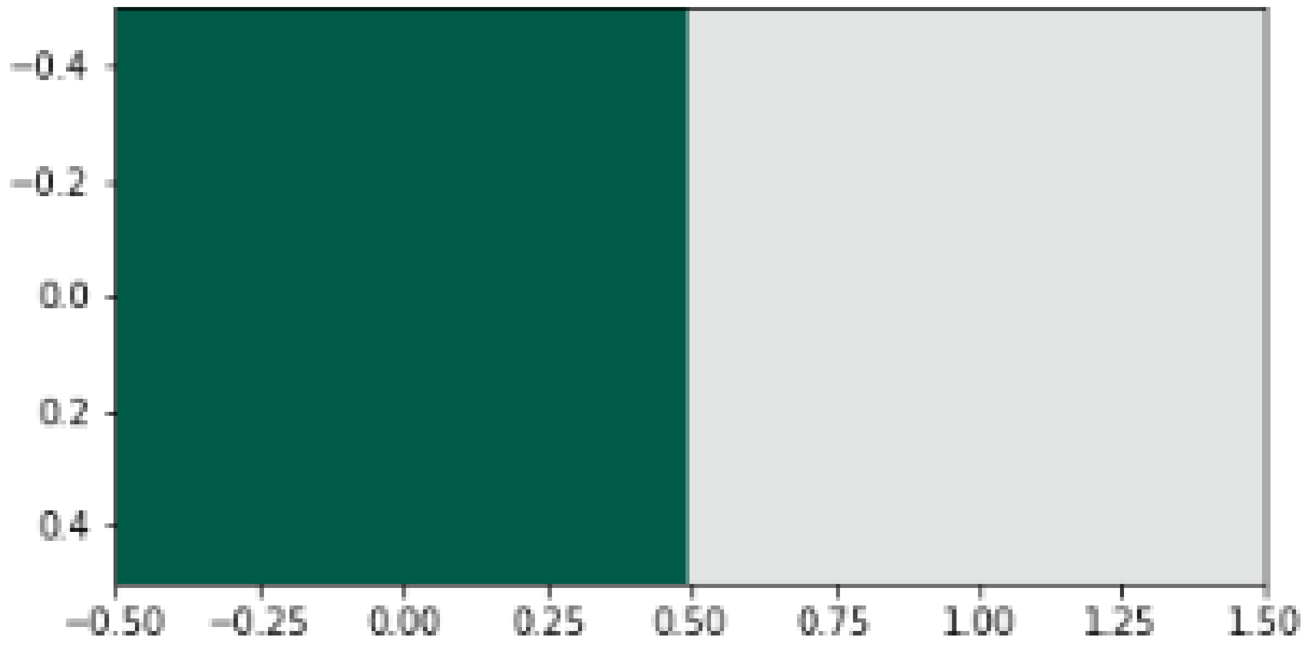
```
cluster_centers, _ = kmeans(pixels[['scaled_red', 'scaled_blue',
                                    'scaled_green']], 2)
```

```
colors = []

# Find Standard Deviations
r_std, g_std, b_std = pixels[['red', 'blue', 'green']].std()

# Scale actual RGB values in range of 0-1
for cluster_center in cluster_centers:
    scaled_r, scaled_g, scaled_b = cluster_center
    colors.append((
        scaled_r * r_std/255,
        scaled_g * g_std/255,
        scaled_b * b_std/255
    ))
```

Display dominant colors

```
#Dimensions: 2 x 3 (N X 3 matrix)
print(colors)
[(0.08192923122023911, 0.34205845943857993, 0.2824002984155429),
 (0.893281510956742, 0.899818770315129, 0.8979114272960784)]
```

```
#Dimensions: 1 x 2 x 3 (1 X N x 3 matrix)
plt.imshow([colors])
plt.show()
```



CLUSTERING METHODS WITH SCIPY

Next up: exercises

CLUSTERING METHODS WITH SCIPY

Document clustering

Shaumik Daityari
Business Analyst

Document clustering: concepts

1. Clean data before processing
2. Determine the importance of the terms in a document (in TF-IDF matrix)
3. Cluster the TF-IDF matrix
4. Find top terms, documents in each cluster

Clean and tokenize data

- Convert text into smaller parts called tokens, clean data for processing

```
from nltk.tokenize import word_tokenize
import re

def remove_noise(text, stop_words = []):
    tokens = word_tokenize(text)

    cleaned_tokens = []
    for token in tokens:
        token = re.sub('[^A-Za-z0-9]+', '', token)

        if len(token) > 1 and token.lower() not in stop_words:
            # Get lowercase
            cleaned_tokens.append(token.lower())
    return cleaned_tokens

remove_noise("It is lovely weather we are having.
              I hope the weather continues.")
```

['lovely', 'weather', 'hope', 'weather', 'continues']

Document term matrix and sparse matrices

- Document term matrix formed
- Most elements in matrix are zeros

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector (Passage Vector)

Document Vector

Source

- Sparse matrix is created

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \Longrightarrow$$

Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

Source

TF-IDF (Term Frequency - Inverse Document Frequency)

- A weighted measure: evaluate how important a word is to a document in a collection

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=50,
                                   min_df=0.2, tokenizer=remove_noise)

tfidf_matrix = tfidf_vectorizer.fit_transform(data)
```

Clustering with sparse matrix

- `kmeans()` in SciPy does not support sparse matrices
- Use `.todense()` to convert to a matrix

```
cluster_centers, distortion = kmeans(tfidf_matrix.todense(), num_clusters)
```

Top terms per cluster

- Cluster centers: lists with a size equal to the number of terms
- Each value in the cluster center is its importance
- Create a dictionary and print top terms

```
terms = tfidf_vectorizer.get_feature_names()

for i in range(num_clusters):
    center_terms = dict(zip(terms, list(cluster_centers[i])))

    sorted_terms = sorted(center_terms, key=center_terms.get, reverse=True)

    print(sorted_terms[:3])
```

['room', 'hotel', 'staff']

['bad', 'location', 'breakfast']

More considerations

- Work with hyperlinks, emoticons etc.
- Normalize words (run, ran, running -> run)
- `.todense()` may not work with large datasets

CLUSTERING METHODS WITH SCIPY

Next up: exercises!

CLUSTERING METHODS WITH SCIPY

Clustering with multiple features

Shaumik Daityari
Business Analyst

Basic checks

- Cluster centers

```
print(fifa.groupby('cluster_labels')[['scaled_heading_accuracy',
    'scaled_volleys', 'scaled_finishing']].mean())
```

cluster_labels	scaled_heading_accuracy	scaled_volleys	scaled_finishing
0	3.21	2.83	2.76
1	0.71	0.64	0.58

- Cluster sizes

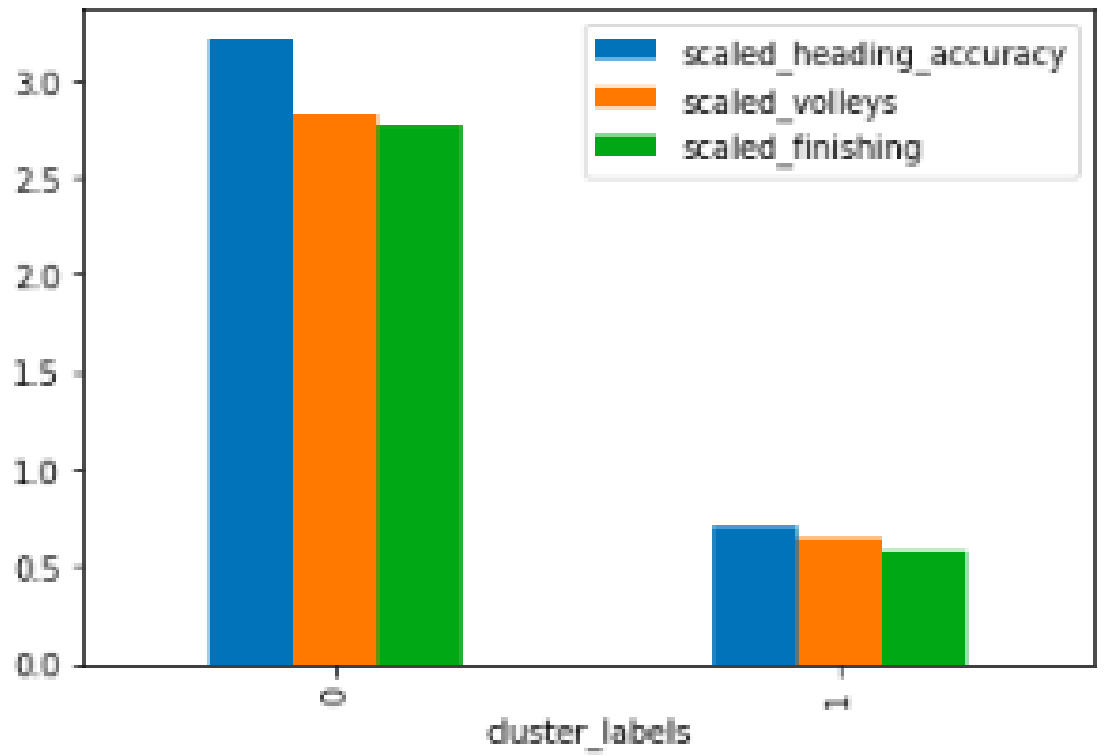
```
print(fifa.groupby('cluster_labels')['ID'].count())
```

cluster_labels	count
0	886
1	114

Visualizations

- Visualize cluster centers
- Visualize other variables for each cluster

```
# Plot cluster centers
fifa.groupby('cluster_labels') \
  [scaled_features].mean()
  .plot(kind='bar')
plt.show()
```



Top items in clusters

```
# Get the name column of top 5 players in each cluster
for cluster in fifa['cluster_labels'].unique():
    print(cluster, fifa[fifa['cluster_labels'] == cluster]['name'].values[:5])
```

Cluster Label	Top Players
0	['Cristiano Ronaldo' 'L. Messi' 'Neymar' 'L. Suárez' 'R. Lewandowski']
1	['M. Neuer' 'De Gea' 'G. Buffon' 'T. Courtois' 'H. Lloris']

Feature reduction

- Factor analysis
- Multidimensional scaling

CLUSTERING METHODS WITH SCIPY

Final exercises!

CLUSTERING METHODS WITH SCIPY

Farewell!

Shaumik Daityari
Business Analyst

What comes next?

- Clustering is one of the exploratory steps
- More courses on DataCamp
- Practice, practice, practice!

CLUSTERING METHODS WITH SCIPY

Until next time