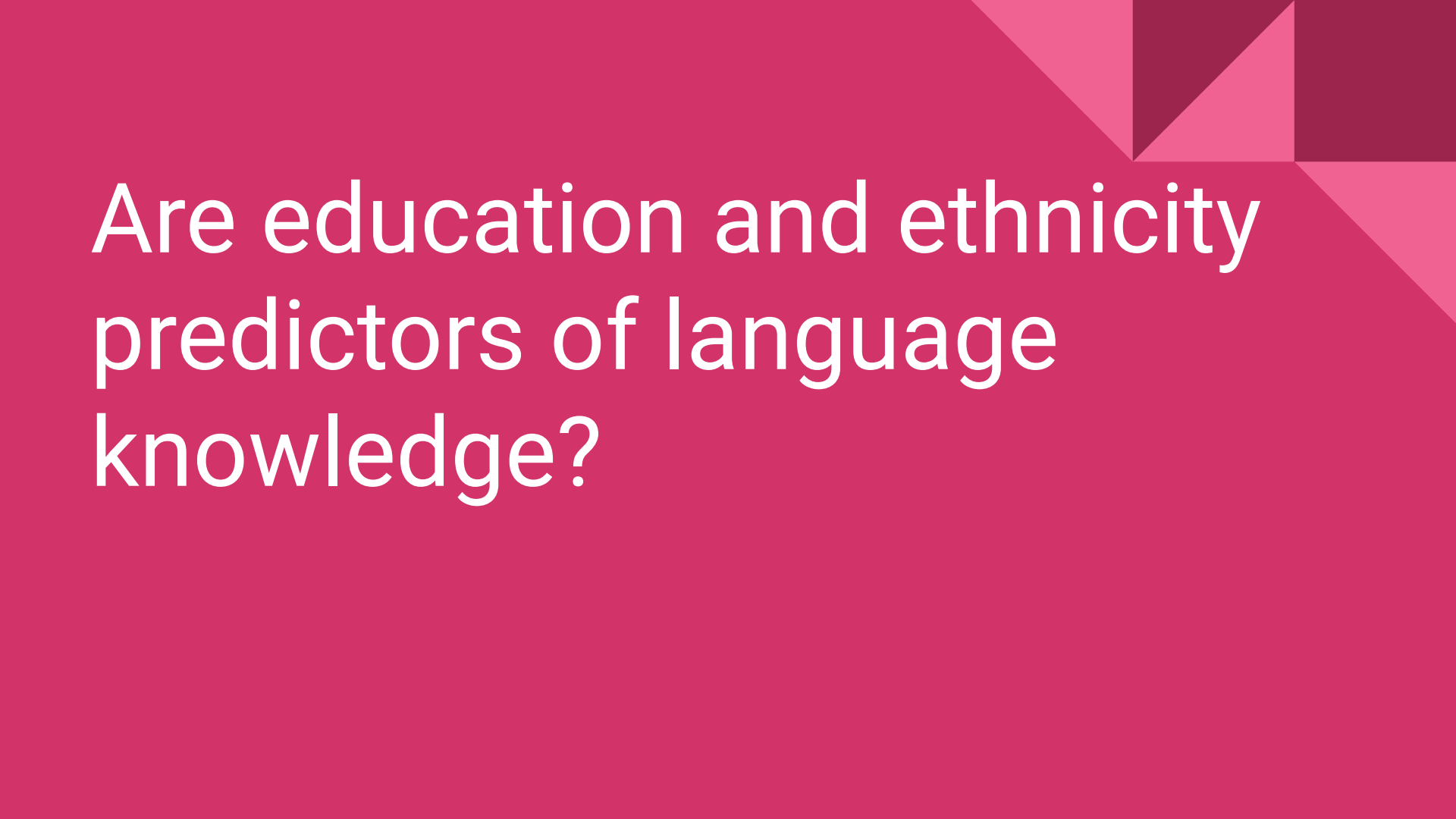


# Predicting language knowledge

Machine Learning Fundamentals

by Lucas Vinzon



Are education and ethnicity  
predictors of language  
knowledge?

# The Experiment

# Exploring the Dataset

Based on OkCupid profiles' ethnicities, education and languages spoken, a few considerations were made:

- self-declarative data
  - fluency spectrum
  - education and languages grading
  - ethnicity x plurality
-

# Self-declarative data

1. Is the data generated by participants stimulated to overestimate their qualities? Is education and language proficiency both cases prone to overestimation? How often would people lie?
2. Should a person exposed to multiple languages have a higher ethnicity awareness?
3. Should a person with multiple backgrounds be prone to having multiple languages?

## Fluency Spectrum

1. Are people overestimating their fluency? Are fluent people underestimating their fluency? Dunning-Kruger Effect
2. How should we evaluate the number of languages a person speaks?
3. Should there be a grading for beginners and fluent people?

# Education and Language Grading

1. How should we grade people's education and their language skills?
2. For recent immigrants, who would most likely speak at least two languages, they would also have a much lower education level. Is there a strategy to tackle this?
3. There are ethnicities which translate into a higher chance of having a plural linguistic background and others that do not. How should this be evaluated?

## Ethnicity x Plurality

1. People could freely determine from 1 to 5 ethnicities on their own will. How could we evaluate their ethnicity?
2. How should people's ethnicities be evaluated numerically?
3. Should there be an evaluation on how plural a person is instead of their ethnicity?

# Augmenting the Dataset

Our decisions on augmenting the dataset with labels to create a Machine Learning model which could predict a person's spoken languages:

1. ["plural\_code"] : Instead of having their ethnicity we created a column which evaluates to 0 if the person declared themselves as white (the least plural), to 1 if the person declared as black, asian or hispanic/latin (some plurality) and 2 to everything else (most plural and ethnically aware).
2. ["speaks\_count"] : we chose to ignore the person's self-declared fluency and went with a count on how many languages the person declared to speak.
3. ["education\_code"] : We chose to grade people on their graduation level: from 0 (college completed) to 3 (Ph.D completed), considering if they completed a certain level of education. If still on the process of completion, last completion was considered.

We are aware of how these three labels could oversimplify, mislabel and underrepresent several cases. Cleanup on speaks\_count and education were necessary. First, We'll try to perform a Logistic Regression.

# First Attempt

```
df['plural code'] = df.ethnicity.apply([lambda x: 0 if (x == 'white') else 1 if (x == 'black') | (x == 'asian') | (x == 'hispanic / latin') else 2]).replace(np.nan, 0, regex=True)
df['speaks count'] = df['speaks'].str.replace('\(([^\]]+)\)', '', case=False).str.replace(' ', '').replace(np.nan, '', regex=True).str.split(",").apply([lambda x: len(x)]).replace(np.nan, 1, regex=True)
df['education_code'] = df['education'].map({'graduated from college/university': 1, 'graduated from masters program': 2, 'working on college/university': 0, 'working on masters program': 1, 'graduated from two-year college': 1, 'graduated from high school': 0, 'graduated from ph.d program': 3, 'graduated from law school': 1, 'working on two-year college': 0, 'dropped out of college/university': 0, 'working on ph.d program': 2, 'college/university': 1, 'graduated from space camp': 1, 'dropped out of space camp': 0, 'graduated from med school': 1, 'working on space camp': 0, 'working on law school': 0, 'two-year college': 1, 'working on med school': 0, 'dropped out of two-year college': 0, 'dropped out of masters program': 1, 'masters program': 1, 'dropped out of ph.d program': 2, 'dropped out of high school': 0, 'high school': 0, 'working on high school': 0, 'space camp': 1, 'ph.d program': 3, 'law school': 1, 'dropped out of law school': 0, 'dropped out of med school': 0, 'med school': 1}).replace(np.nan, 0, regex=True)
```

```
features = df[['plural code', 'education code']].values
labels = df['speaks count']
```

```
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2)
```

```
model = LogisticRegression().fit(x_train, y_train)
print(np.unique(y_test, return_counts=True))
predict = model.predict(x_test)
print(np.unique(predict, return_counts=True))
```



# Results

From our first round we had the following results:

actual labels	[1, 2, 3, 4, 5]
actual labels respective counts:	[5815, 3428, 1721, 690, 336]

predicted labels:	[1, 2]
predicted labels respective counts:	[11934, 56]

From these results, we can safely say that we have failed to conform to any sort of prediction. We have to perform find a different strategy:

Use different models? Label differently? Have a different approach?

# K Neighbors Classifier

First we tried a Linear Regression which had no success then we tried KNeighborsClassifier:

```
kneighbors: 1 (array([1, 2, 3, 5], dtype=int64), array([4396, 6110, 1203, 281], dtype=int64))
kneighbors: 2 (array([1, 2], dtype=int64), array([8484, 3506], dtype=int64))
kneighbors: 3 (array([1, 2], dtype=int64), array([8481, 3509], dtype=int64))
kneighbors: 4 (array([1, 2, 3], dtype=int64), array([8704, 3228, 58], dtype=int64))
kneighbors: 5 (array([1, 2], dtype=int64), array([8481, 3509], dtype=int64))
kneighbors: 6 (array([1, 2, 3], dtype=int64), array([8423, 3509, 58], dtype=int64))
kneighbors: 7 (array([1, 2, 3], dtype=int64), array([7220, 4712, 58], dtype=int64))
kneighbors: 8 (array([1, 2, 3], dtype=int64), array([8423, 3509, 58], dtype=int64))
kneighbors: 9 (array([1, 2, 3], dtype=int64), array([7698, 3018, 1274], dtype=int64))
kneighbors: 10 (array([1, 2, 3], dtype=int64), array([8423, 3018, 549], dtype=int64))
kneighbors: 11 (array([1, 2, 3], dtype=int64), array([8423, 3018, 549], dtype=int64))
kneighbors: 12 (array([1, 2, 3], dtype=int64), array([8423, 3509, 58], dtype=int64))
kneighbors: 13 (array([1, 2], dtype=int64), array([8948, 3042], dtype=int64))
kneighbors: 14 (array([1, 2, 3], dtype=int64), array([8457, 3042, 491], dtype=int64))
kneighbors: 15 (array([1, 2], dtype=int64), array([10036, 1954], dtype=int64))
kneighbors: 16 (array([1, 2], dtype=int64), array([11160, 830], dtype=int64))
kneighbors: 17 (array([1, 2, 3], dtype=int64), array([9432, 2500, 58], dtype=int64))
kneighbors: 18 (array([1, 2, 3], dtype=int64), array([11011, 921, 58], dtype=int64))
kneighbors: 19 (array([1, 2, 3], dtype=int64), array([9887, 2045, 58], dtype=int64))
```

*actual*

```
(array([1, 2, 3, 4, 5], dtype=int64), array([5772, 3527, 1741, 644, 306], dtype=int64))
```

looping through a range of 1-19 kneighbors, there is some variance.  
Are people grouped somehow? Does performing KNeighbors or KMeans will have better performance?

The following is the best performing k-neighbors (4) confusion matrix.  
Not very good.

[[3695	1483	664	30	0]
[2362	763	303	29	0]
[1243	303	141	29	0]
[468	99	50	10	0]
[235	39	39	5	0]]

Everything indicates that no correlation, bad labeling or bad feature labeling.  
Let's retry

# Reconsidering education:

What if we consider that education is a get in thing, where dropouts should still be grouped with who completed such education?

## KNeighborsClassifier:

```
kneighbors: 5 (array([1, 2, 3], dtype=int64), array([9316, 2247, 427],  
(array([1, 2, 3, 4, 5], dtype=int64), array([5887, 3422, 1712, 641, 32  
[[4951 777 159 0 0]  
[2445 866 111 0 0]  
[1222 392 98 0 0]  
[ 461 149 31 0 0]  
[ 237 63 28 0 0]]
```

## LogisticRegression:

```
predict (array([1, 2], dtype=int64), array([11479, 511],  
actual (array([1, 2, 3, 4, 5], dtype=int64), array([5903,  
[[5720 183 0 0 0]  
[3280 140 0 0 0]  
[1603 105 0 0 0]  
[ 615 53 0 0 0]  
[ 261 30 0 0 0]]
```

Or considering dropping out or in the process of completing an intermediate step?

## KNeighborsClassifier:

```
[[4701 559 496 7 0]  
[2643 455 354 12 0]  
[1314 222 186 8 0]  
[ 483 90 99 1 0]  
[ 268 49 43 0 0]]
```

## LogisticRegression:

```
predict (array([1, 2], dtype=int64), array([11831, 159],  
actual (array([1, 2, 3, 4, 5], dtype=int64), array([5851, 3  
[[5807 44 0 0 0]  
[3356 49 0 0 0]  
[1699 33 0 0 0]  
[ 644 17 0 0 0]  
[ 325 16 0 0 0]]
```

# !Disconsidering education:

What if education is actually misleading?

**KNeighborsClassifier:**

```
kneighbors: 7 (array([1, 3], dtype=int64), array([9916, 2074], dtype=int64), array([1, 2, 3, 4, 5], dtype=int64), array([5861, 3447, 1722, 653, 307], dtype=int64))
[[5096  0 765  0  0]
 [2605  0 842  0  0]
 [1389  0 333  0  0]
 [ 560  0  93  0  0]
 [ 266  0  41  0  0]]
```

**LogisticRegression:**

```
predict (array([1], dtype=int64), array([11990], dtype=int64))
actual (array([1, 2, 3, 4, 5], dtype=int64), array([5861, 3447, 1722, 653, 307], dtype=int64))
[[5861  0  0  0  0]
 [3447  0  0  0  0]
 [1722  0  0  0  0]
 [ 653  0  0  0  0]
 [ 307  0  0  0  0]]
```

It seems like with *less* information we have a worse performance...

Aha!

let's use education (giving the model the most information we can) again but add more features!

What would indicate a plural background?

Religion?

Could age have an impact on the prediction?

# Better!

By implementing  
['religion\_code'] and ['age'], we  
had better results!

	precision	recall	f1-score	support
1	0.50	0.97	0.66	5880
2	0.30	0.05	0.08	3443
3	0.09	0.00	0.00	1719
4	0.00	0.00	0.00	639
5	0.00	0.00	0.00	309
accuracy			0.49	11990
macro avg	0.18	0.20	0.15	11990
weighted avg	0.34	0.49	0.34	11990

And what if we use Random Forest  
Classifier?

	precision	recall	f1-score	support
1	0.51	0.87	0.64	5770
2	0.37	0.21	0.27	3467
3	0.22	0.02	0.04	1781
4	0.16	0.01	0.01	631
5	0.04	0.00	0.01	341
accuracy			0.48	11990
macro avg	0.26	0.22	0.19	11990
weighted avg	0.40	0.48	0.39	11990

Looks quite interesting, it performed way better for  
people who spoke more than one language (despite  
that it still performs quite badly).

---

# Iterating Through Features as Labels

**What if we are actually trying to predict a label which is hard to predict but a good predictor?**

**We attempted to predict ['speaks\_count'] but let's try to use it as a label and use one of the features as a label to predict!**

**[['religion\_code', 'plural\_code', 'education\_code', 'age']]**

**From those columns, the one that had the best performance was...**

# ['plural\_code'] !

It is a result which seems quite intuitive. It is easier to make a prediction between people of distinct backgrounds if they speak a different amount of languages, but it is harder to make a prediction on how many languages they speak based on if they are self declared racial background.

It still performs pretty badly.

```
[[5864 258 513]
 [1589 261 241]
 [2576 248 440]]
```

	precision	recall	f1-score	support
0	0.58	0.88	0.70	6635
1	0.34	0.12	0.18	2091
2	0.37	0.13	0.20	3264
accuracy			0.55	11990
macro avg	0.43	0.38	0.36	11990
weighted avg	0.48	0.55	0.48	11990