Date-A-Scientist

Machine Learning Fundamentals
Nicholas Payanoff
12/17/2018

Table of Contents

Exploring the Dataset

There were numerous features in the data. There were several multiple choice options, such as:

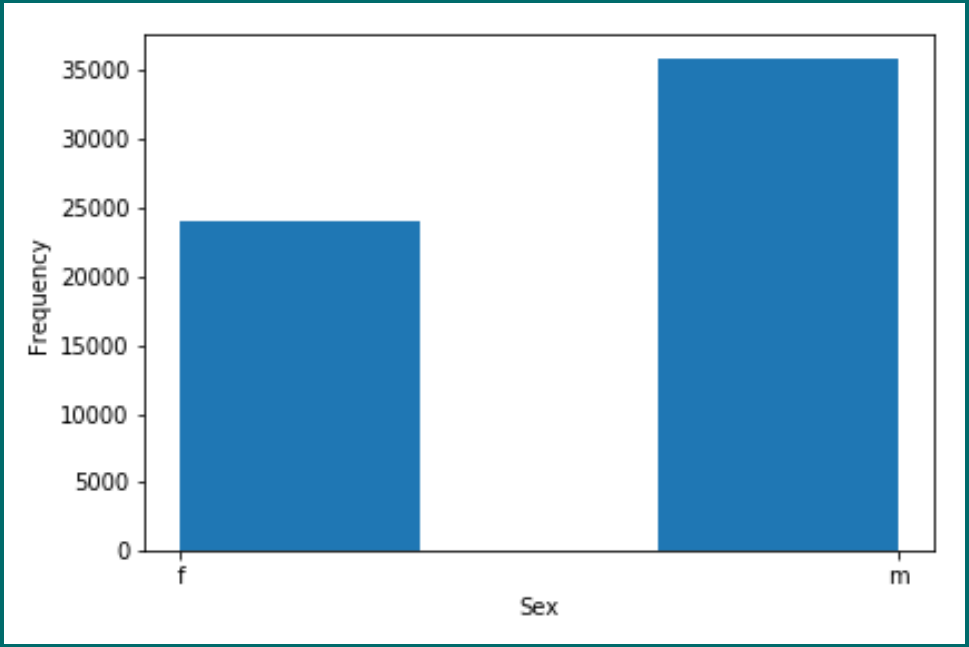
`Drinks, smokes, drugs, job, income, sign, religion`

There were also 9 short-answer essays, including topics such as a self-summary, favorite movies, and the ideal match for them

Exploring the Dataset

I wanted to look more in depth at a couple of features.

First, I wanted to look at the gender spread of the dataset:

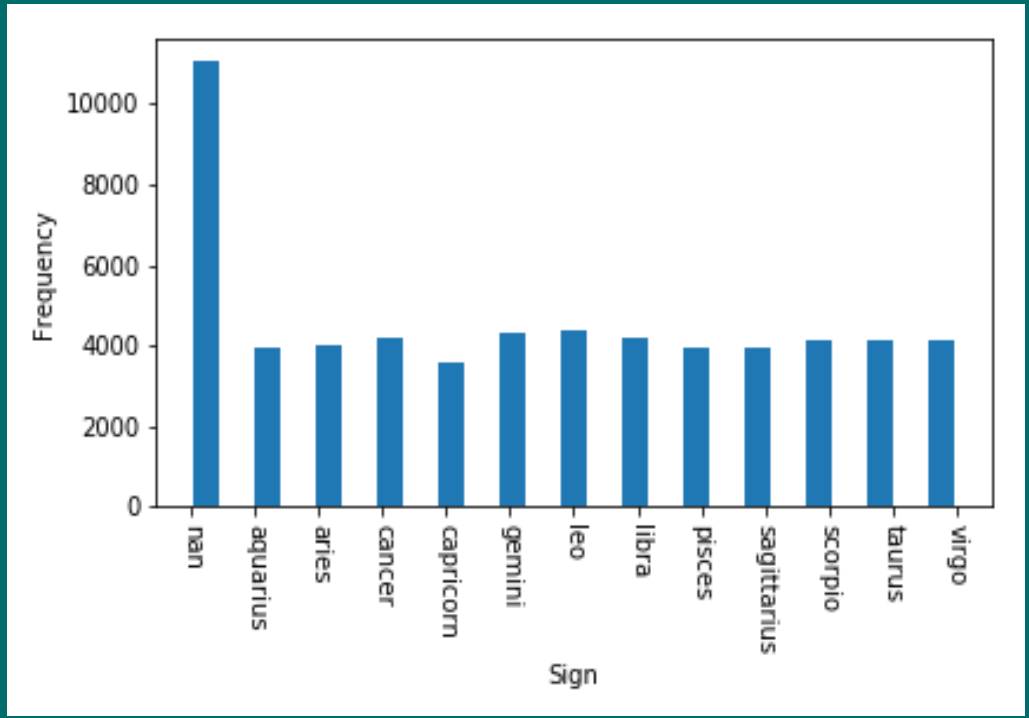


There are more males than females, but it is not a huge disparity (about 58% male users)

Exploring the Dataset

Astrological sign is a popular feature dating sites use to match people, as it is speculated that the sign one is born under has underlying effects on personality.

I was curious about the spread of these as well:



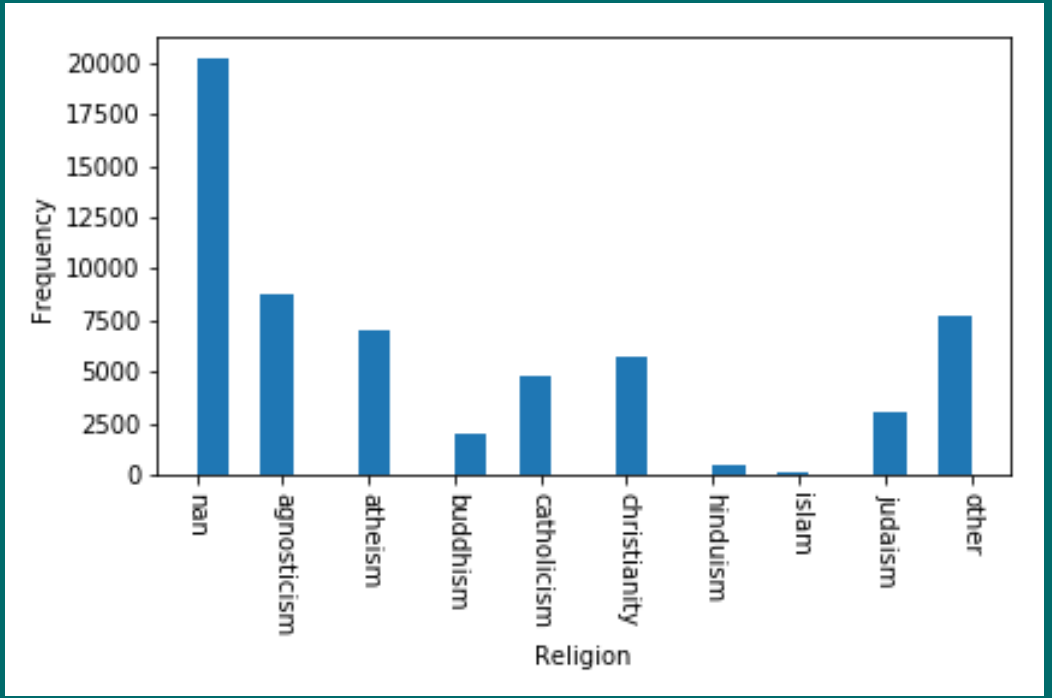
Aside from there being a large number who didn’t specify, it was relatively balanced

Exploring the Dataset

The final feature I looked at was religion:

Again, a LOT of ‘nan’.

It was interesting to note that some groups seem underrepresented, but this could be due to more dating sites that cater to specific religions (Jdate, Christian Mingle, etc)



Questions

I came up with 4 questions that interested me:

- Can `sex` be determined purely by the choice of words used in the essays?
- Can `ethnicity` be determined by word choice?
- Can `income` be accurately predicted from certain features?
- Can `ethnicity` be classified by certain behavioral features?

Preparing the Dataset

After exploring the data, there were some things I wanted to clean up. First, I needed to simplify the `religion` and `sign` features in order to better understand the spreads

`religion` and `sign` were more than just a choice. The site also allowed for the user to specify how serious these were to them. As I was not interested in that(at this time) I wanted to trim those out.

Preparing the Dataset

The entries looked something like this:

```
agnostic
judaism and very serious
christianity but not very serious
```

`sign` showed the same pattern. To simplify these features, I only needed the first word of the entry.

Preparing the Dataset

Extracting this was fairly straightforward. After importing the DataFrame as ‘`data`’, I used some `lambda` functions to add new columns with just the religion and sign of the user.

```
data['signs_trimmed'] = data.sign.apply(lambda x: str(x).split()[0])
data['religion_trimmed'] = data.religion.apply(lambda x: str(x).split()[0])
```

The entry was converted to a string, which was then split into a list. The first word (index 0) was then extracted and added to a new column

Preparing the Dataset

There were numerous features that would make more sense as numerical data. These would be mapped to numbers.

These all used the same method. I will show the steps for doing this using `ethnicity`, as there were a few extra steps involved.

Preparing the Dataset

The `ethnicity` feature gave users the option to specify multiple ethnicities, for example:

```
asian, native american
asian, black, native american, indian, pacific islander, hispanic / latin
middle eastern, pacific islander, hispanic / latin
```

To simplify this, I assumed the first listed ethnicity would be the one they most strongly associate with.
The procedure to map this to numbers is a combination of extracting the first word and mapping it to a number.

Preparing the Dataset

First I defined a map of the first words:

```
ethnicity_switch = {'other':0, 'white':1, 'black':2, 'asian':3, 'middle':4,
                    'pacific':5, 'indian':6, 'native':7, 'hispanic':8}
```

Then I mapped it to the first word, similar to `sign`:

```
data['ethnicity_code'] = data.ethnicity.apply(lambda x:
    ethnicity_switch.get(str(x).split()[0]))
```

I performed a similar process to mapping `job`, as well

Preparing the Dataset

A final column I added was one for the number of languages spoken, `num_speaks`. For this, I split the `speaks` entry and used the length of the resulting list:

```
data['num_speaks'] = data.speaks.apply(lambda x: len(str(x).split(',')))
```

Splitting by ‘,’ instead of ‘ ‘ allows for equal weight to be placed on whether the user specified fluency or not. This would make `english, spanish (fluently)` and `english, spanish` have the same length (`2`)

Classification

In order to tackle the question of classifying based on word choices, I used a Naive Bayes classifier.

All of the essays were combined, and the resulting string was transformed into a CountVectorizer. The model was then trained on this, using `sex` as a label:

```
counter = CountVectorizer()
counter.fit(all_essays)

NBtraining_data, NBvalidation_data, NBtraining_labels, NBvalidation_labels = \
train_test_split(all_essays, data['sex'], random_state=13)

NBtrain_counts = counter.transform(NBtraining_data)
NBvalidation_counts = counter.transform(NBvalidation_data)

NBclassifier = MultinomialNB()
NBclassifier.fit(NBtrain_counts, NBtraining_labels)
```

The model was then retrained using `ethnicity` as labels

Classification

To address classification by behaviors, I used a K-Nearest Neighbor classifier with the following features:

```
Kfeatures = ['smoke_code', 'drink_code', 'drugs_code', 'religion_code',
                'age', 'num_speaks', 'ethnicity_code']
```

`ethinicity_code` was passed initially to be included when ‘NaN’s were dropped.

Data for this classifier was normalized, as well.

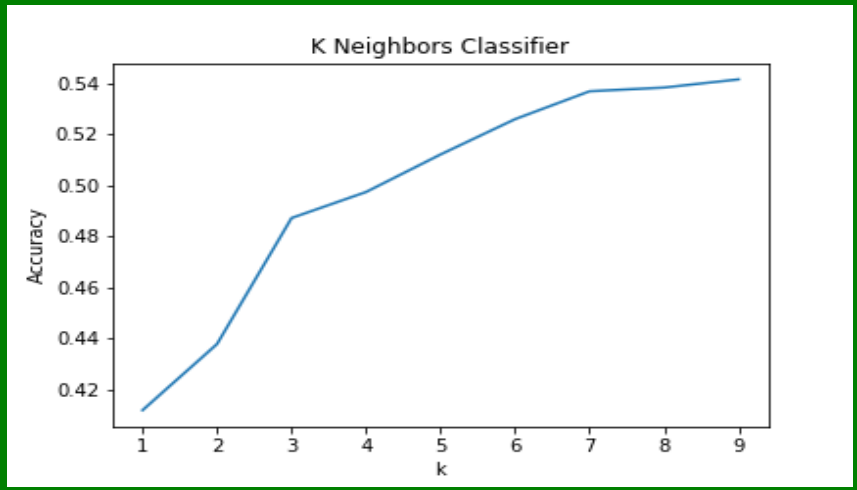
Classification

Comparing the 2 models:

283 ms ± 2.65 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Naive Bayes classifier: Sex
Score: 0.723027
Accuracy: 0.723027
Recall: 0.738472
Precision: 0.730244
F1: 0.721813

297 ms ± 816 µs per loop (mean ± std. dev. of 7 runs, 1 loop each)
Naive Bayes classifier accuracy: Ethnicity
Score: 0.553079
Accuracy: 0.553079
Recall: 0.150116
Precision: 0.226000
F1: 0.151038

508 ms ± 233 µs per loop (mean ± std. dev. of 7 runs, 1 loop each)
K Neighbors Classifier k = 5
Score: 0.512103
Accuracy: 0.512103
Precision: 0.257772
Recall: 0.177866
F1 Score: 0.191868



Classification

Naive Bayes was significantly faster then KNeighbors, this was most likely due to features used.

Classifying `sex` by word choice was fairly good, being better than guessing (which would be .5 chance. The precision, recall, F1 score were decent, as well.

Both NB and KNN classifying `ethnicity` performed poorly. While better then guessing in terms of accuracy (guessing would be .125), the precision, recall and F1 are very low.

NB was simpler to set up than KNN. Again, this was most likely due to the features used.

Regression

To tackle the question of predicting income from a set of features, I used 2 regression methods: LinearRegression and KNeighborsRegression.

Both models were trained on the same set of features, which were normalized:

```
LRfeatures = ['smoke_code', 'drink_code', 'drugs_code', 'signs_code',
              'religion_code', 'essays_len', 'age', 'job_code', 'num_speaks']
```

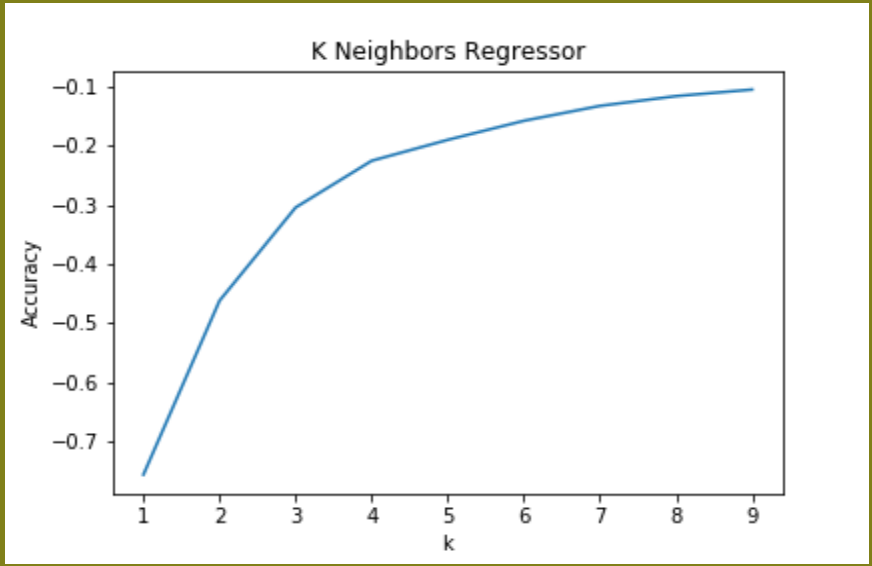
I was interested to see if certain behavioral features could predict income

Regression

Comparing the models:

```
1.33 ms ± 9.89 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```
Linear Regression Model
```
Score: 0.012924

11 ms ± 76.5 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```
KNeighbors Regressor
```
Score for k=5: -0.189292
```



Regression

LinearRegression was notably faster than KNeighbors at 5 neighbors(nearly 10 times)

Both models performed poorly, both with accuracies less than 0.25.

KNeighbors was the worst, with a negative accuracy score.

Conclusion

Going back to the questions:
Can `sex` be determined purely by the choice of words used in the essays? **Yes**
Can `ethnicity` be determined by word choice? **Not quite**
Can `income` be accurately predicted from certain features? **No**
Can `ethnicity` be classified by certain behavioral features? **Not quite**

With all of the scores being around 0.70, classifying `sex` by word choice can be performed. Some tweaks can make it better (perhaps some other essay subjects)

`Ethnicity` does not seem to be as easy to classify based on the parameters I used. Using the actual languages spoken may have improved it significantly.

`Income` does not seem to be something that varies linearly based on behaviors used for the model.

Conclusion

Originally, I wanted to see if it would be possible to use location for a feature, but the data was a bit one-sided...

california	**59855**
new york	17
illinois	8
massachusetts	5
oregon	4
texas	4
michigan	4
florida	3
arizona	3
colorado	2
minnesota	2
ohio	2
hawaii	2
utah	2
spain	2
district of columbia	2
washington	2
virginia	2
united kingdom	2
georgia	2
idaho	1
connecticut	1
nevada	1
missouri	1
west virginia	1
north carolina	1
louisiana	1
switzerland	1
pennsylvania	1
mexico	1
ireland	1
canada	1
netherlands	1
vietnam	1
wisconsin	1
tennessee	1
rhode island	1
new jersey	1
montana	1
germany	1
mississippi	1

Conclusion

Next steps:

- Tweak the `ethnicity` model to include actual languages spoken.
- Try to improve `sex` classification by adding more features.
- Look more into what data is present to see if `income` can be more reliably modeled, perhaps by adding `education` levels or `sex` to the feature list.

While the current dataset has a wealth of information in it, it would be nice to gather one with more geographic variety in it

END