DATE-A-SCIENTIST

Machine Learning Fundamentals

Arian Mingo

2018-12-09

CONTENT

- Adding additional columns and Exploration
  - Zodiac signs (graphs)
  - Geolocation (graph)
- Classification
  - Living east from Great Plains
  - Zodiac signs and word counts
  - Income against others
- Regression
  - Income against others
  - Everything
- Conclusion
- What I learned - Next steps

QUESTIONS

- When looking at the data I was interested in the Zodiac sign feature, since I already made a study on regression between music interests and the four temperaments when I was in scool. Maybe you had bad results on the Zodiac sign data feature since you mixed up all, whether interested or not in that topic.
- I'm interested in finding differences between people in different geolocations. I will also explore that data.

ADDITIONAL COLUMNS AND EXPLORATION OF THE DATA - ZODIAC SIGNS -

I ORDERED THE SIGNS BY MONTH (AQUARIUS=JANUARY=1, PISCES=FEBRUARY=2, ETC) AND THEN MAPPED AND CREATED`SIGNCODE`

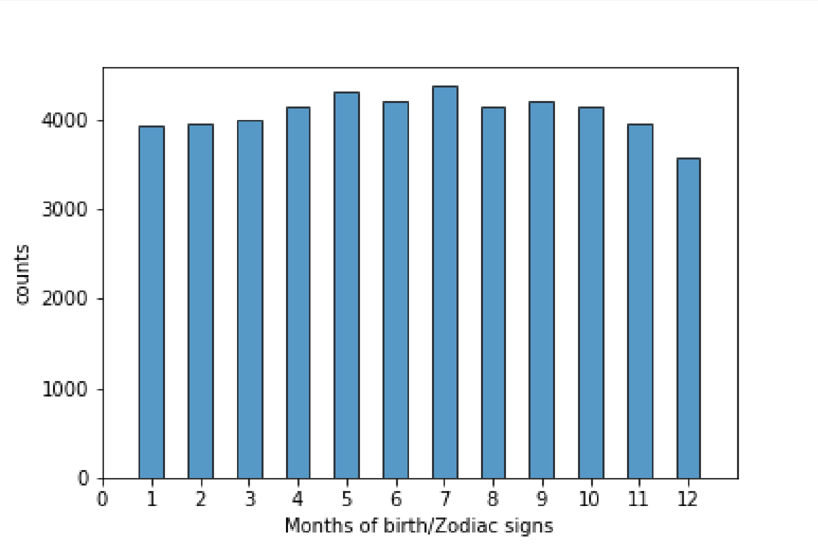
```
profiles.sign=profiles.sign.replace(np.nan, '', regex=True)

sign_mapping={
    'aquarius':1,
    'pisces':2,
    'aries':3,
    'taurus':4,
    'gemini':5,
    'cancer':6,
    'leo':7,
    'virgo':8,
    'libra':9,
    'scorpio':10,
    'sagittarius':11,
    'capricorn':12
}

profiles['signcode']=sign.map(sign_mapping)
```

I DEFINED 4 GROUPS OF INTEREST WEIGHT AND MAPPED THAT VALUES TO `SIGNCODE_W`.

```
#0: not interested in Zodiac signs
#1: just gave the value
#2: its fun to think about
#3: it matters a lot
regex_fun = re.compile('.*?fun to think about.*?')
regex_not = re.compile('.*?but it doesn.*?')
regex_lot = re.compile('.*?matters a lot.*?')

def get_weight(x):
    if regex_fun.match(x) is not None:
        return 2
    elif regex_lot.match(x) is not None:
        return 3
    elif regex_not.match(x) is not None:
        return 0
    else:
        return 1

signcode_w=[]
for i in profiles['sign']:
    signcode_w.append(get_weight(i))

profiles['signcode_w']=signcode_w
```

ADDITIONAL COLUMNS AND EXPLORATION OF THE DATA - ZODIAC SIGNS -

IN THIS GRAPH YOU SEE THAT MOST BIRTHS ARE IN SUMMER, SO PEOPLE USED TO MAKE CHILDREN BEFORE CHRISTMAS (RIGHT NOW)



THIS PLOT SHOWS, THAT ONLY FEW PEOPLE GIVE HIGH WEIGHT TO ZODIAC SIGNS. MOST OF THEM GIVE ONLY THE VALUE OR THINK ITS FUNNY

ADDITIONAL COLUMNS AND EXPLORATION OF THE DATA - GEOLOCATION -

- In the google API for geolocation I found a way to map the location feature to longitude and latitude on the world map.

```
#I ran this just first time and then I saved the data in CSV (It was translated within the $300 limit of the geolocation API ;-)
loc = profiles.location
lonlat = pd.DataFrame(columns={'lat','lng'})
for x in loc:
    maps_req = req.get("https://maps.google.com/maps/api/geocode/json?key=###myGeolocationAPIKey###&address="+x)
    maps=maps_req.json()
    coordinates=maps['results'][0]['geometry']['location']
    coordinates = pd.DataFrame([coordinates], columns=coordinates.keys())
    lonlat = lonlat.append([coordinates], ignore_index=True)
```

ADDITIONAL COLUMNS AND EXPLORATION OF THE DATA - GEOLOCATION -

- In the google API for geolocation I found a way to map the location feature to longitude and latitude on the world map. It took more than an hour to calculate each location

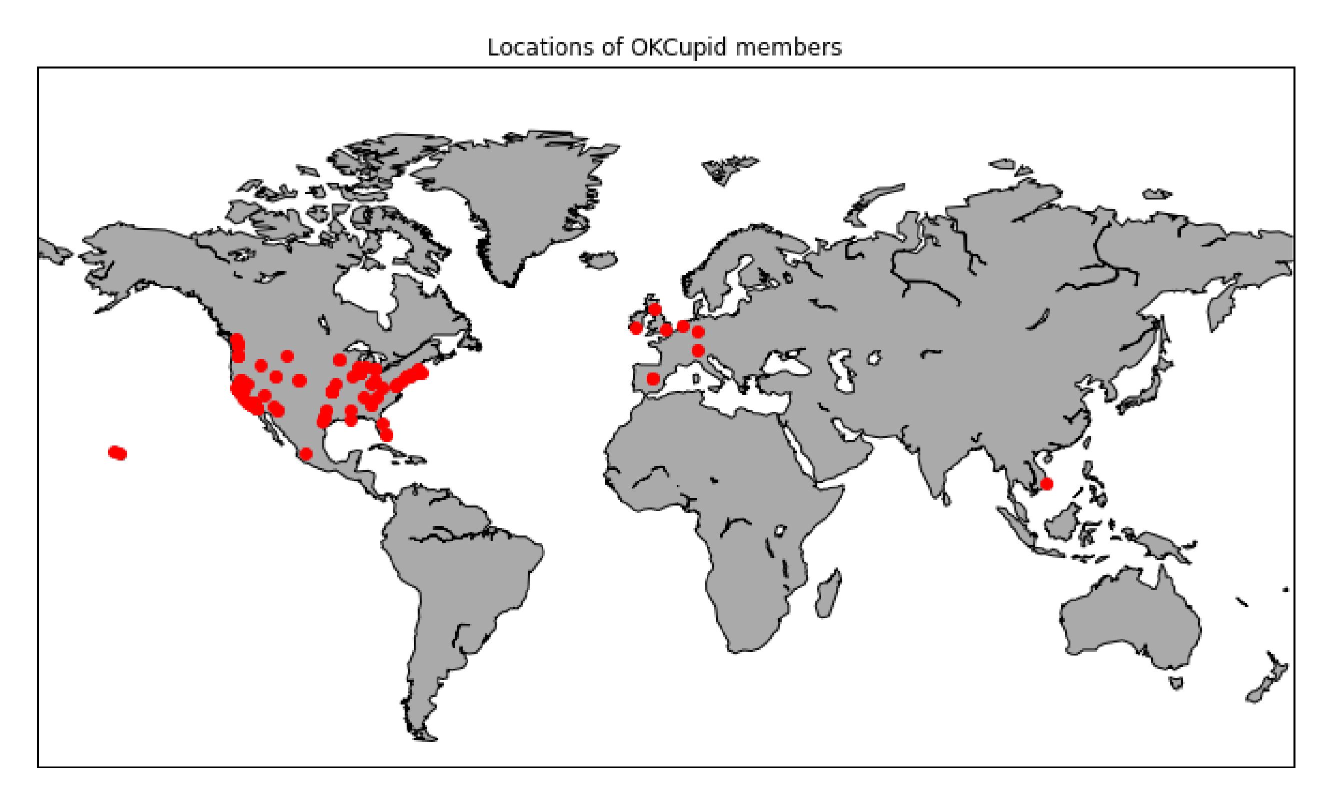
```
#I ran this just first time and then I saved the data in CSV (It was translated within the $300 limit of the geolocation API ;-)
loc = profiles.location
lonlat = pd.DataFrame(columns={'lat','lng'})
for x in loc:
    maps_req = req.get("https://maps.google.com/maps/api/geocode/json?key=###myGeolocationAPIKey###&address="+x)
    maps=maps_req.json()
    coordinates=maps['results'][0]['geometry']['location']
    coordinates = pd.DataFrame([coordinates], columns=coordinates.keys())
    lonlat = lonlat.append([coordinates], ignore_index=True)
```

ADDITIONAL COLUMNS AND EXPLORATION OF THE DATA
- GEOLOCATION -

- I imported Basemap from mpl_toolkits and plotted the data

```
from mpl_toolkits.basemap import Basemap
m = Basemap(projection='mill',llcrnrlat=-60,urcrnrlat=90, llcrnrlon=-180,urcrnrlon=180,resolution='c')
m.drawcoastlines()
m.fillcontinents(color='#AAAAAA',lake_color='#FFFFFF')

m.drawmapboundary(fill_color='#FFFFFF')
plt.title("Locations of OKCupid members")

lonlat = pd.read_csv("lonlat.csv")
profiles['lon']=lonlat.values[:, 0]
profiles['lat']=lonlat.values[:, 1]

x1, y1 = m(profiles['lon'].values.tolist(),profiles['lat'].values.tolist())
m.scatter(x1, y1, zorder=100, color='red')

plt.show()
```

ADDITIONAL COLUMNS AND EXPLORATION OF THE DATA
- GEOLOCATION -



CLASSIFICATION #1
LIVING EAST FROM GREAT PLAINS

- I define the border as longitude: -100,0°
- Map FALSE if <= -100 and TRUE if > -100
- Use KNN to classify
- Features to classify are
  - The word counts
  - The income
  - And the smoking counts

CLASSIFICATION #1
LIVING EAST FROM GREAT PLAINS

- First I wanted to separate east and west from Atlantic Ocean. But the result was that 9 out of 59946 are coming from Europe and Asia.
- Then I wanted to separate east and west from the Great Plains , but the result was 71 out of 59946 living in the east of the Great Plains
- I found out, that the longitude that separates all profiles is about -122.419, while the mean of all values is about -122.287
- Only few profile owners are not coming from near California Sacramento
- I think there is no sense in further investigation of the geolocation

```
profiles['east_of_great_plains']=profiles['lon'].apply(lambda x: x > -100)
profiles['east_of_great_plains'].value_counts()

print(profiles['lon'].mean())
```

```
-122.28704817987133
```

CLASSIFICATION #2
ZODIAC SIGN AND ESSAY WORDS

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

profiles_explore=profiles[['all_essays', 'signcode']].dropna()

train_data, test_data, train_labels, test_labels = \
                            train_test_split(profiles_explore['all_essays'], \
                            profiles_explore['signcode'], test_size=0.2, random_state=1)


counter = CountVectorizer()
counter.fit(profiles_explore['all_essays'])

train_counts = counter.transform(train_data)
test_counts = counter.transform(test_data)

classifier = MultinomialNB()
classifier.fit(train_counts, train_labels)
predictions = classifier.predict(test_counts)

print(accuracy_score(test_labels, predictions))
```

```
0.0840662712211
```

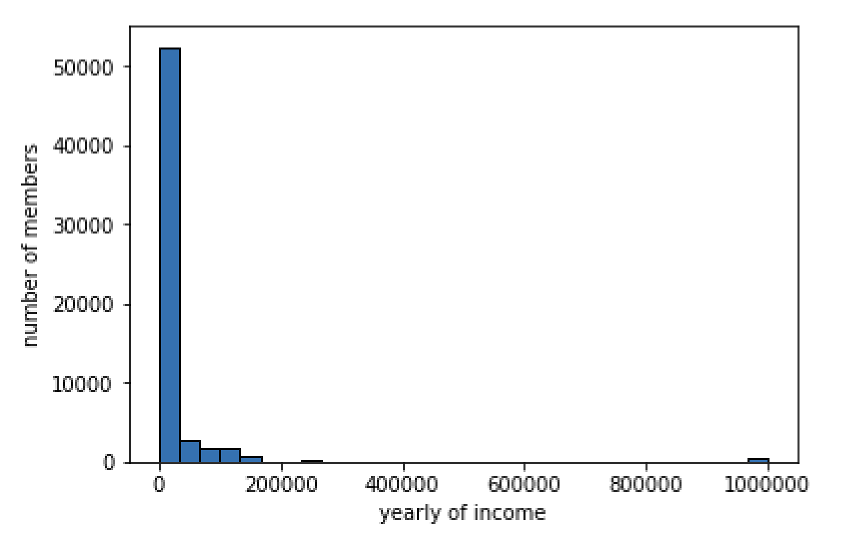
- I tried to find a regression with Naive Bayes Classifier, but I got approximately the same value as you: 0.084
- The script took more than 2 minutes (slow machine)

CLASSIFICATION #3
INCOME – AGAINST OTHERS

- From 'smokescode', 'drugscode', 'essay_len‘ and 'height‘ I got a pretty good regression to income
- But looking at the distribution I decided to calculate f1 score for safety. f1 had the same result. I‘m impressed
- The script took 47ms

```
feature_data_prescaled = \
    profiles[['smokescode', 'drugscode', 'essay_len', 'height', 'income']].dropna()

x = feature_data_prescaled.values
min_max_scaler = pre.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)

feature_data = pd.DataFrame(x_scaled, columns=feature_data_prescaled.columns)

train_data, test_data, train_labels, test_labels = \
    train_test_split(feature_data[['smokescode', 'drugscode', 'essay_len', 'height']], \
    feature_data_prescaled['income'], test_size=0.2, random_state=1)

classifier = MultinomialNB()
classifier.fit(train_data, train_labels)
predictions = classifier.predict(test_data)

accuracy=accuracy_score(test_labels, predictions)
print(accuracy)
```

```
0.800128593735648
```



REGRESSION #1
SAME SAMPLE AS BEFORE WITH REGRESSION

```
%%time
from sklearn.metrics import r2_score

feature_data_prescaled = \
    profiles[['smokescode', 'drugscode', 'essay_len', 'height', 'income']].dropna()

x = feature_data_prescaled.values
min_max_scaler = pre.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)

feature_data = pd.DataFrame(x_scaled, columns=feature_data_prescaled.columns)

train_data, test_data, train_labels, test_labels = \
    train_test_split(feature_data[['smokescode', 'drugscode', 'essay_len', 'height']], \
    feature_data_prescaled['income'], test_size=0.2, random_state=1)

regression = linear_model.LinearRegression()
regression.fit(train_data, train_labels)
predictions = regression.predict(test_data)

print(r2_score(test_labels, predictions))
%time
```

```
0.004504061562487283
Wall time: 0 ns
Wall time: 33 ms
```

- With the same data that gave me the 80% before I calculated the regression, which was very bad with r^2=0.0045
- This fore sure is cased by the bad distribution of the income

REGRESSION #2
EVERYTHING

```
%%time
from sklearn.metrics import r2_score

feature_data_prescaled = \
    profiles[['smokescode', 'drugscode', 'essay_len', 'signcode', 'signcode_w', 'height', 'income', 'lat', 'lon']].dropna()

x = feature_data_prescaled.values
min_max_scaler = pre.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)

feature_data = pd.DataFrame(x_scaled, columns=feature_data_prescaled.columns)

train_data, test_data, train_labels, test_labels = \
    train_test_split(feature_data[['smokescode', 'drugscode', 'signcode', 'signcode_w', 'height', 'income', 'lat', 'lon']], \
    feature_data['essay_len'], test_size=0.2, random_state=1)

regression = linear_model.LinearRegression()
regression.fit(train_data, train_labels)
predictions = regression.predict(test_data)

print(r2_score(test_labels, predictions))
%time
```

```
0.0067885064686815655
Wall time: 0 ns
Wall time: 46 ms
```

- Giving all the data in the model, that I mapped to numbers, gave me a bad result
- I calculated each against the others.
- No r^2>0.5 all less than 0.01
- This script ran 46ms

CONCLUSION

- My questions from the beginning were to explore the geolocation and the zodiac signs.
- The geolocation was focused on California. So I was not able to see significant differences between people from other states or countries.
- Also the zodiac signs was not as expacted. The only thing I was able to see from the data is that in summer there are more births than in winter.
- I would like to have more exact geolocations (maybe from mobile tracking)
- I need more knowledge about the people living there. There were only 9 people from the east side of the atlantic ocean, where I live.
- Next time I would explore more other data that is difficult to map to numbers, or combine different features where I know about a correlation, to one feature. For example body_type, diet and status.

WHAT I LEARNED - NEXT STEPS

- This was the first course I took part of in machine learning. The last two months I learned a lot. I also booked a further course at the near University and I want to study for masters degree with focus on mechatronics and data science.
- I learned some Python, that I didn‘t use before
- I tested Colaboration (google), microsoft azure, jupyther notebook on my own aws instance, anaconda on my local machine, june on my iPad…
- The main difficulty I didn‘t expect was to handle Python environment and packages espacially pandas. It took me a long time to make syntax run.