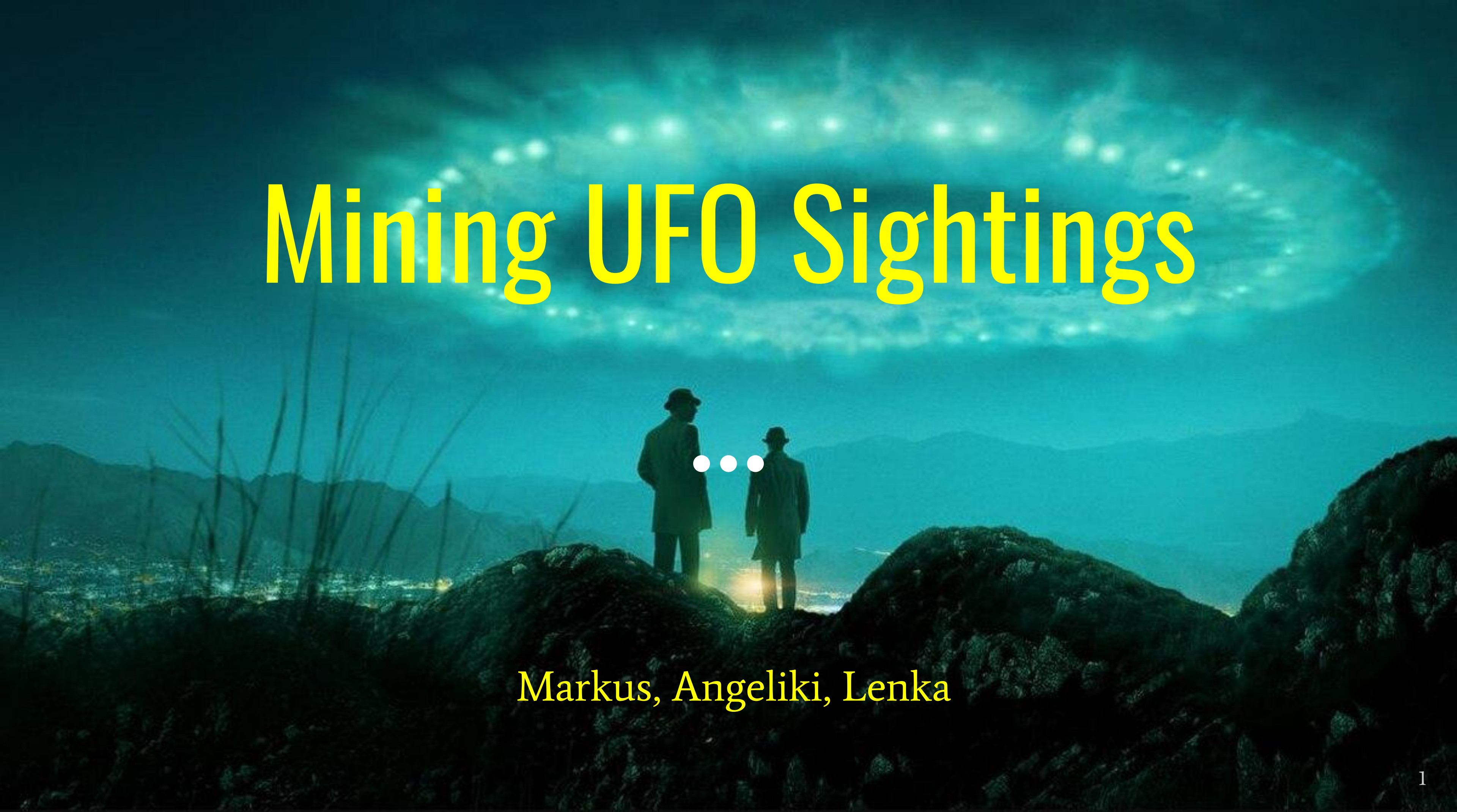


Mining UFO Sightings

Markus, Angeliki, Lenka

Agenda

- Introduction to the UFO Sightings Data
- Initial Cleaning Data

1. Prediction of countries from longitude / latitude (Markus)
2. Prediction of season from the duration in sec (Angeliki)
3. Textual analysis of the comments (Lenka)

UFO Sightings Data

Data from Kaggle https://www.kaggle.com/NUFORC/ufo-sightings

- CSV file containing ~80.000 rows of data loaded into a Pandas DataFrame.
- 11 features:

```
df.shape
```

```
(80332, 11)
```

```
df.head()
```

	datetime	city	state	country	shape	duration (seconds)	duration (hours/min)	comments	date posted	latitude	longitude
0	10/10/1949 20:30	san marcos	tx	us	cylinder	2700.0	45 minutes	This event took place in early fall around 194...	4/27/2004	29.883056	-97.941111
1	10/10/1949 21:00	lackland afb	tx	NaN	light	7200.0	1-2 hrs	1949 Lackland AFB&#44 TX. Lights racing acros...	12/16/2005	29.384210	-98.581082

Initial cleaning

```
df = df.rename(columns = {'longitude ':'longitude' })
```

- Stripped of **weird characters** and made duration (sec) and latitude **to floats.**
- Time duration to sec (already pre-done)

```
df['duration (seconds)'] = df['duration (seconds)'].str.strip('`')
```

```
df['duration (seconds)'] = df['duration (seconds)'].astype(float)
```

```
df['latitude'] = df['latitude'].replace('33q.200088', '33.200088')
df.loc[df['latitude'] == '33.200088']
```

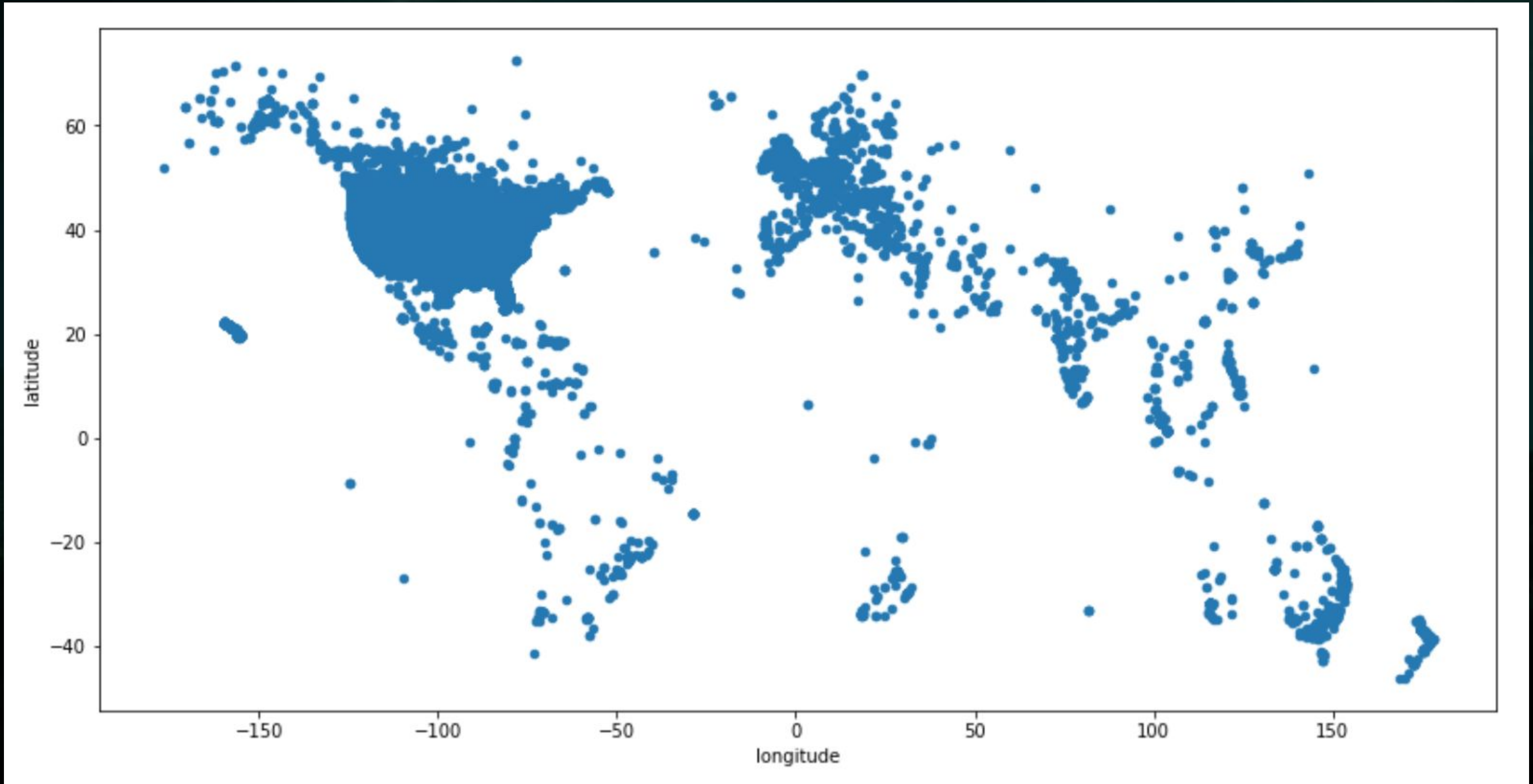
- Cities () -> new countries
  - Only a few countries in Country - in %:
  - Countries often mention in () in city
  - Retrieved new countries from city

us	81.056117
NaN	12.037544
ca	3.734502
gb	2.371409
au	0.669721
de	0.130708

saddle lake
(canada)

us	82.581676
NaN	6.726867
ca	3.804789
gb	2.416041
au	0.682325
canada	0.648082
uk/england	0.379211
mexico	0.251116
india	0.244775
de	0.133168
netherlands	0.126826
new zealand	0.112875
south africa	0.106534
brazil	0.081169
australia	0.077364
spain	0.064681
malaysia	0.060877
france	0.054535
japan	0.051999
philippines	0.050731
sweden	0.045657
belgium	0.045657
china	0.044389
norway	0.044389

Map of longitude and latitude



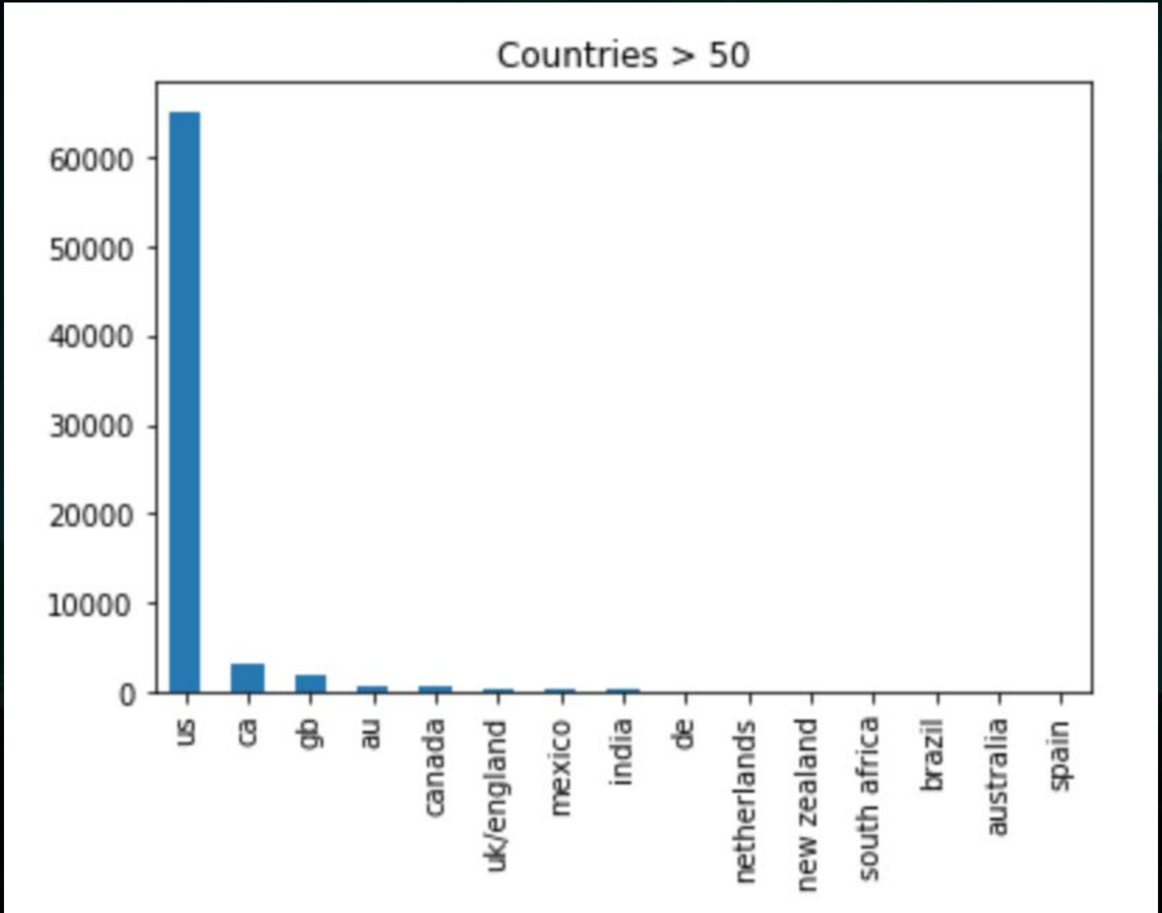
Completing the dataset - Countries

	datetime	city	state	country	shape	duration (seconds)	duration (hours/min)	comments	date posted	latitude	longitude
0	10/10/1949 20:30	san marcos	tx	us	cylinder	2700.0	45 minutes	This event took place in early fall around 194...	4/27/2004	29.883056	-97.941111
1	10/10/1949 21:00	lackland afb	tx	NaN	light	7200.0	1-2 hrs	1949 Lackland AFB&#44 TX. Lights racing acros...	12/16/2005	29.384210	-98.581082
2	10/10/1955 17:00	chester (uk/england)	NaN	gb	circle	20.0	20 seconds	Green/Orange circular disc over Chester&#44 En...	1/21/2008	53.200000	-2.916667
3	10/10/1956 21:00	edna	tx	us	circle	20.0	1/2 hour	My older brother and twin sister were leaving ...	1/17/2004	28.978333	-96.645833
4	10/10/1960 20:00	kaneohe	hi	us	light	900.0	15 minutes	AS a Marine 1st Lt. flying an FJ4B fighter/att...	1/22/2004	21.418056	-157.803611

Filling out the blanks

Classification

- Train/Test set
- Simplest solution → SKLearn ≈ 92 %
- 89 % USA, Continents


Countries > 50
60000
50000
40000
30000
20000
10000
0
us
ca
gb
au
canada
uk/england
mexico
india
de
netherlands
new zealand
south africa
brazil
australia
spain

Databases of Map Coordinates and Countries

Google API

- Google Earth, Google Static Maps
- Online database → SLOW
- Restricted Access

Import reverse_geocode

- Offline database → FASTER
- 120,000 cities
- Country, City and Coordinate

Reverse_geocode

- k-Dimensional Tree
- Train/Test set → NaN
- 97.9 %
- Errors → spelling mistakes vs border areas

Reverse_geocode

- k-Dimensional Tree
- Train/Test set → NaN
- 97.9 %
- Errors → spelling mistakes vs

	country
0	us
1	NaN
2	gb
3	us
4	us
5	us
6	gb
7	us
8	us
9	us
10	us
11	us
12	us
13	us
14	us
15	us
16	us
17	us
18	NaN
19	us

	Country Code	Country
0	us	united states
1	us	united states
2	gb	united kingdom
3	us	united states
4	us	united states
5	us	united states
6	gb	united kingdom
7	us	united states
8	us	united states
9	us	united states
10	us	united states
11	us	united states
12	us	united states
13	us	united states
14	us	united states
15	us	united states
16	us	united states
17	us	united states
18	bm	bermuda
19	us	united states

Can we predict the duration of sightings?

Steps covered:

1. Import the Data
2. Clean up and transform the Data
3. Visualize Data
4. Split training set and test set
5. Fine tune Algorithms (SGDClassifier,AdaBoostClassifier,RandomForestClassifier)
6. Compare accuracy scores
7. End up with the best prediction model

Change variables ufo_date

```
ufo_date = ufo_data.datetime.str.replace('24:00', '00:00')  # clean illegal values
ufo_date = pd.to_datetime(ufo_date, format='%m/%d/%Y %H:%M')  # now in datetime

ufo_data['datetime'] = ufo_data.datetime.str.replace('24:00', '00:00')
ufo_data['datetime'] = pd.to_datetime(ufo_data['datetime'], format='%m/%d/%Y %H:%M')
```

Add season column to ufo_date

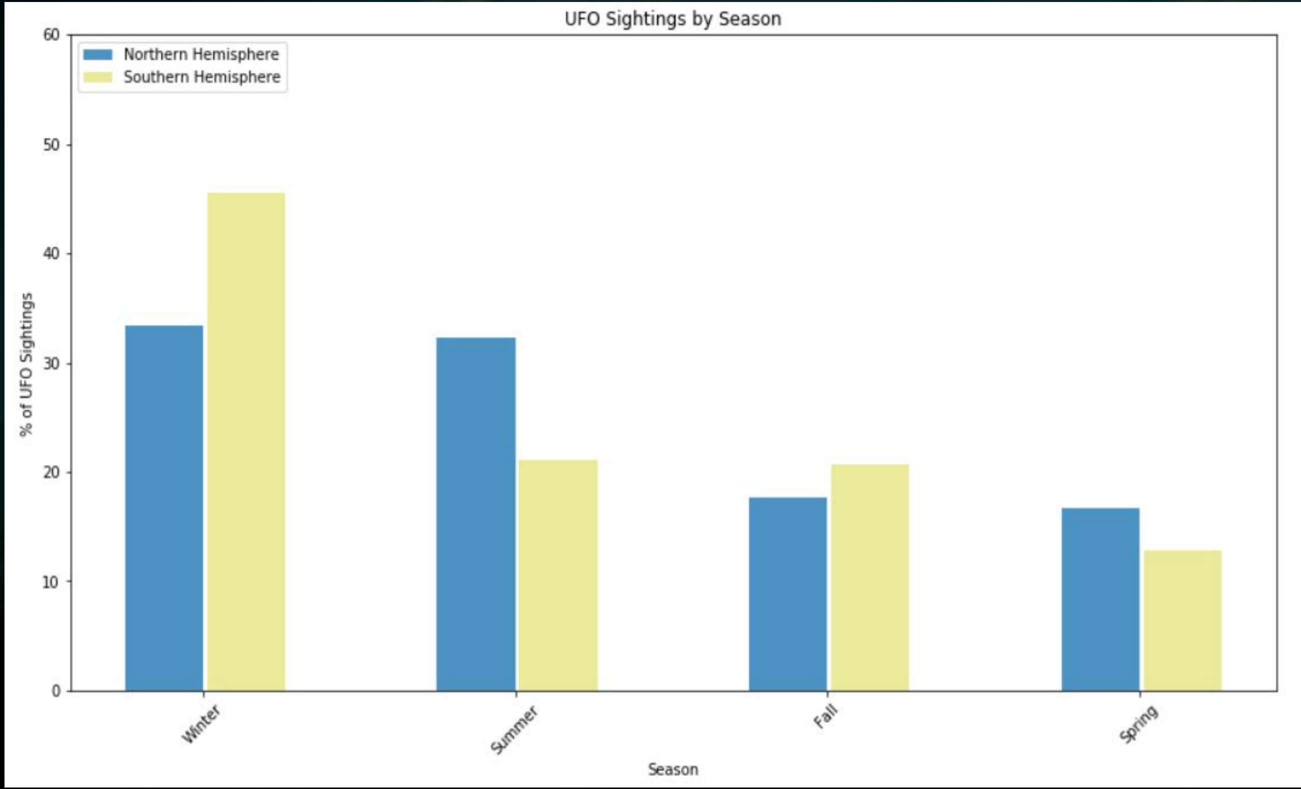
```
ufo_datem = ufo_date.dt.month
spring = range(5,7)
summer = range(7,10)
fall = range(10,12)
seasons = []

for st_date in ufo_datem:
    # Conversion Process #
    if st_date in spring:
        seasons.append('Spring')
    elif st_date in summer:
        seasons.append('Summer')
    elif st_date in fall:
        seasons.append('Fall')
    else:
        seasons.append('Winter')
```

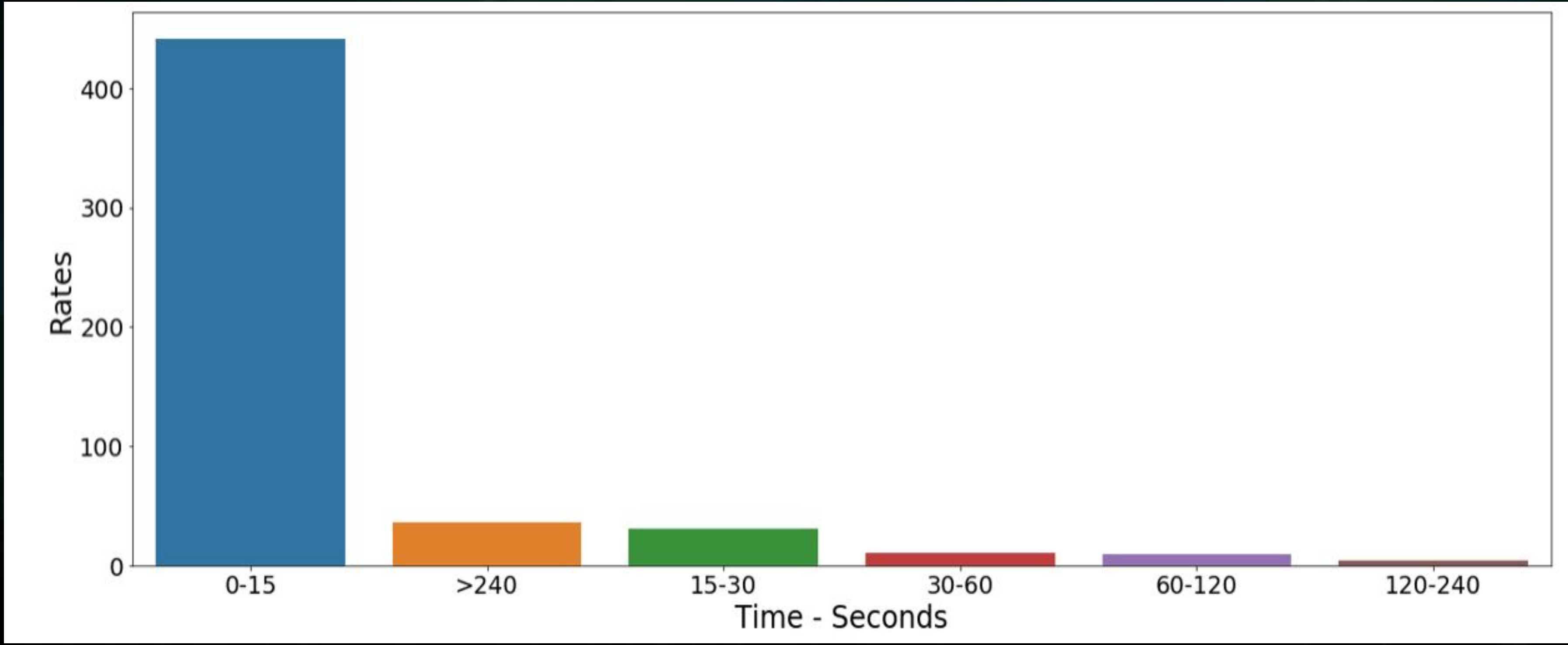
Add hemisphere column to ufo_date

```
hemis = []
for st_loc in ufo_data['latitude']:
    if st_loc >= 0 :
        hemis.append('Northern Hemisphere')
    else:
        hemis.append('Southern Hemisphere')
```

Percentage of UFO sightings in dependence of season and hemisphere



How many seconds was sighted?



- Encode variables

```
ufo_data['hemisphere'] = ufo_data['hemisphere'].cat.codes
```

```
ufo_data['season'] = ufo_data['season'].cat.codes
```

- Set train and test set

```
#test and train split using sklearn.model_selection
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size = 0.22, random_state = 1)
```

```
y_train.shape
```

```
(61449,)
```

- Algorithms: AdaBoostClassifier , SGDClassifier, RandomForestClassifier

- AdaBoostClassifier

```
print('accuracy score:', accuracy_score(y_test, pred_abc))
```

```
accuracy score: 0.3368718629204408
```

- SGDClassifier

```
print('accuracy score:', accuracy_score(y_test, pred_sgd))
```

```
accuracy score: 0.3170253274101425
```

- RandomForestClassifier

```
print('accuracy score:', accuracy_score(y_test, pred_rfc))
```

```
accuracy score: 0.33716033000634626
```

Evaluation of ML performance

<u>AdaBoostClassifier:</u>

- Medium accuracy score , slowest
- Each successive tree uses residuals of the previous tree

<u>SGDClassifier:</u>

- Lowest accuracy score
- Requires a number of hyperparameters

<u>RandomForestClassifier:</u>

- Best accuracy score, fastest
- Ensemble of many trees
- Strong predictive power

Textual analysis of Comments

Flying beer barrel shaped metallic object

Large...beautiful...and brighter than anything I've ever seen....How small I have felt since....

Bag of words

- Cleaning data, removing digits, non-letters, unicode
- Stemming, spellcheck, removing stop words

Count Vectorizer -> Matrix

```python
# Create the bag of words feature matrix
count = CountVectorizer(max_features=1000) #Max_feartures optional

bag_of_words = count.fit_transform(text_numpy)

# Show feature matrix
X = bag_of_words.toarray()
X.shape
```

```
(78848, 1000)
```

```python
# Create data frame, X = bag_of_words.toarray()
textdf = pd.DataFrame(X, columns=feature_names)
```

```python
textdf.head()
```

	about	above	accross	across	afb	after	afternoon	again	against	ago	...	yellow	yellowish	yes	york	you	young	zag	zagging	zig	zoomed
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Stemming and autocorrect

Stemming streamlines the different grammartic ways a word can be spelled.

NLTK library (Natural Language ToolKit), PorterStemmer module (different stemming modules exist)

Autocorrect Library, Spell module

```python
from nltk.stem.porter import PorterStemmer
from autocorrect import spell

stemmer = PorterStemmer()

for i in range(len(feature_names)):
    feature_names[i] = stemmer.stem(spell(feature_names[i]))

len(feature_names) #1000 after first run, 904 after the last run
```

```
908
```

```python
stemmer.stem("having") #Example
```

```
'have'
```

Removing stop words

Stopwords are common words such as “the”, “a”, an”, “in”

Frequent words with little value

NLTK Corpus package

```
import nltk
#nltk.download('stopwords') #- only need this once
from nltk.corpus import stopwords

for word in feature_names:
    if word in stopwords.words('english'):
        feature_names.remove(word)

print("Feature names:", len(feature_names))
print("Unique feature names:", len(unique(feature_names)))
```

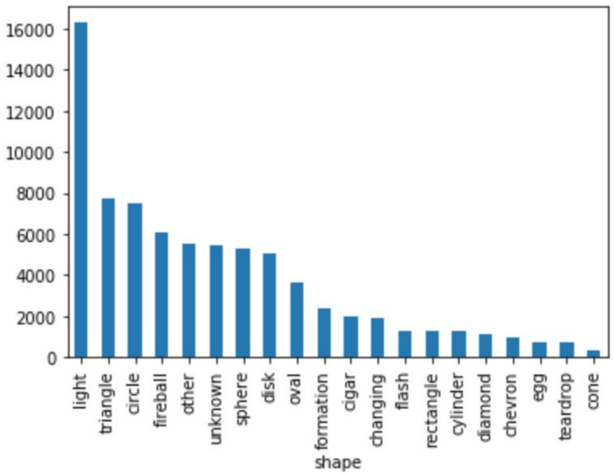
```
Feature names: 904
Unique feature names: 707
```

Textual analysis of description - Shape Classification

Classification from the words in comments: **Possible to predict shape?**

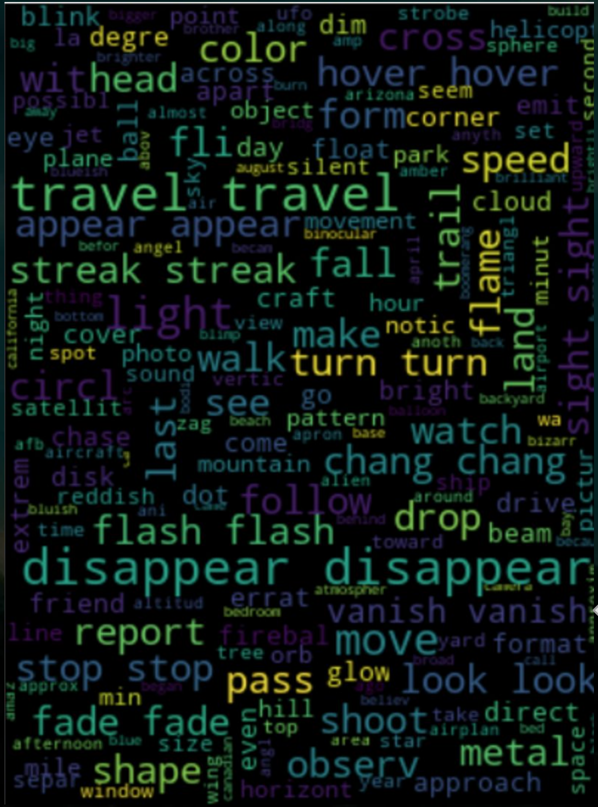
```
# shapes?
df.groupby('shape')['datetime'].count().sort_values(ascending=False).head(20).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a24b8c588>
```



WordCloud from words (1000 most frequent from corpus)

Before stemming and spellcheck



After stemming etc

Example and results

X: Bag of words.toArray
Y: Shapes

Tried classification algorithms:
(accuracy average of 10 runs)

- **Guassian Naive Bayes:**
  Accuracy ~ 0.03
- **Random Forest Classifier:**
  Accuracy ~0.42 (most accurate)
- **AdaBoostClassifier:**
  Accuracy ~0.24 (slowest)

SPLIT DATA IN TRAIN AND TEST

```
# split train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, shapes)
```

```
y_train.shape
```

```
(59136,)
```

AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
clf_abc = AdaBoostClassifier()
clf_abc.fit(X_train, y_train)

# Predict Class
abc_y_pred = clf_abc.predict(X_test)

abc_accuracy = accuracy_score(y_test, abc_y_pred)
```

```
abc_accuracy
```

```
0.23782467532467533
```

Reflection on results comments vs. shape

Comments are probably not the best prediction parameters for shape ...

- **Why does Random Forest give the best results?**
  - Parallel algorithm - trains all (random chosen) subsets/Decision Trees at the same time.
  - Uses **best guess** for each DT as a “total vote”

- **Why is Naive Bayes so much worse?**
  - Works best when classes are clearly separable - in this case, maybe not so much.

- **Why is AdaBoost the slowest?**
  - Sequential algorithms, that learns from the previous step.
  - **Why not better than random forest?** Not a clear connection between comment and shape.

Word2Vec

- Different models for word embedding in NLP
- Word list -> Vectors with lower dimension than Bag of Words

- Retains semantic meaning / context
- Can compute similar words and group related

```
sim_words = word2vec.wv.most_similar('cloud')
sim_words
```

```
[('north', 0.9484417438507080),
 ('disk', 0.9463559985160828),
 ('lights', 0.9463207721710205),
 ('large', 0.9457476139068604),
 ('ufo', 0.944894015789032),
 ('hovering', 0.9445918798446655),
 ('sky', 0.9445518255233765),
 ('observed', 0.9445128440856934),
 ('like', 0.9445018768310547),
 ('light', 0.9443306922912598)]
```

Doc2Vec

- Can group related documents by word processing
- Group sightings? (future work)