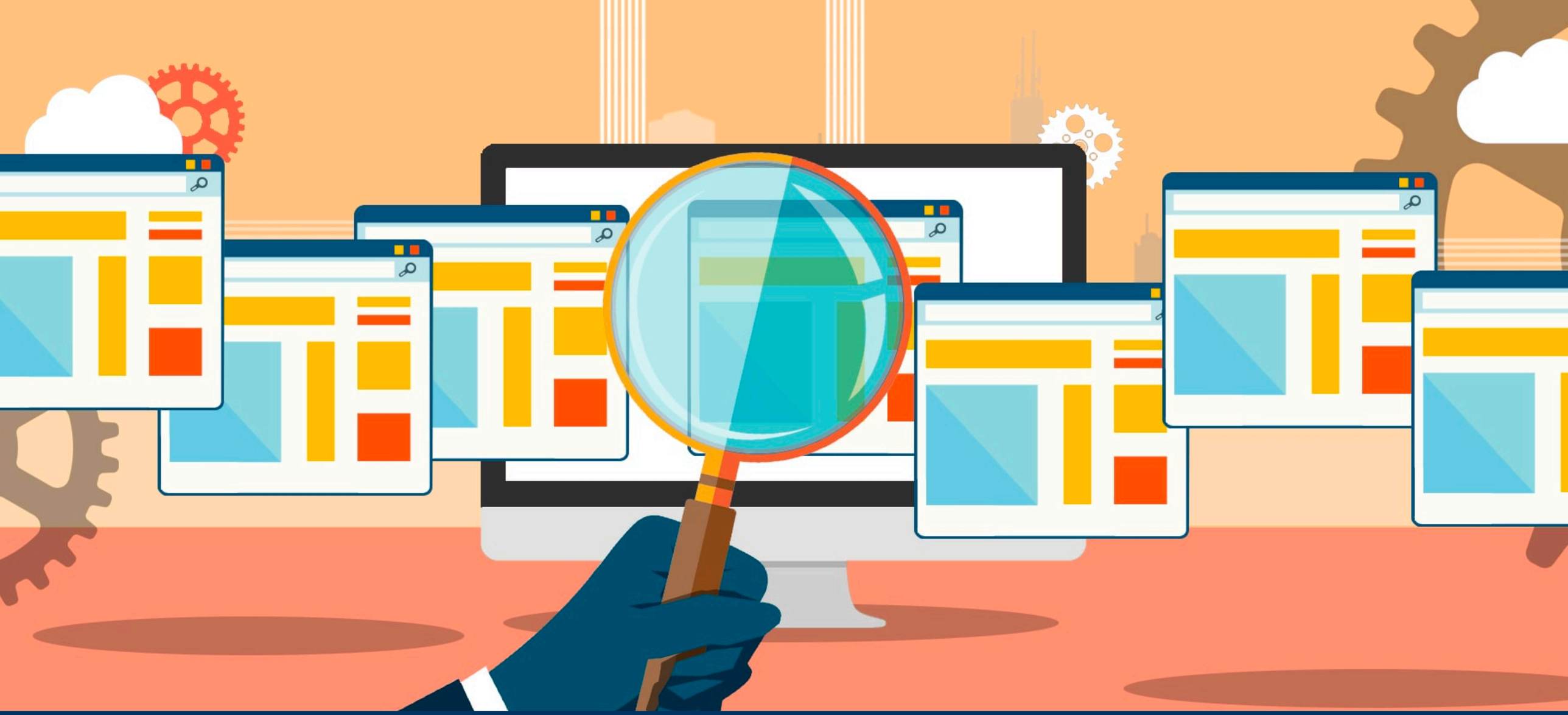


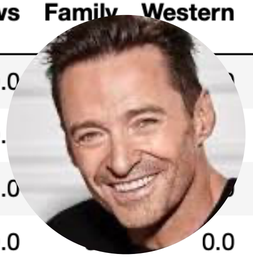
Data Hygiene and Cleaning

INST414 - Data Science Techniques

Motivating Question: What does it mean for two actors to be similar?

```
actor_genre_df.head(20)
```

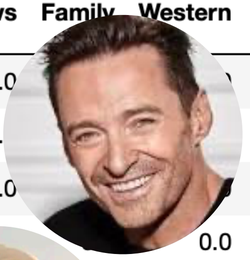
	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0			0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
nm0462648	1.0	0.0	1.0	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
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

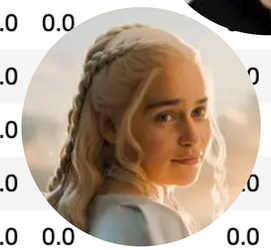


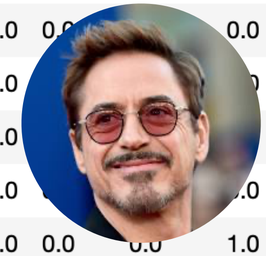
```
In [19]: actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0			0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0			0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0				0.0	0.0
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0					
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0						0.0					
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0						0.0	0.0		0.0		
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0						0.0	0.0		0.0		
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0						0.0				0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0						0.0				0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0							0.0				0.0	0.0
nm0001379	0.0	0.0	0.0	0.0			0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
nm0462648	1.0	0.0	1.0	0.0			0.0	0.0	2.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0000953	6.0	0.0	0.0	1.0			0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								







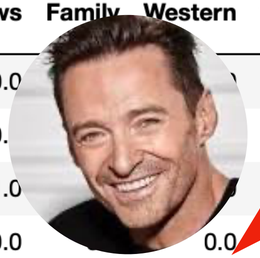
Are you sure?
How do you know no one is missing?

Who is closest to Hugh Jackman?

In [19]:
```
actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0			
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0



Cosine Similarity

In [21]:
```
#Printing the top ten most similar actors to our target
for similar_actor_id, similar_genre_score in sorted(query_distances, key=lambda x: x[1], reverse=False)[:10]:
    similar_actor = actor_genre_map[similar_actor_id]
    print(similar_actor_id, actor_name_map[similar_actor_id], similar_genre_score)
```

```
nm0413168 Hugh Jackman 0.0
nm3592338 Emilia Clarke 0.023038353871383088
nm1663205 Sharlto Copley 0.03680695049665894
nm0000375 Robert Downey Jr. 0.04015740464249473
nm3772243 Theo James 0.04358329306959263
nm0262635 Chris Evans 0.04805589651668307
nm0881631 Karl Urban 0.05508543978437186
nm0159789 Hayden Christensen 0.060195990212304484
nm1517976 Chris Pine 0.07953201413353528
nm1475594 Channing Tatum 0.08366839174462504
```

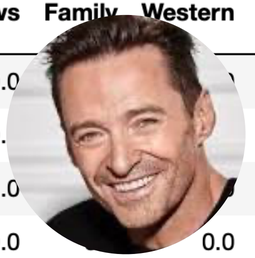
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								

Who is closest to Hugh Jackman?

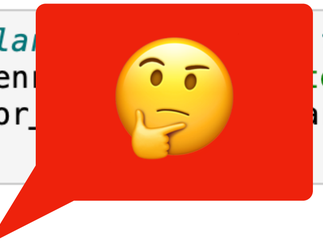
In [19]:
```
actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0		0.0	0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0



In [25]:
```
#Printing the top ten most similar                    target
for similar_actor_id, similar_genre                  ted(query_distances, key=lambda x: x[1], reverse=False)[:10]:
    print(similar_actor_id, actor_                   ar_actor_id], similar_genre_score)
```



```
nm0413168 Hugh Jackman 0.0
nm04131681 Hugh M. Jackman 0.0
nm3592338 Emilia Clarke 0.023038353871383088
nm1663205 Sharlto Copley 0.03680695049665894
nm0000375 Robert Downey Jr. 0.04015740464249473
nm3772243 Theo James 0.04358329306959263
nm0262635 Chris Evans 0.04805589651668307
nm0881631 Karl Urban 0.05508543978437186
nm0159789 Hayden Christensen 0.060195990212304484
nm1517976 Chris Pine 0.07953201413353528
```

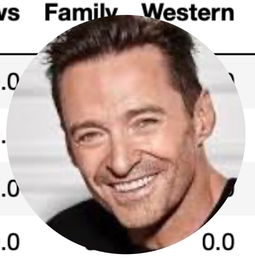
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								

Who is closest to Hugh Jackman?

In [19]:
```
actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0			0.0
nm0413168	8.0	3.0	6.0	140	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
nm0462648	1.0	0.0	1.0	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
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								

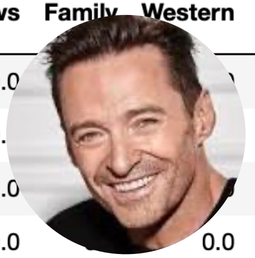


Is 140 a reasonable number?

In [19]:
```
actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0			0.0
nm0413168	8.0	3.0	6.0	1,400	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
nm0462648	1.0	0.0	1.0	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
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								



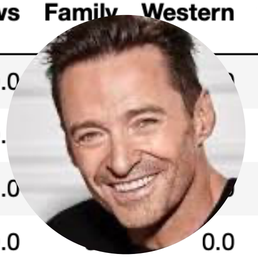
What about 1,400?

Is 1,400 reasonable?

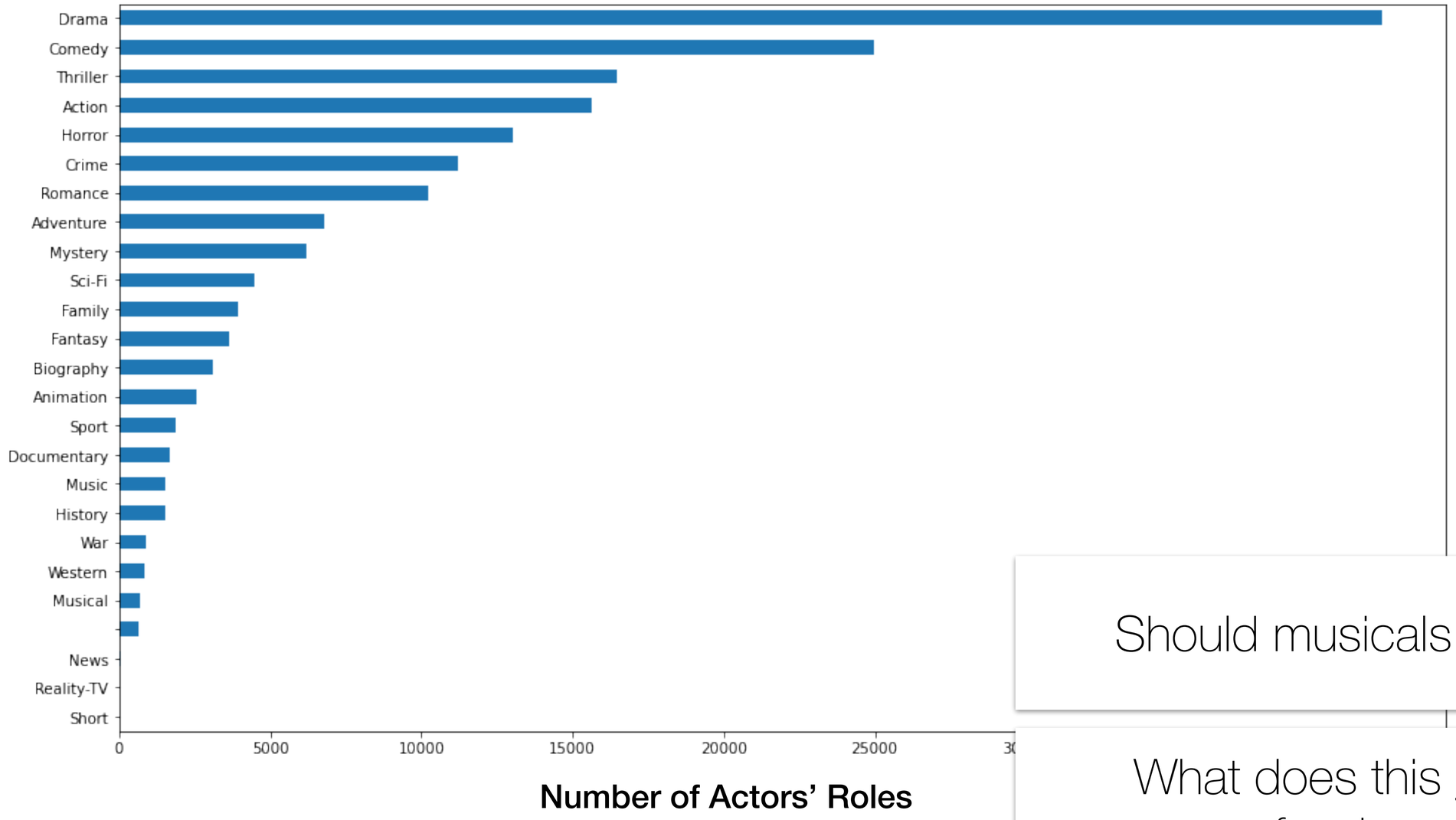
In [19]:
```
actor_genre_df.head(20)
```

Out[19]:

	Comedy	antasy	Romanc	Action	rime	Adventure	Mystery	Thriller	Drama	B ography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0		0.0	0.0
nm0413168	8.0	3.0	6.	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0
nm0697338	1.0	0.0	0.	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm1300519	1.0	0.0	0.	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0940707	1.0	0.0	0.	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0625977	1.0	0.0	0.	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
nm0462648	1.0	0.0	1.	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
nm0000953	6.0	0.0	0.	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001782	0.0	0.0	0.	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0005077	1.0	0.0	0.	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.	1.0	0.0	1.0	0.0	0.0	0.0								



Are these the most important genres?



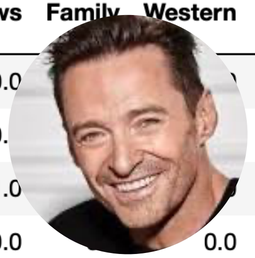
Should musicals be so rare?

What does this **imbalance** mean for dramas vs. sci-fi?

In [19]:
```
actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0		0.0	0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0



In [21]:
```
#Printing the top ten most similar actors to our target
for similar_actor_id, similar_genre_score in sorted(query_distances, key=lambda x: x[1], reverse=False)[:10]:
    similar_actor = actor_genre_map[similar_actor_id]
    print(similar_actor_id, actor_name_map[similar_actor_id], similar_genre_score)
```

```
nm0413168 Hugh Jackman 0.0
nm3592338 Emilia Clarke 0.023038353871383088
nm1663205 Sharlto Copley 0.03680695049665894
nm0000375 Robert Downey Jr. 0.04015740464249473
nm3772243 Theo James 0.04358329306959263
nm0262635 Chris Evans 0.04805589651668307
nm0881631 Karl Urban 0.05508543978437186
nm0159789 Hayden Christensen 0.060195990212304484
nm1517976 Chris Pine 0.07953201413353528
nm1475594 Channing Tatum 0.08366839174462504
```

Cosine distance between Hugh and other actors…

nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

In [21]:
```
#Printing the top ten most similar actors to our target
for similar_actor_id, similar_genre_score in sorted(query_distances, key=lambda x: x[1], reverse=False)[:1
    similar_actor = actor_genre_map[similar_actor_id]
    print(similar_actor_id, actor_name_map[similar_actor_id], similar_genre_score)
```

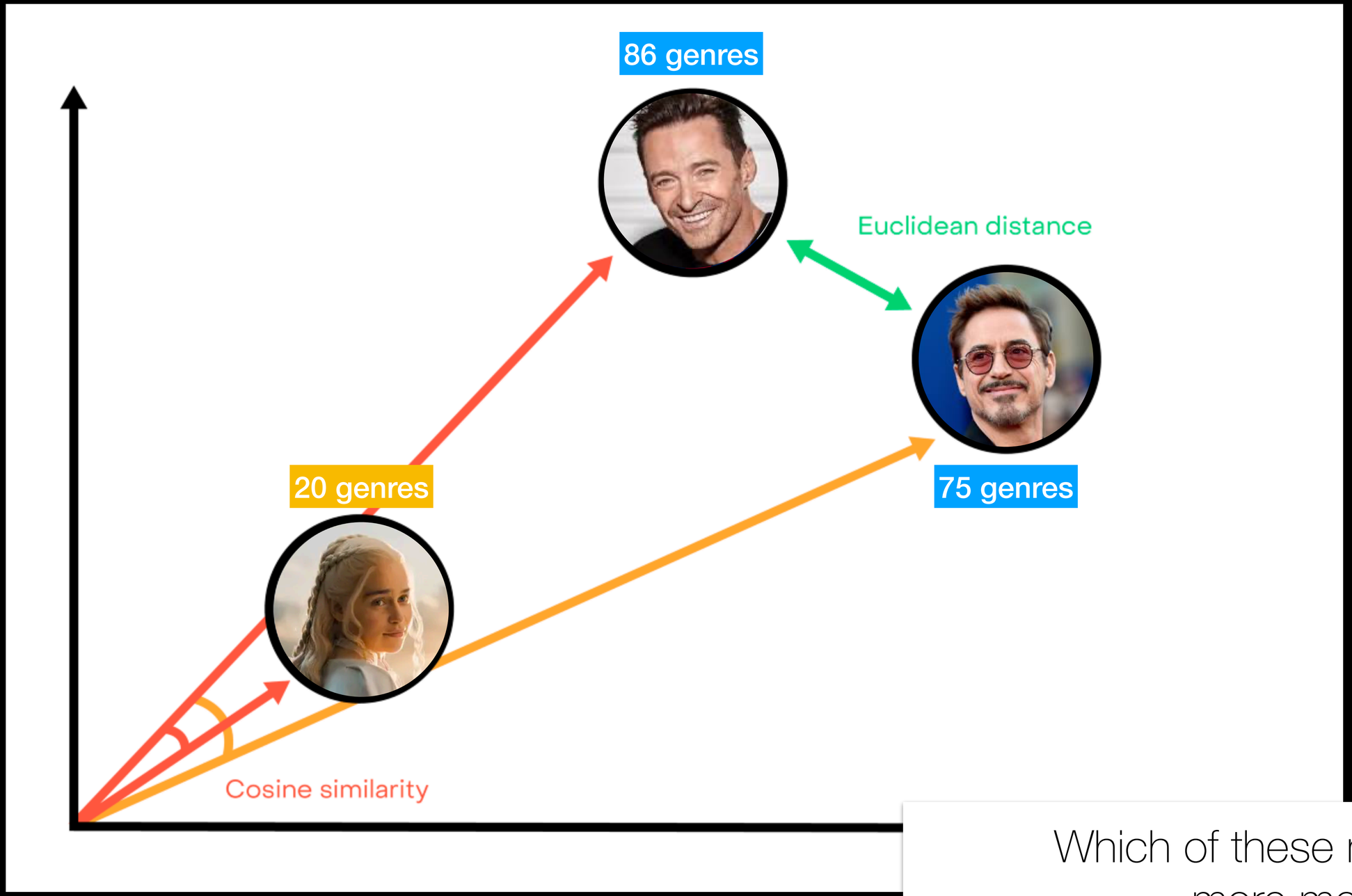
```
nm0413168 Hugh Jackman 0.0
nm3592338 Emilia Clarke 0.023038353871383088
nm1663205 Sharlto Copley 0.03680695049665894
nm0000375 Robert Downey Jr. 0.040157404642494 73
nm3772243 Theo James 0.04358329306959263
nm0262635 Chris Evans 0.04805589651668307
nm0881631 Karl Urban 0.05508543978437186
nm0159789 Hayden Christensen 0.060195990212304484
nm1517976 Chris Pine 0.07953201413353528
nm1475594 Channing Tatum 0.08366839174462504
```

Cosine distance between Hugh and other actors…

In [42]:
```
#Printing the top ten most similar actors to our target
for similar_actor_id, similar_genre_score in sorted(query_distances, key=lambda x: x[1], reverse=False)[:10]:
    print(similar_actor_id, actor_name_map[similar_actor_id], similar_genre_score)
```

```
nm0413168 Hugh Jackman 0.0
nm0000375 Robert Downey Jr. 7.937253933193772
nm0262635 Chris Evans 8.366600265340756
nm1517976 Chris Pine 10.770329614269007
nm1475594 Channing Tatum 10.954451150103322
nm0185819 Daniel Craig 11.445523142259598
nm0000226 Will Smith 11.916375287812984
nm0757855 Zoe Saldana 12.328828005937952
nm0004937 Jamie Foxx 12.529964086141668
nm0000234 Charlize Theron 12.922847983320086
```

Euclidean distance



Which of these metrics is more meaningful?

```
In [2]: pd.read_csv("Actor data.csv")
```

Out[2]:

	Actor	Film	Year	Salary	Total income	Total income_cleaned
0	Keanu Reeves	The Matrix ReloadedThe Matrix Revolutions	2003	$30,000,000	$156,000,000	15,60,00,000
1	Bruce Willis	The Sixth Sense	1999	$14,000,000	$100,000,000	10,00,00,000
2	Tom Cruise	Mission: Impossible 2	2000	NaN	$100,000,000	10,00,00,000
3	Tom Cruise	War of the Worlds	2005	NaN	$100,000,000	10,00,00,000
4	Will Smith	Men in Black 3	2012	NaN	$100,000,000	10,00,00,000
5	Robert Downey Jr.	Avengers: Infinity War	2018	NaN	$75,000,000+	7,50,00,000
6	Robert Downey Jr.	Avengers: Endgame	2019	$20,000,000	$75,000,000	7,50,00,000
7	Robert Downey Jr.	Iron Man 3	2013	NaN	$75,000,000	7,50,00,000
8	Sandra Bullock	Gravity	2013	$20,000,000	$70,000,000+	7,00,00,000
9	Tom Hanks	Forrest Gump	1994	NaN	$70,000,000	7,00,00,000
10	Tom Cruise	Mission: Impossible	1996	NaN	$70,000,000	7,00,00,000
11	Harrison Ford	Indiana Jones and the Kingdom of the Crystal S...	2008	NaN	$65,000,000	6,50,00,000
12	Jack Nicholson	Batman	1989	$6,000,000	$60,000,000	6,00,00,000
13	Leonardo DiCaprio	Inception	2010	NaN	$59,000,000	5,90,00,000
14	Johnny Depp	Pirates of the Caribbean: On Stranger Tides	2011	$35,000,000	$55,000,000	5,50,00,000
15	Robert Downey Jr.	The Avengers	2012	NaN	$50,000,000	5,00,00,000
16	Cameron Diaz	Bad Teacher	2011	NaN	$42,000,000	4,20,00,000
17	Robert Downey Jr.	Captain America: Civil War	2016	$40,000,000	$40,000,000+	4,00,00,000
18	Robert Downey Jr.	Avengers: Age of Ultron	2015	NaN	$40,000,000	4,00,00,000
19	Leonardo DiCaprio	Titanic	1997	NaN	$40,000,000	4,00,00,000
20	Tom Hanks	Saving Private Ryan	1998	NaN	$40,000,000	4,00,00,000

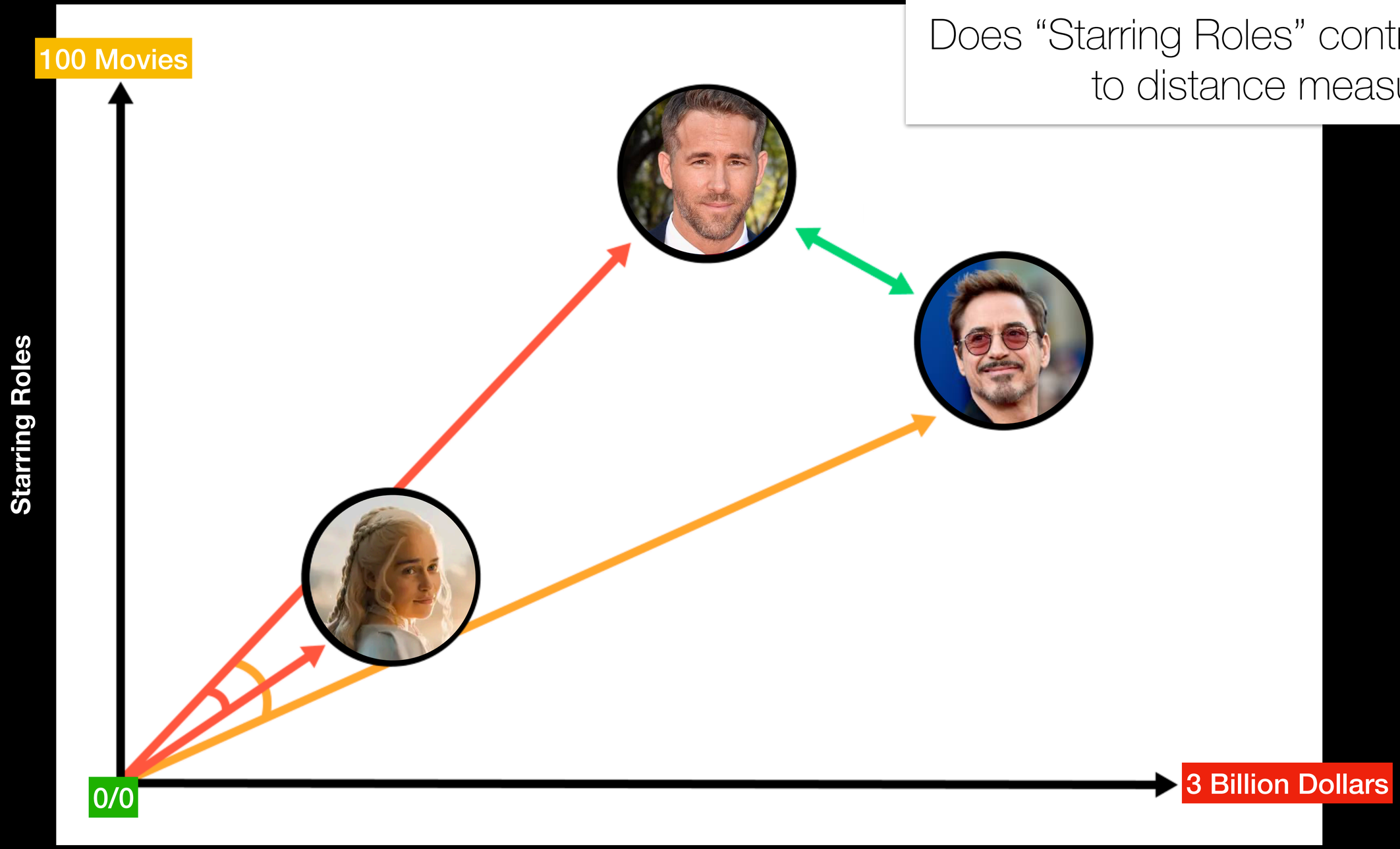
```
In [19]: actor_genre_df.head(20)
```

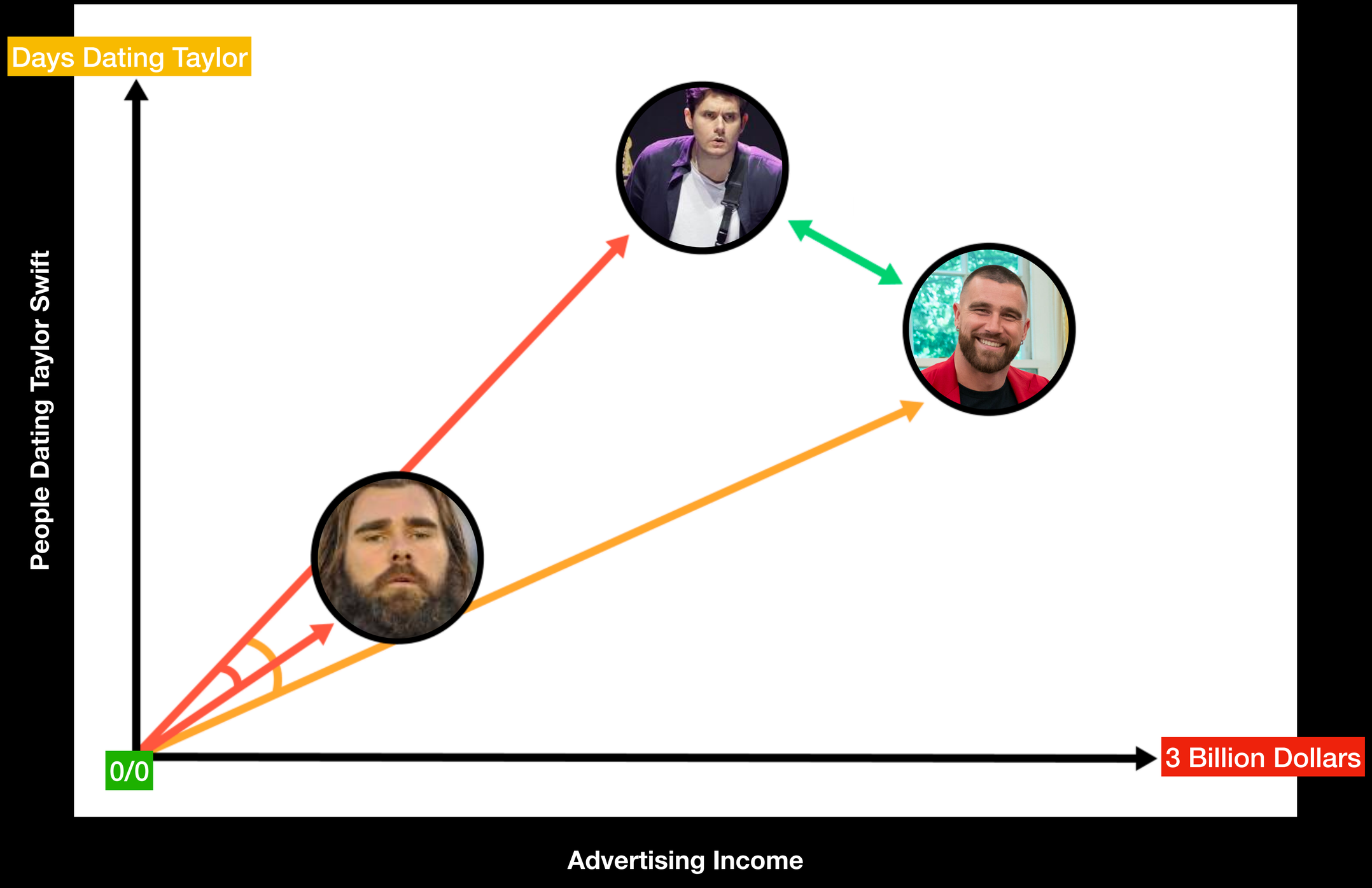
Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Shor	Total income
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$156,000,000
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$100,000,000
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0	0.0	0.0	0.0	$100,000,000
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0	3.0	0.0	0.0	$100,000,000
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$100,000,000
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000+
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$70,000,000+
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$70,000,000
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$70,000,000
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$65,000,000
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0	$60,000,000
nm0462648	1.0	0.0	1.0	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	$59,000,000
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$55,000,000
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$50,000,000
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	$42,000,000
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$40,000,000+
nm0177016																		
nm0907480																		

What does this inclusion mean for our distance metrics?

Does “Starring Roles” contribute to distance measures?




Days Dating Taylor
People Dating Taylor Swift
0/0
3 Billion Dollars
Advertising Income

These issues are all different data hygienes problems

Some can be addressed through models

Others through domain expertise

This Week's Learning Objectives

- Describe at least three ways data-hygiene problems may manifest
- Define two kinds of "normalization" for numeric data
- Convert categorical data to numeric data

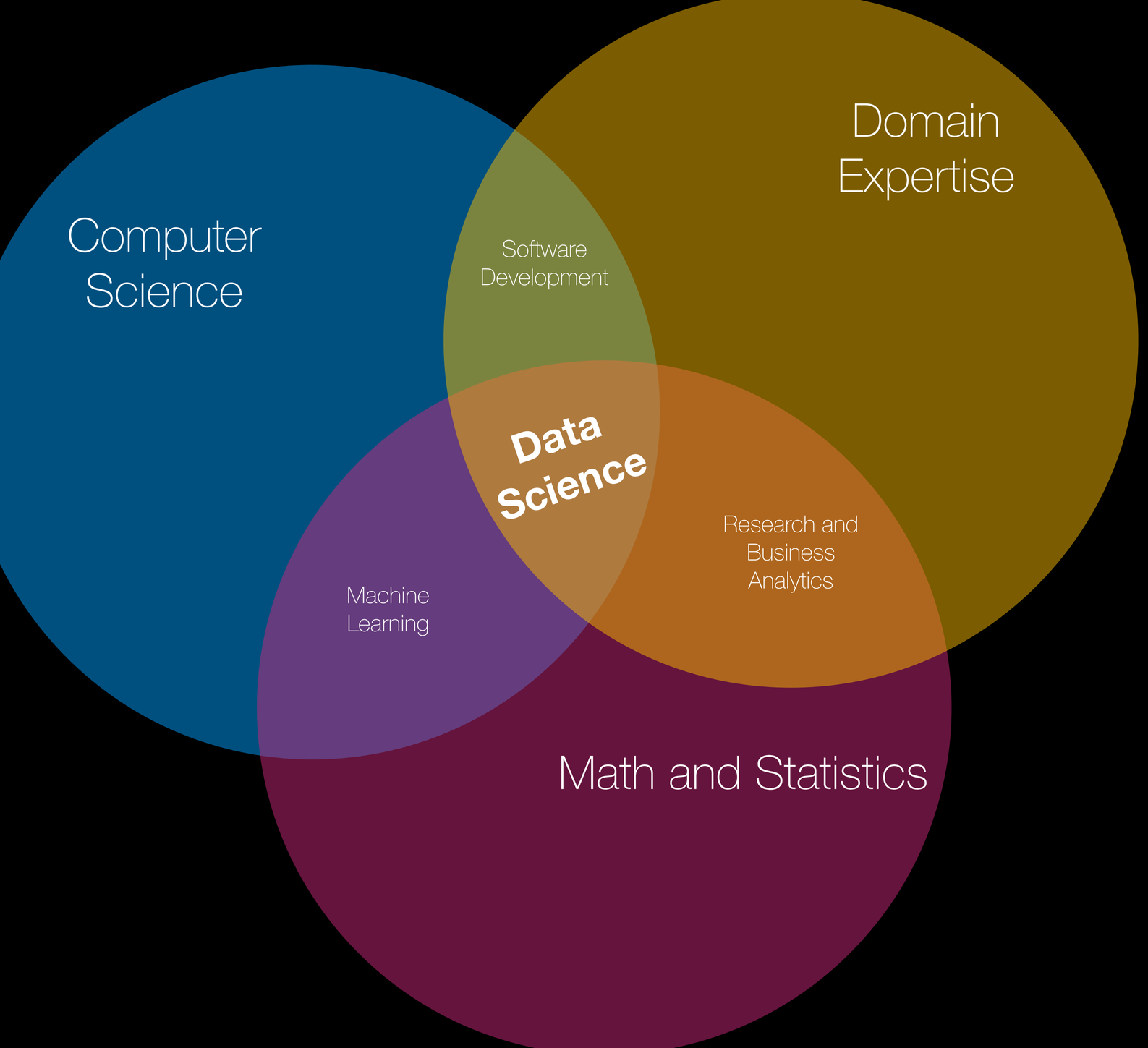
This Week's Learning Objectives

Describe at least three ways data-hygiene problems may manifest

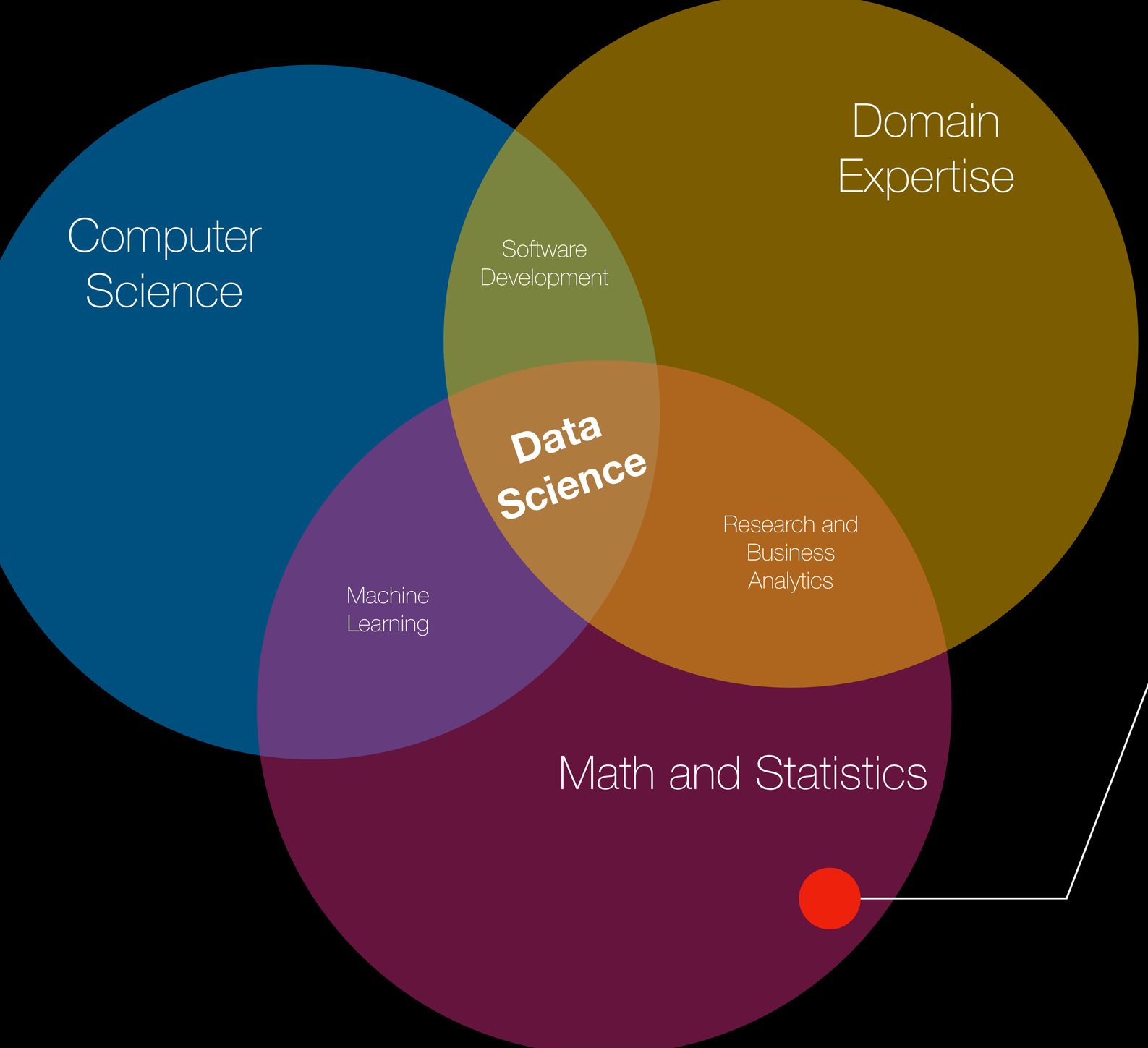
Define two kinds of "normalization" for numeric data

Convert categorical data to numeric data

How can you define “good” data?

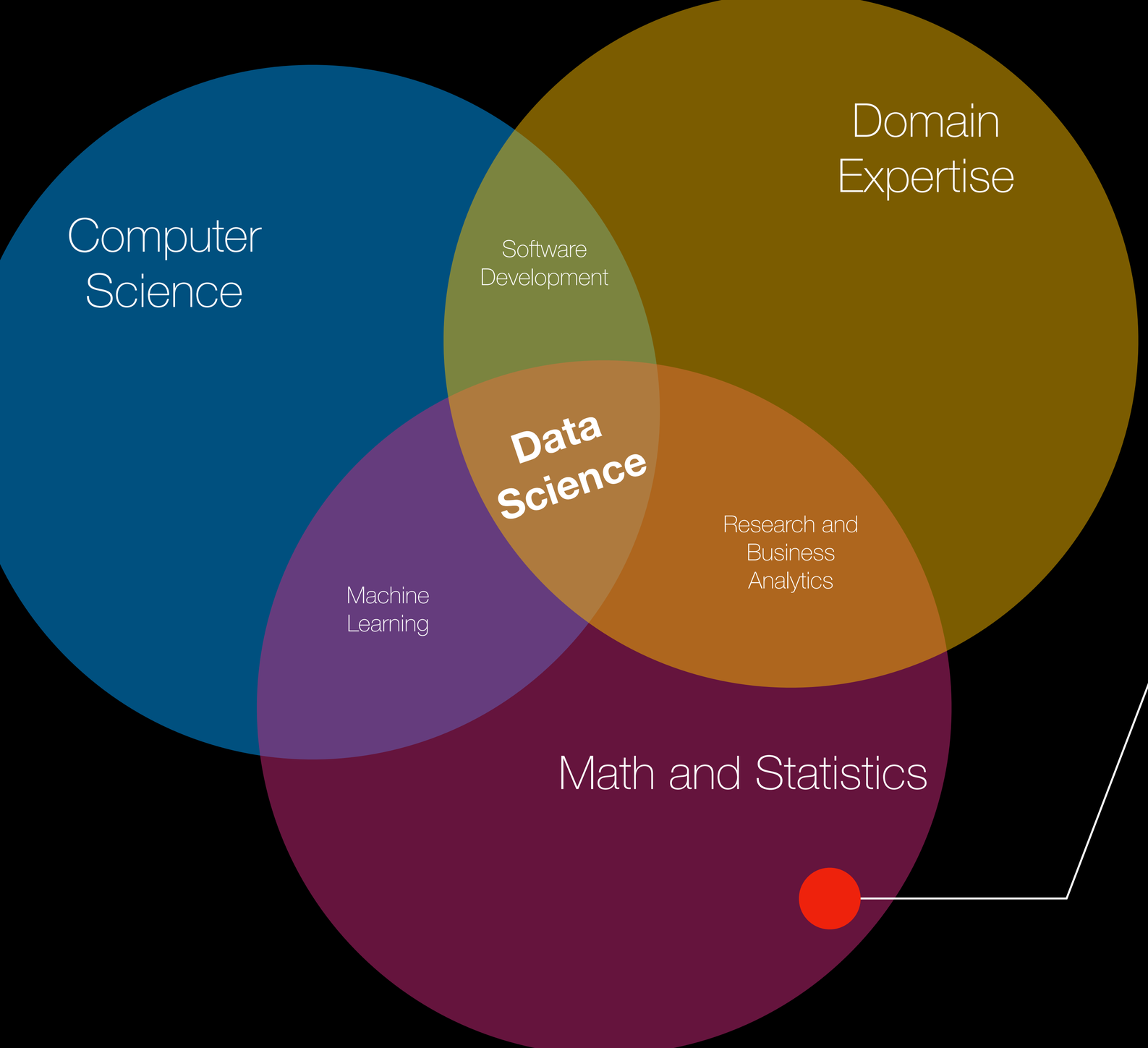


Data quality depends
on your perspective



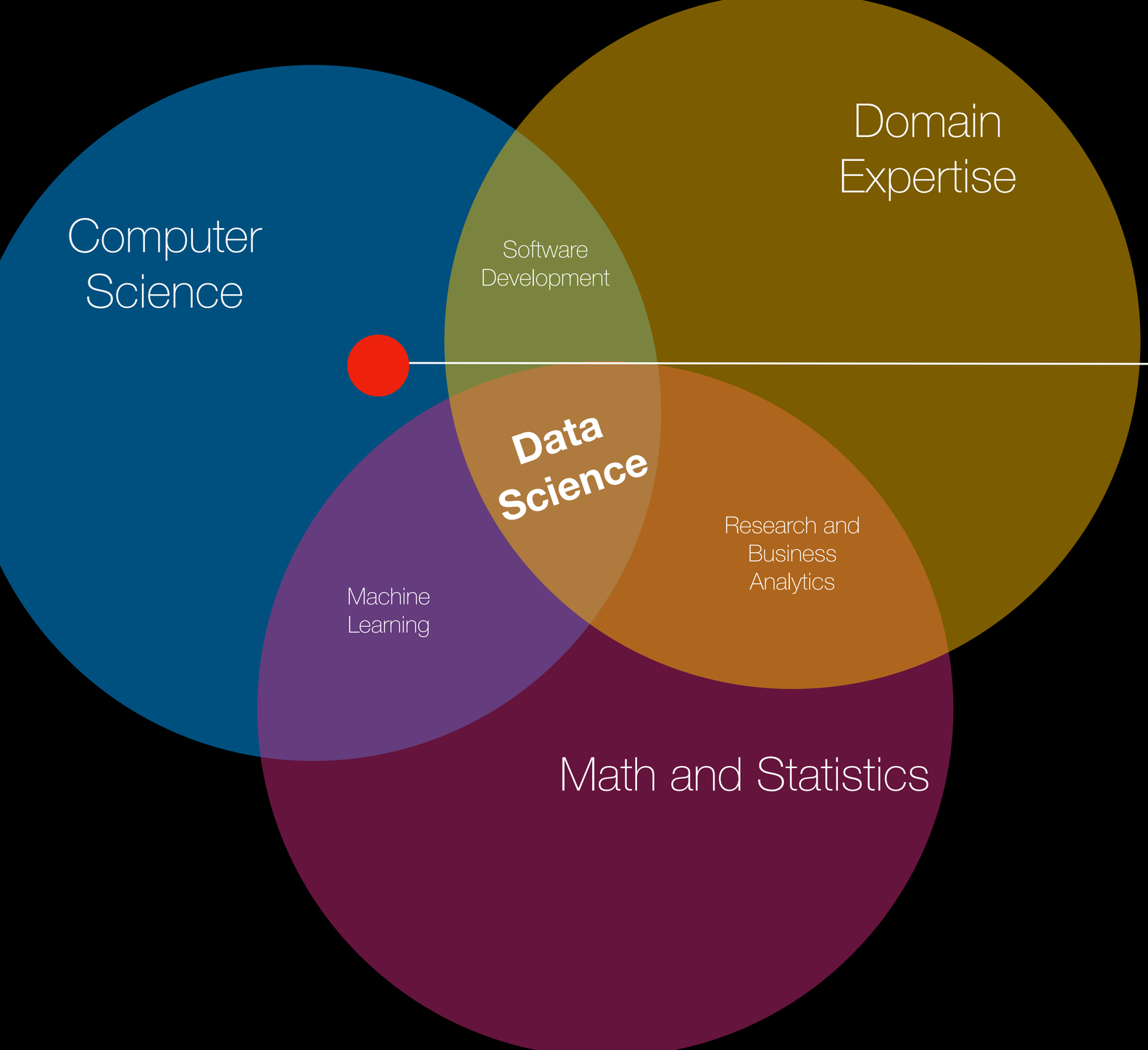
Underlying processes producing data are non-ideal

- Skewed
- Dependent
- Distorted
- Biased



Underlying processes producing data are non-ideal

Fix via more complex models

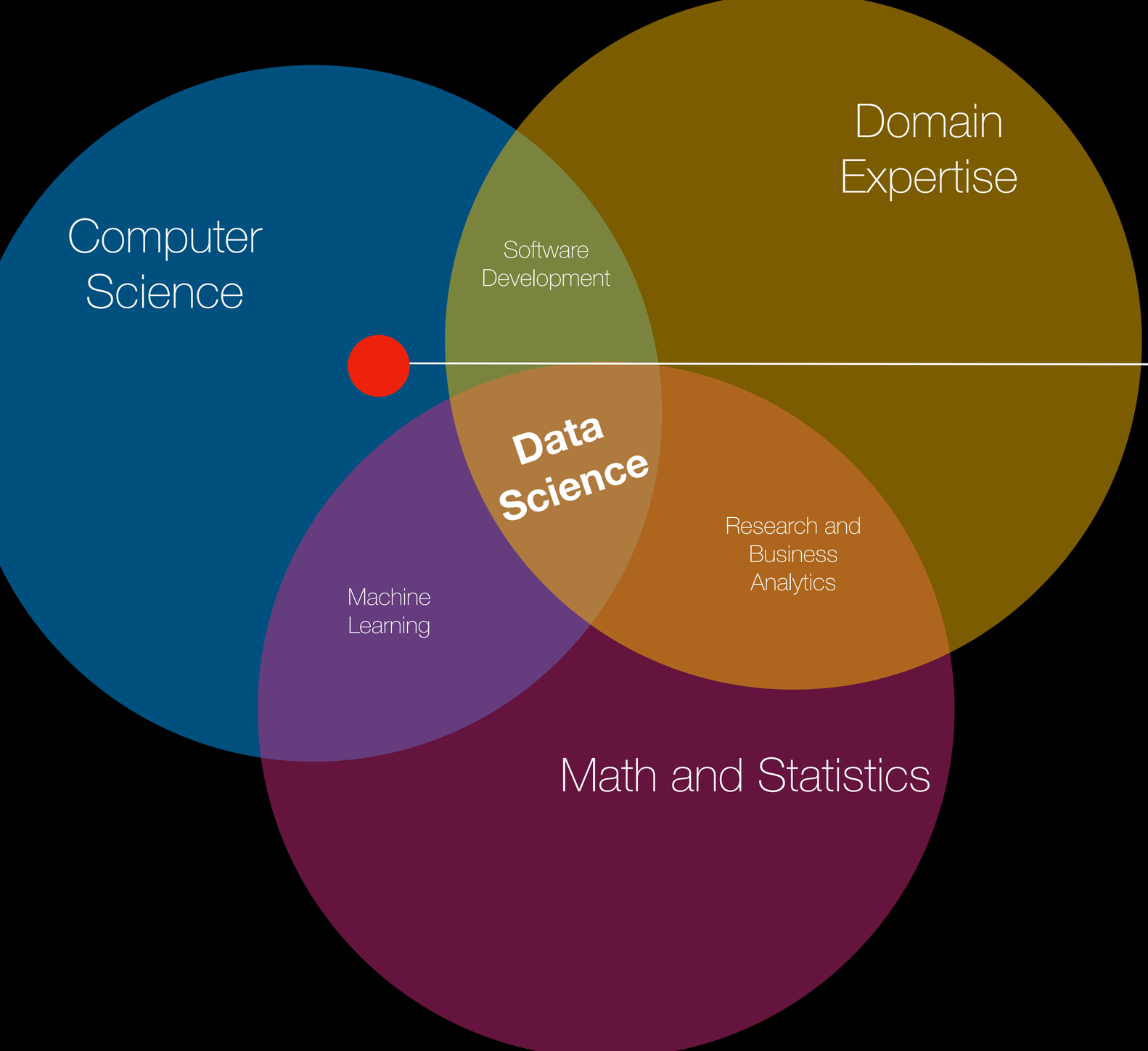


Values in my DB are missing or corrupt

Empty/NA/NULL cells

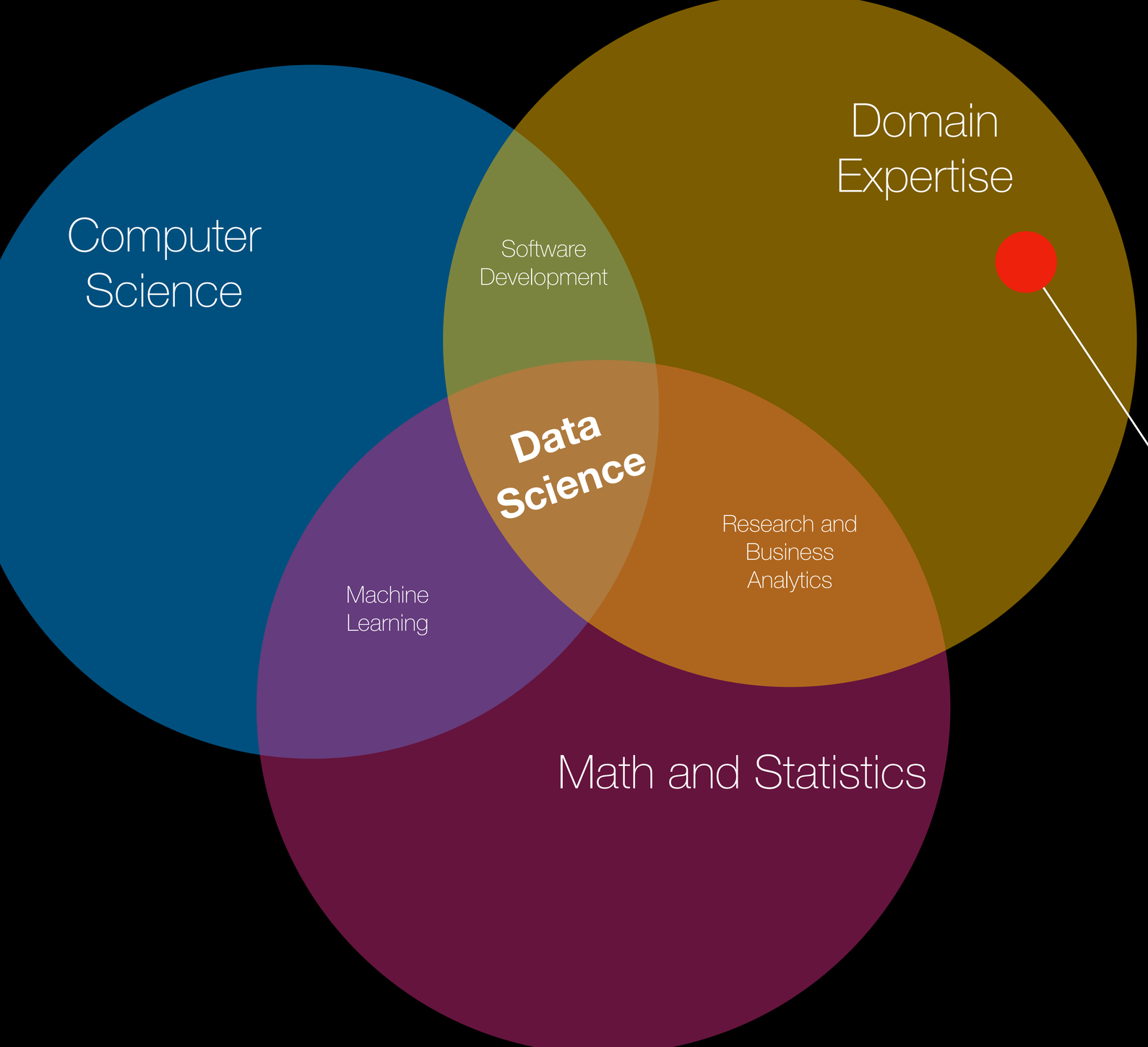
Corrupted

“Bit rot”



Values in my DB are missing or corrupt

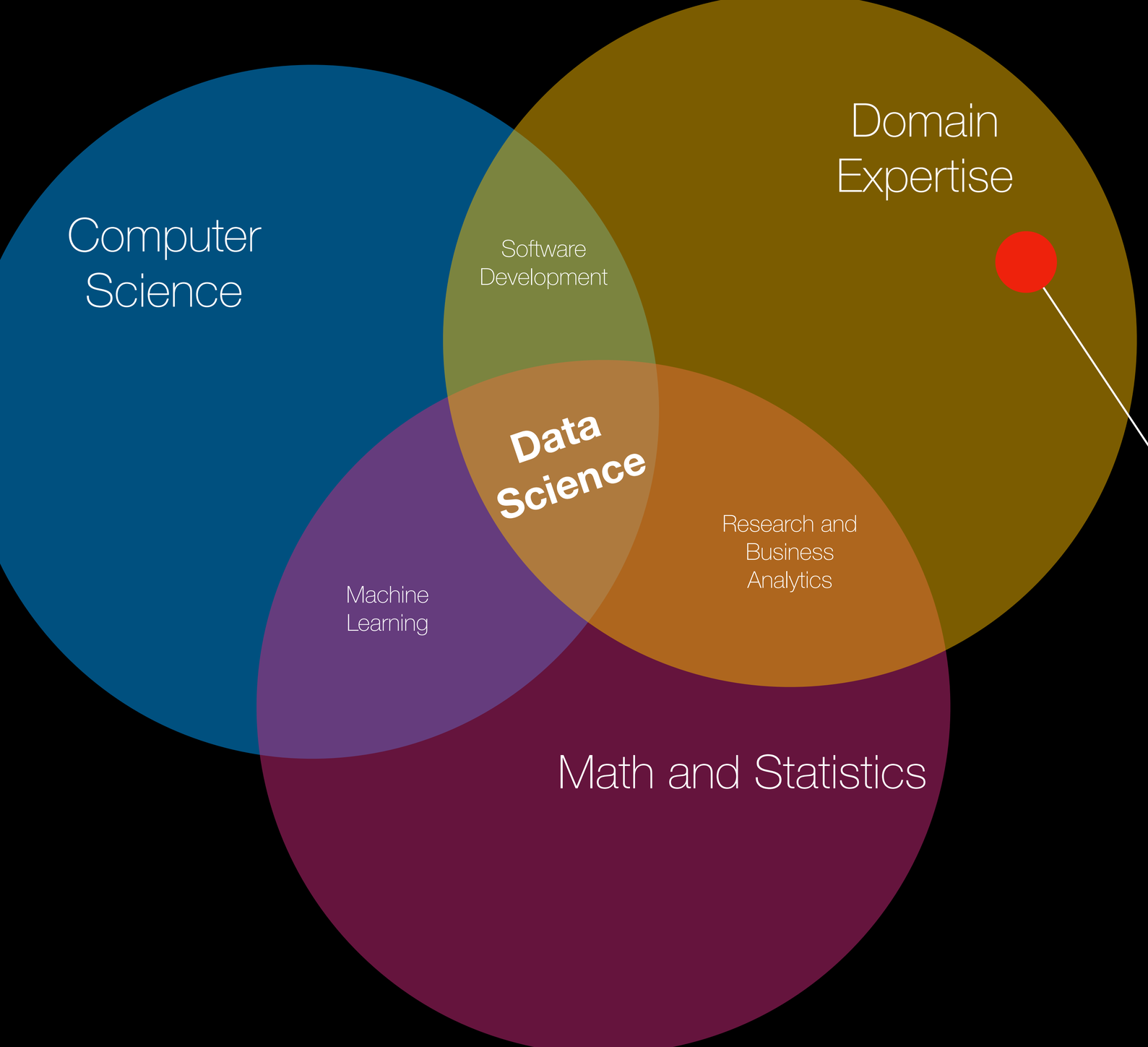
Drop invalid rows in the database



Data and results look “wrong”

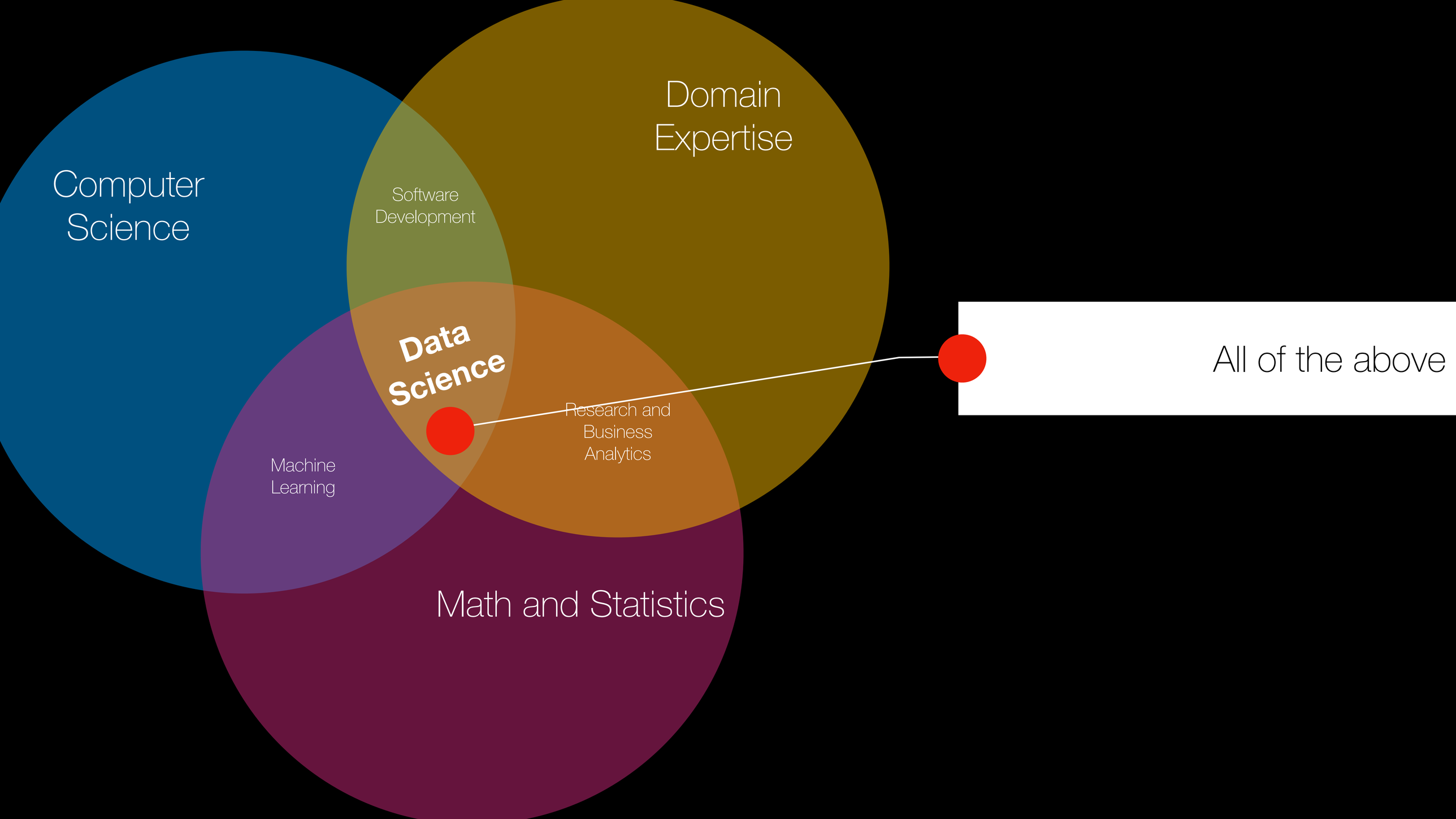
Violate expectations

Nonsensical results



Data and results look “wrong”

Fix by cleaning data or validating results


Computer Science
Domain Expertise
Software Development
Data Science
Research and Business Analytics
Machine Learning
Math and Statistics
All of the above

Five Aspects of (Traditional Data) Quality

Accuracy: Data recorded correctly

Completeness: All relevant data was recorded

Uniqueness: Entities are only recorded once

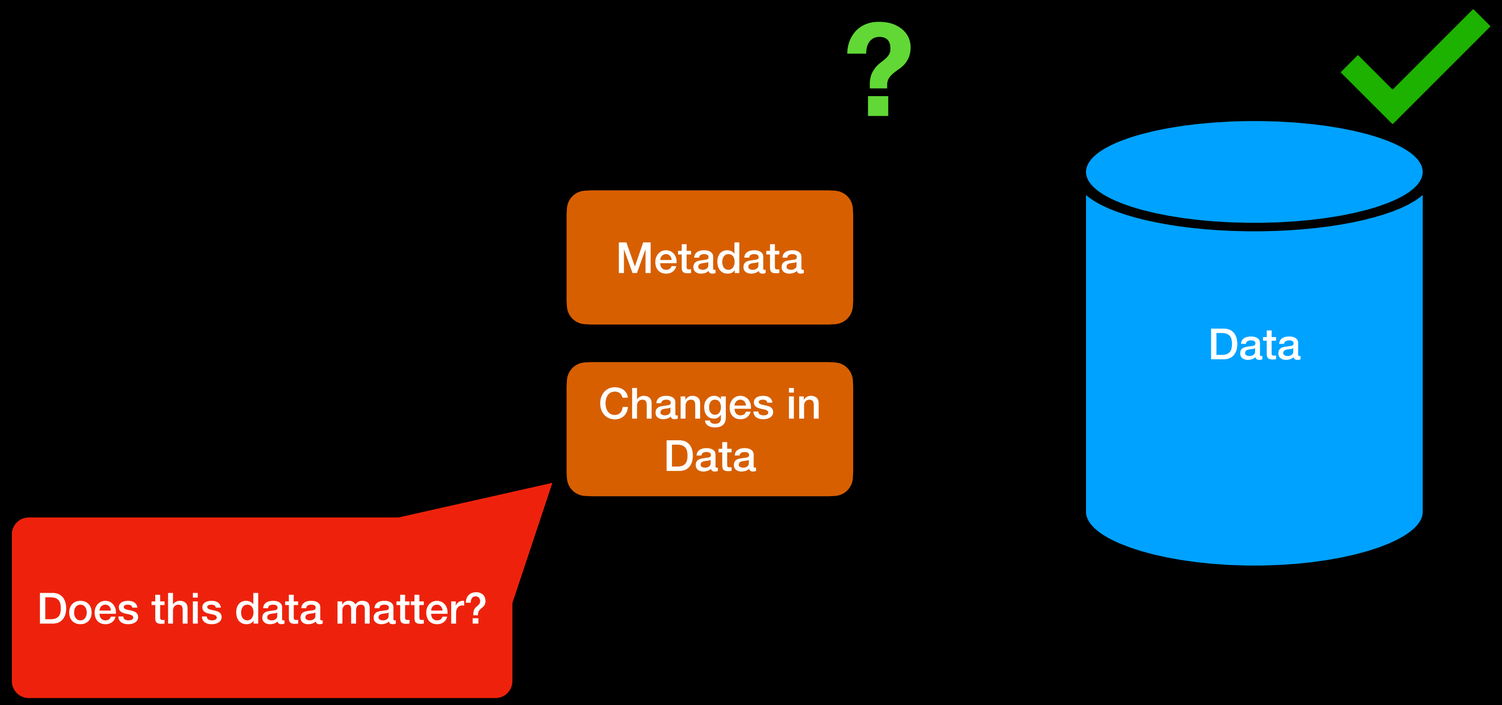
Timeliness: Data is kept up to date

Consistency: Data agrees with itself

Many of these qualities are difficult/impossible to measure

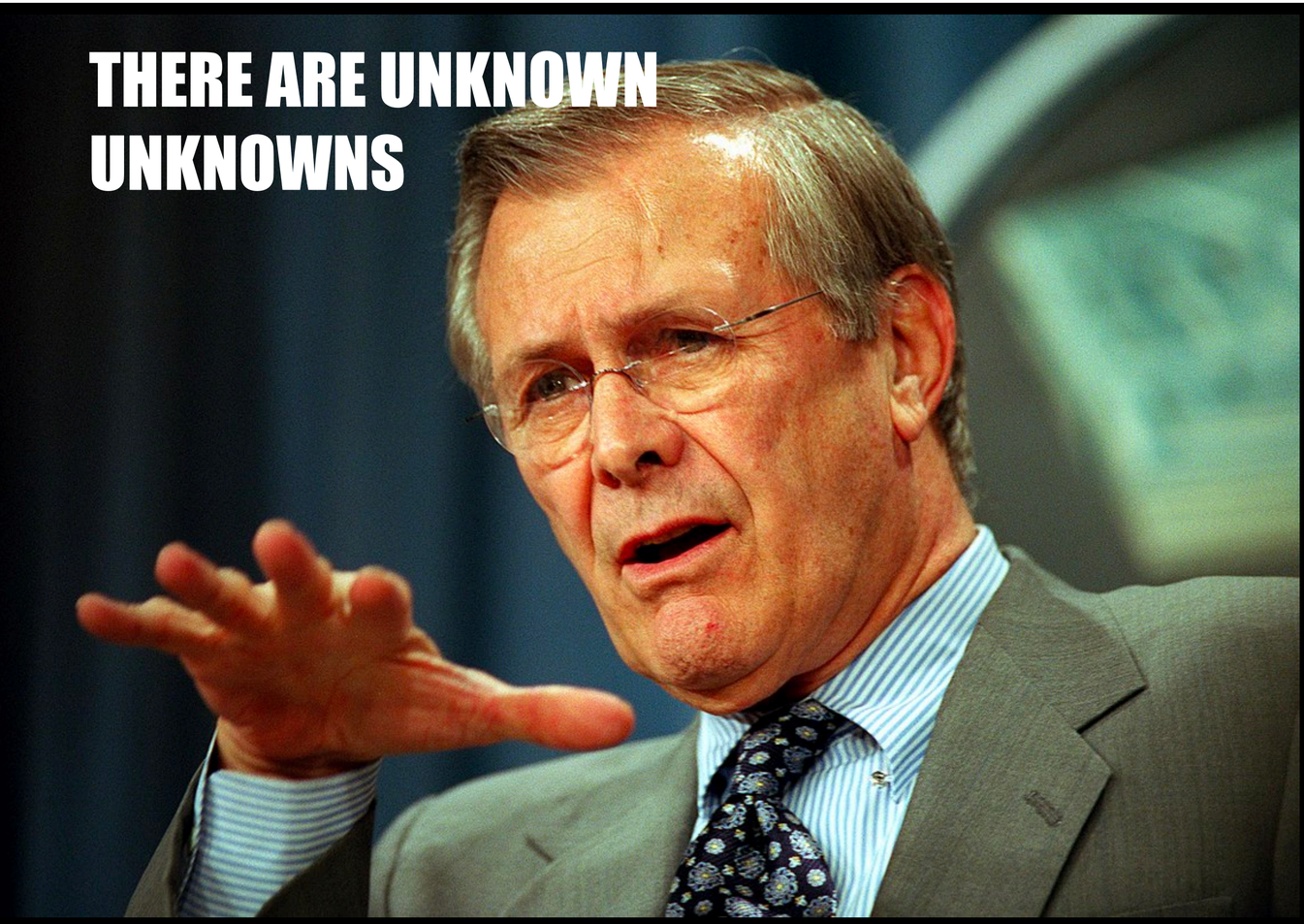
Five Aspects of (Traditional Data) Quality

Accuracy and Completeness



Five Aspects of (Traditional Data) Quality

Accuracy and Completeness



Five Aspects of (Traditional Data) Quality

Uniqueness

Five Aspects of (Traditional Data) Quality

Timeliness

We can take a more process-oriented approach to data quality

Data Collection

Broken Sensors

Human Error

Data Collection

Broken Sensors

Human Error

G2.com

Address Validation

Checks a postal address against an authoritative database to **determine deliverability.**

Address Verification

Formats, transliterates, and enhances address lists to **ensure accuracy.**

Data Formats

Format Validation

Data Validation

Data Collection

Broken Sensors

Human Error

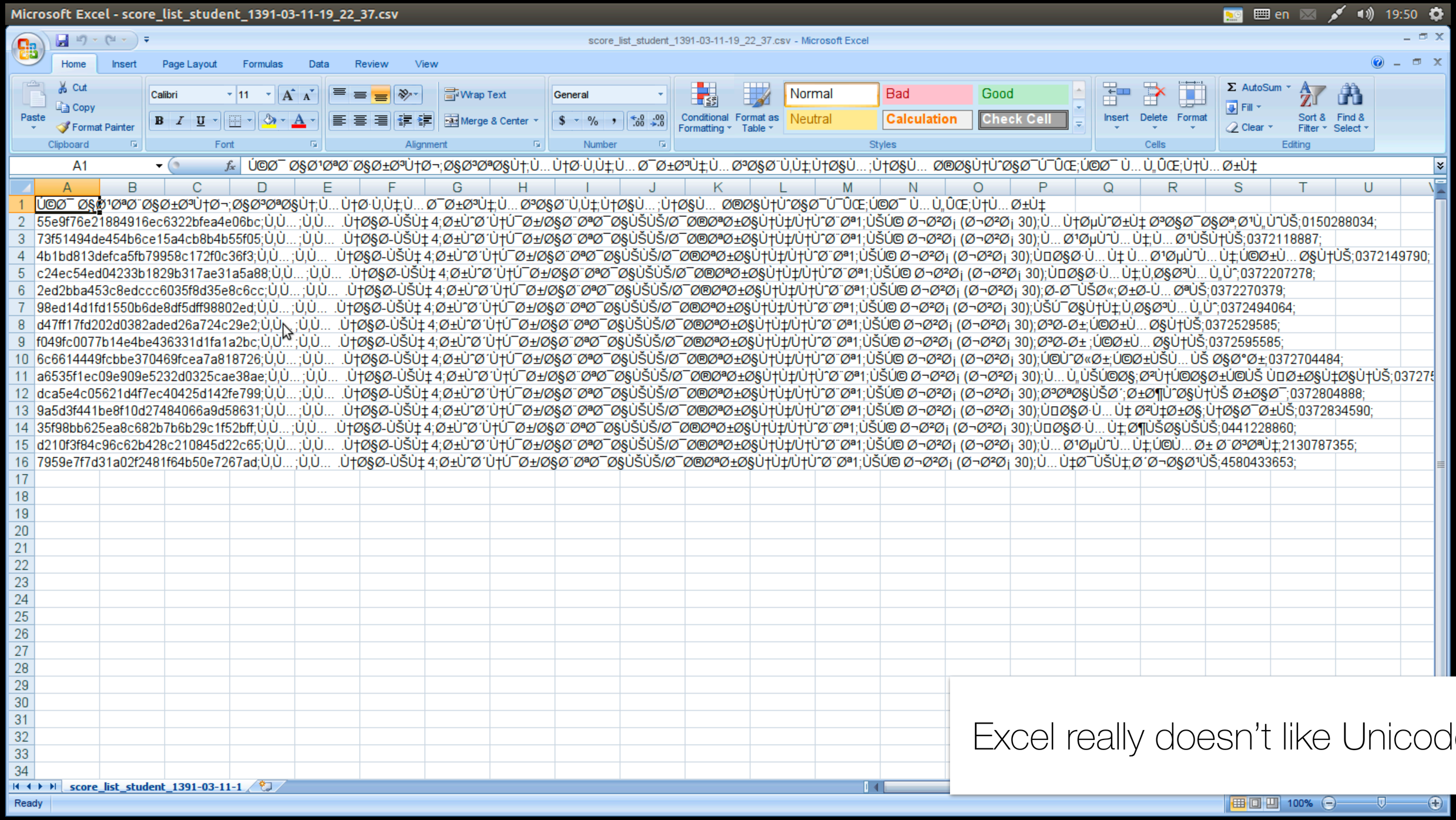
Provenance

Versioning

“Bit Rot”

Transcoding Issues

Ad Hoc Modification


Microsoft Excel - score_list_student_1391-03-11-19_22_37.csv
score_list_student_1391-03-11-19_22_37.csv - Microsoft Excel
Home
Insert
Page Layout
Formulas
Data
Review
View
Cut
Copy
Format Painter
Paste
Clipboard
Calibri
11
Font
Wrap Text
Merge & Center
Alignment
General
Number
Conditional Formatting
Format as Table
Normal
Bad
Good
Neutral
Calculation
Check Cell
Styles
Insert
Delete
Format
Cells
AutoSum
Fill
Clear
Sort & Filter
Find & Select
Editing
A1
score_list_student_1391-03-11-1
Ready
100%
Excel really doesn't like Unicode

Checksums

Manual Validation

Data Specifications

Data Collection

Data Storage

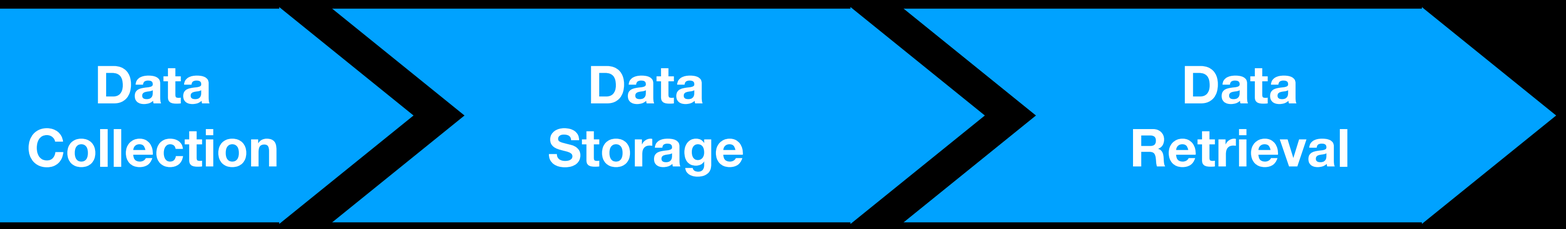
Provenance

Versioning

“Bit Rot”

Transcoding Issues

Ad Hoc Modification



Incomplete Data

Bias in Samples

Incomplete Documentation

developer.twitter.com/en/docs/twitter-api

Developer Platform | Products | Docs | Use Cases | Community

Documentation

Search the docs

- Getting started
- Fundamentals
- Tools and libraries
- Tutorials
- API reference index

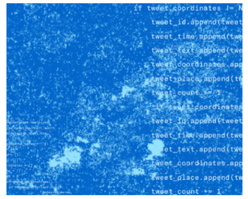
Twitter API

- Getting started
- Tools and libraries
- What to build
- Migrate
- Twitter API v2
- Enterprise - Gnip 2.0
- Premium v1.1
- Standard v1.1

Twitter Ads API

Twitter API v2

Twitter API v2 is ready for prime time! We recommend that the majority of de
to v2 of the API, and for any new users to get started with v2. Why migrate?



New and more detailed data objects

V2 Access Levels

Essential Elevated

Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose

Fred Morstatter
Arizona State University
699 S. Mill Ave.
Tempe, AZ, 85281

Jürgen Pfeffer
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA, 15213

Huan Liu
Arizona State University
699 S. Mill Ave.
Tempe, AZ, 85281

Kathleen M. Carley
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA, 15213

Abstract

Twitter is a social media giant famous for the exchange of short, 140-character messages called "tweets". In the scientific community, the microblogging site is known for openness in sharing its data. It provides a glance into its millions of users and billions of tweets through a "Streaming API" which provides a sample of all tweets matching some parameters preset by the API user. The API service has been used by many researchers, companies, and governmental institutions that want to extract knowledge in accordance with a diverse array of questions pertaining to social media. The essential drawback of the Twitter API is the lack of documentation concerning what and how much data users get. This leads researchers to question whether the sampled data is a valid representation of the overall activity on Twitter. In this work we embark on answering this question by comparing data collected using Twitter's sampled API service with data collected using the full, albeit costly, Firehose stream that includes every single published tweet. We compare both datasets using common statistical metrics as well as metrics that allow us to compare topics, networks, and locations of tweets. The results of our work will help researchers and practitioners understand the implications of using the Streaming API.

Introduction

Twitter is a microblogging site where users exchange short, 140-character messages called "tweets". Ranking as the 10th most popular site in the world by the Alexa rank in January of 2013[1], the site boasts 500 million registered users publishing 400 million tweets per day. Twitter's platform for rapid communication is said to be a vital communication platform in recent events including Hurricane Sandy[2], the Arab Spring of 2011 (Campbell 2011), and several political campaigns (Tumasjan et al. 2010; Gayo-Avello, Metaxas, and Mustafaraj 2011). As a result, Twitter's data has been coveted by both computer and social scientists to better understand human behavior and dynamics.

Social media data is often difficult to obtain, with most social media sites restricting access to their data. Twitter's policies lie opposite to this. The "Twitter Streaming API"[3] is a capability provided by Twitter that allows anyone to retrieve at most a 1% sample of all the data by providing some parameters. According to the documentation, the sample will return at most 1% of all the tweets produced on Twitter at a given time. Once the number of tweets matching the given parameters eclipses 1% of all the tweets on Twitter, Twitter will begin to sample the data returned to the user. The methods that Twitter employs to sample this data is currently unknown. The Streaming API takes three parameters: keywords (words, phrases, or hashtags), geographical boundary boxes, and user ID.

One way to overcome the 1% limitation is to use the Twitter Firehose—a feed provided by Twitter that allows access to 100% of all public tweets. A very substantial drawback of the Firehose data is the restrictive cost. Another drawback is the sheer amount of resources required to retain the Firehose data (servers, network availability, and disk space). Consequently, researchers as well as decision makers in companies and government institutions are forced to decide between two versions of the API: the freely-available but limited Streaming, and the very expensive but comprehensive Firehose version. To the best of our knowledge, no research has been done to assist those researchers and decision makers by answering the following: How does the use of the Streaming API affect common measures and metrics performed on the data? In this article we answer this question from different perspectives.

We begin the analysis by employing classic statistical measures commonly used to compare two sets of data. Based on unique characteristics of tweets, we design and conduct additional comparative analysis. By extracting topics using a frequently used algorithm, we compare how topics differ between the two datasets. As tweets are linked data, we perform network measures of the two datasets. Because tweets can be geo-tagged, we compare the geographical distribution of geolocated tweets to better understand

fort.fb.com/researcher-datasets

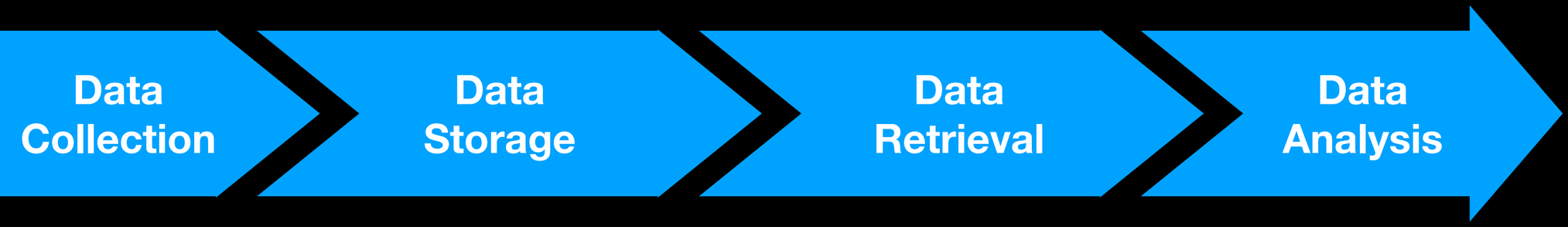
Meta

FORT Approach Researcher Products Key Research Areas

Researcher Datasets

Does “NULL” mean the link was never sent or was sent and not rated?

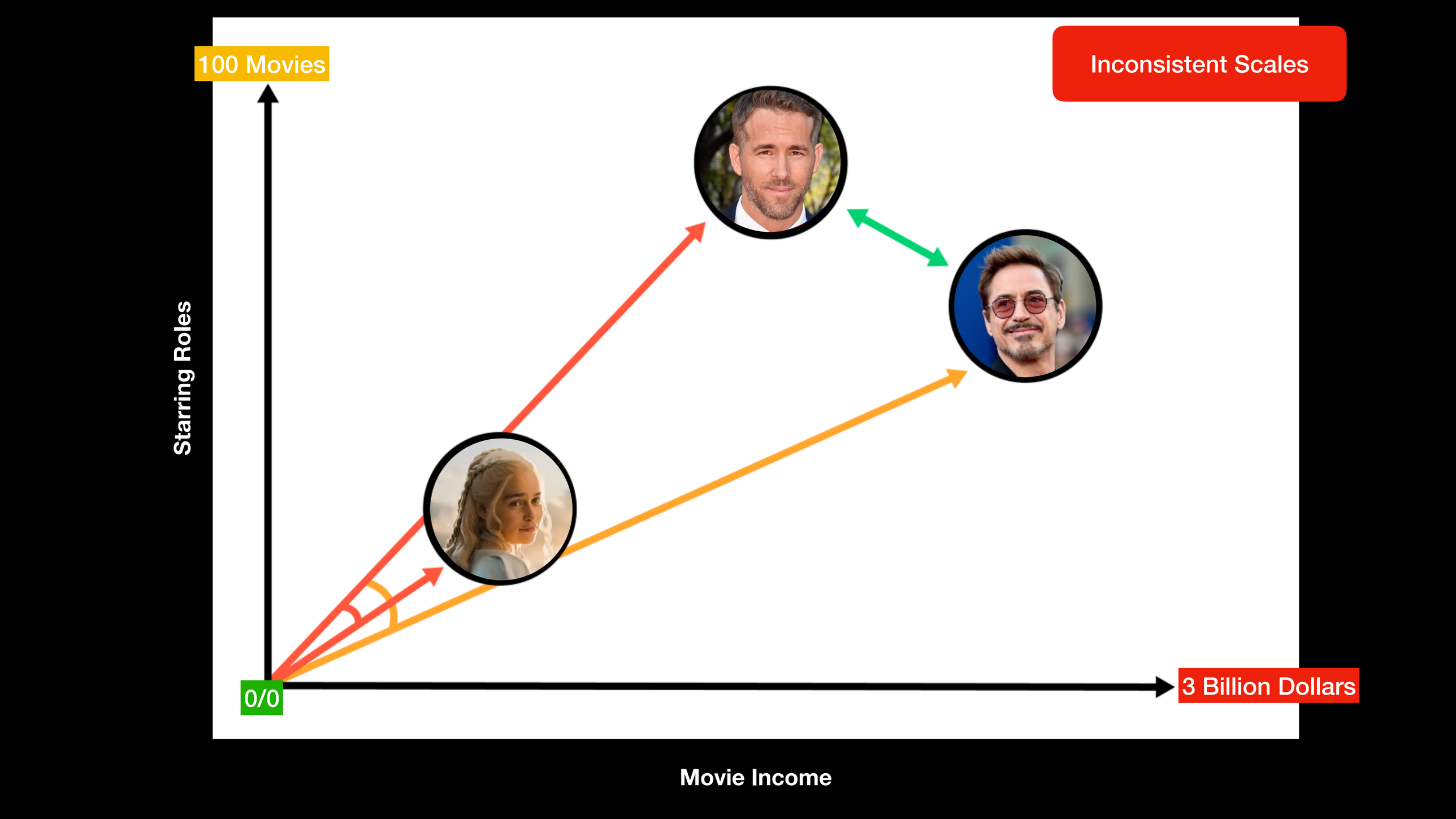
- **Tpfc_rating [text]:** If URL was sent to third-party fact-checkers (tpfc), did they rate it (NULL if not) and if so, how did they rate it? Category values include: ‘True’, ‘False’, ‘Prank Generator’ , ‘False Headline or Mixture’, ‘Opinion’, ‘Satire’, ‘Not Eligible, ‘Not Rated.’ Definitions, and a list of fact checkers, are available here: https://www.facebook.com/help/publisher/182222309230722 and https://www.facebook.com/help/572838089565953?helpref=faq_content. More information on how news that may be false is selected can be found here: https://www.facebook.com/help/1952307158131536. Only available for some stories, only available in Argentina, Brazil, Cameroon, Canada, Colombia, Denmark, France, Germany, India, Indonesia, Ireland, Italy, Kenya, Mexico, Middle East and North Africa, Netherlands, Nigeria, Norway, Pakistan, Philippines, Senegal, South Africa, Sweden, Turkey, UK, US. When more than one rating is given to a story, we use Facebook’s *precedence rules*, described below.



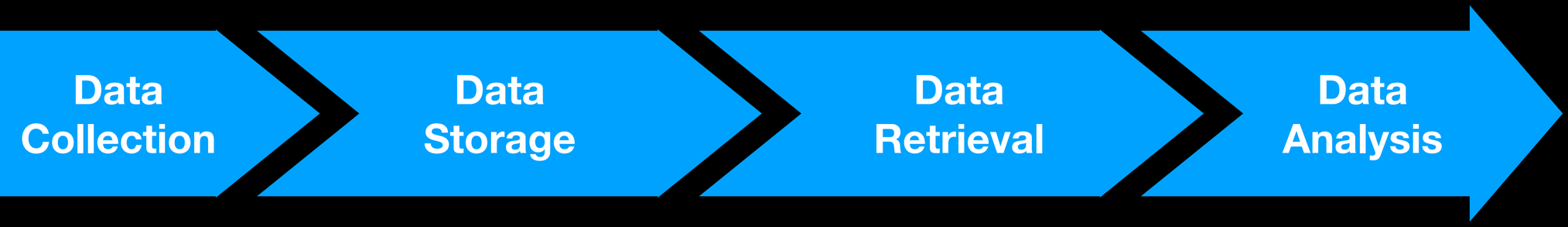
Inconsistent Scales

Instability in Results

Inappropriate Models


100 Movies
Inconsistent Scales
Starring Roles
0/0
3 Billion Dollars
Movie Income

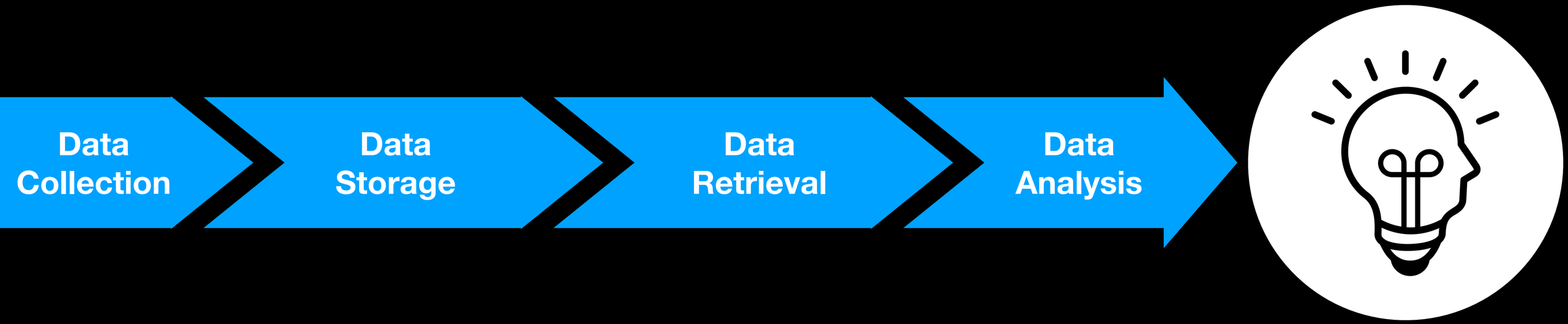
Data Validation



Inconsistent Scales

Instability in Results

Inappropriate Models


Data Collection
Data Storage
Data Retrieval
Data Analysis

This Week's Learning Objectives

Describe at least three ways data-hygiene problems may manifest

Define two kinds of "normalization" for numeric data

Convert categorical data to numeric data

This Week's Learning Objectives

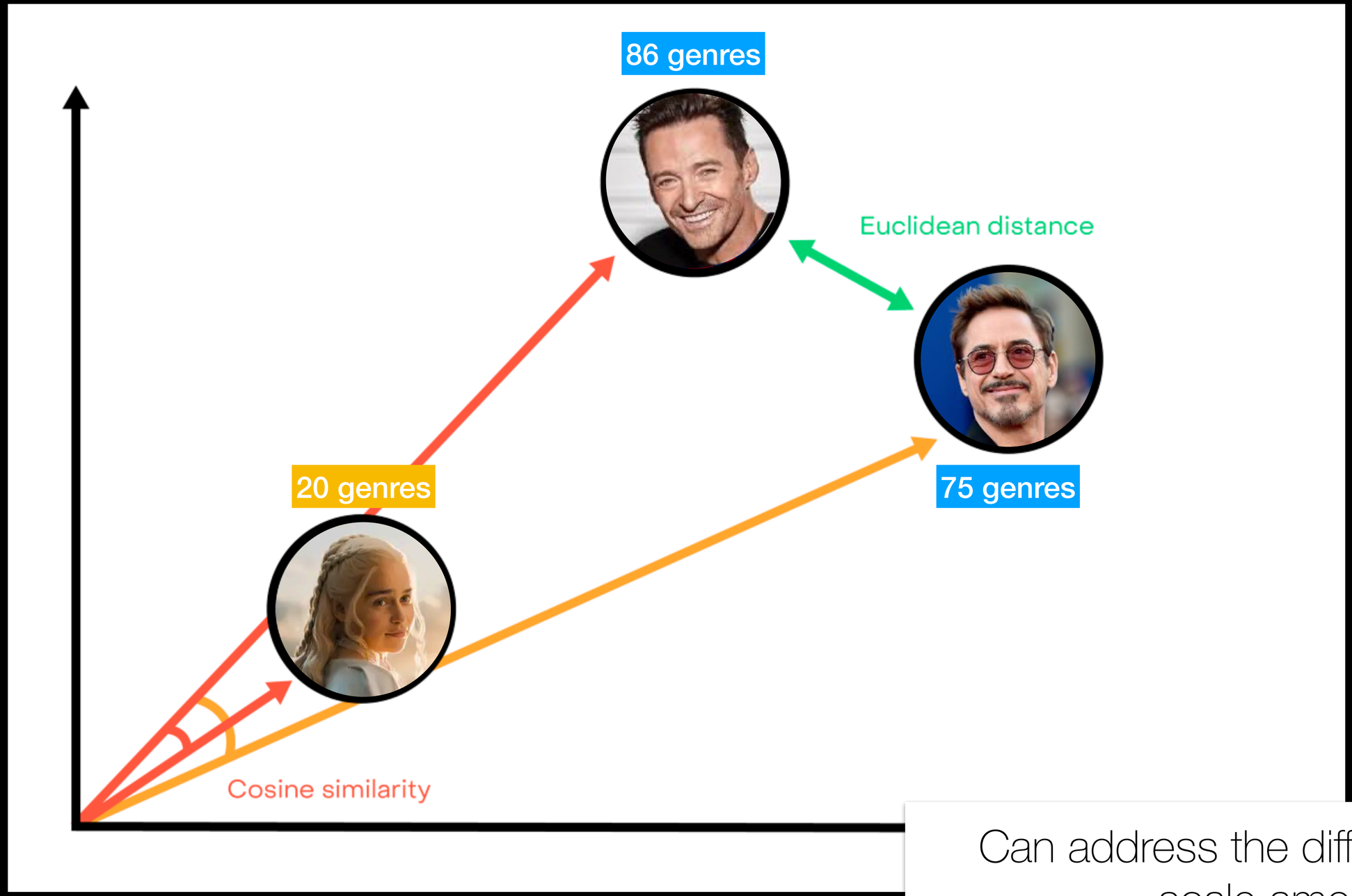
Describe at least three ways data-hygiene problems may manifest

Define two kinds of "normalization" for numeric data

Convert categorical data to numeric data

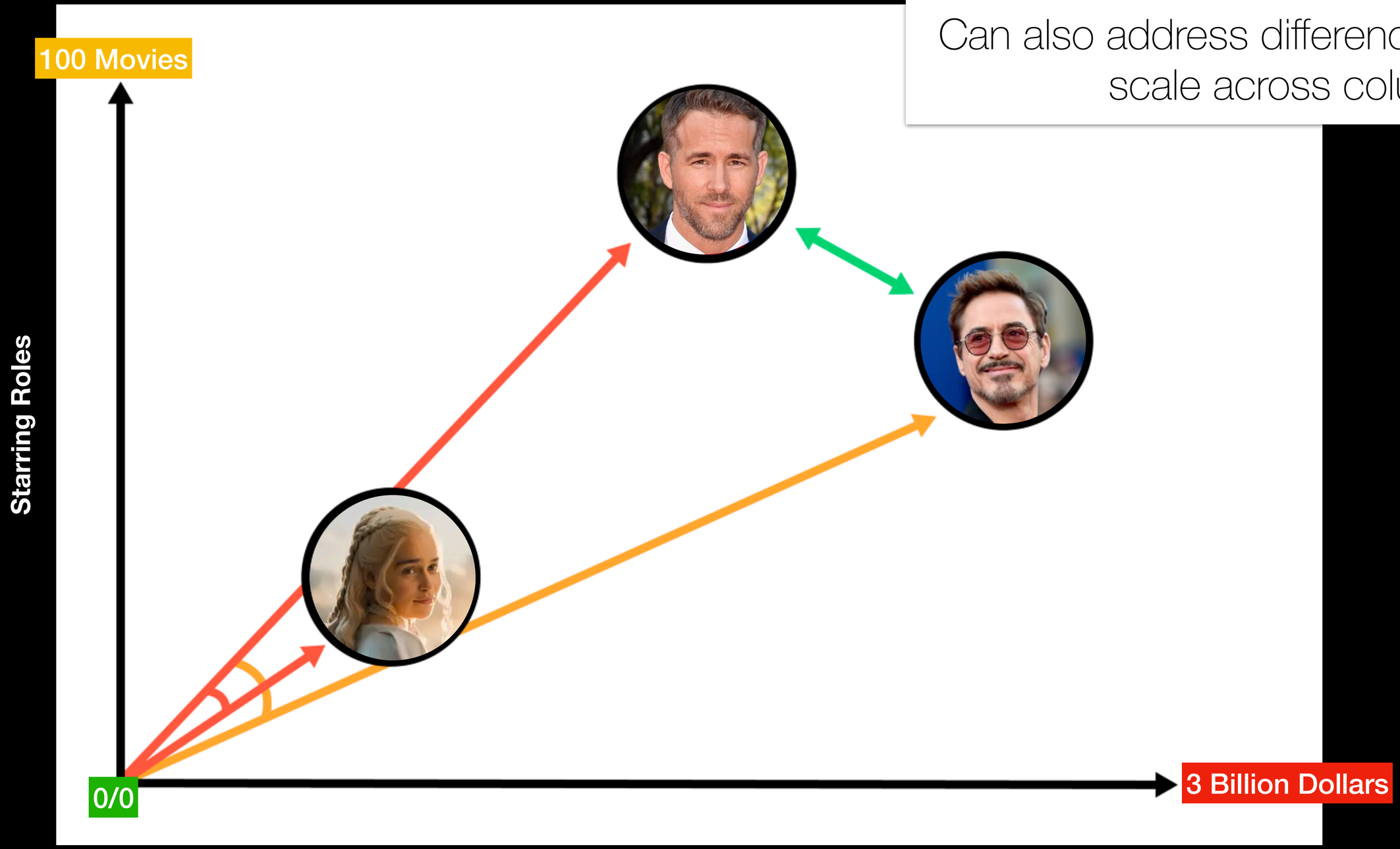
Many issues can’t be solved purely computationally

But some can

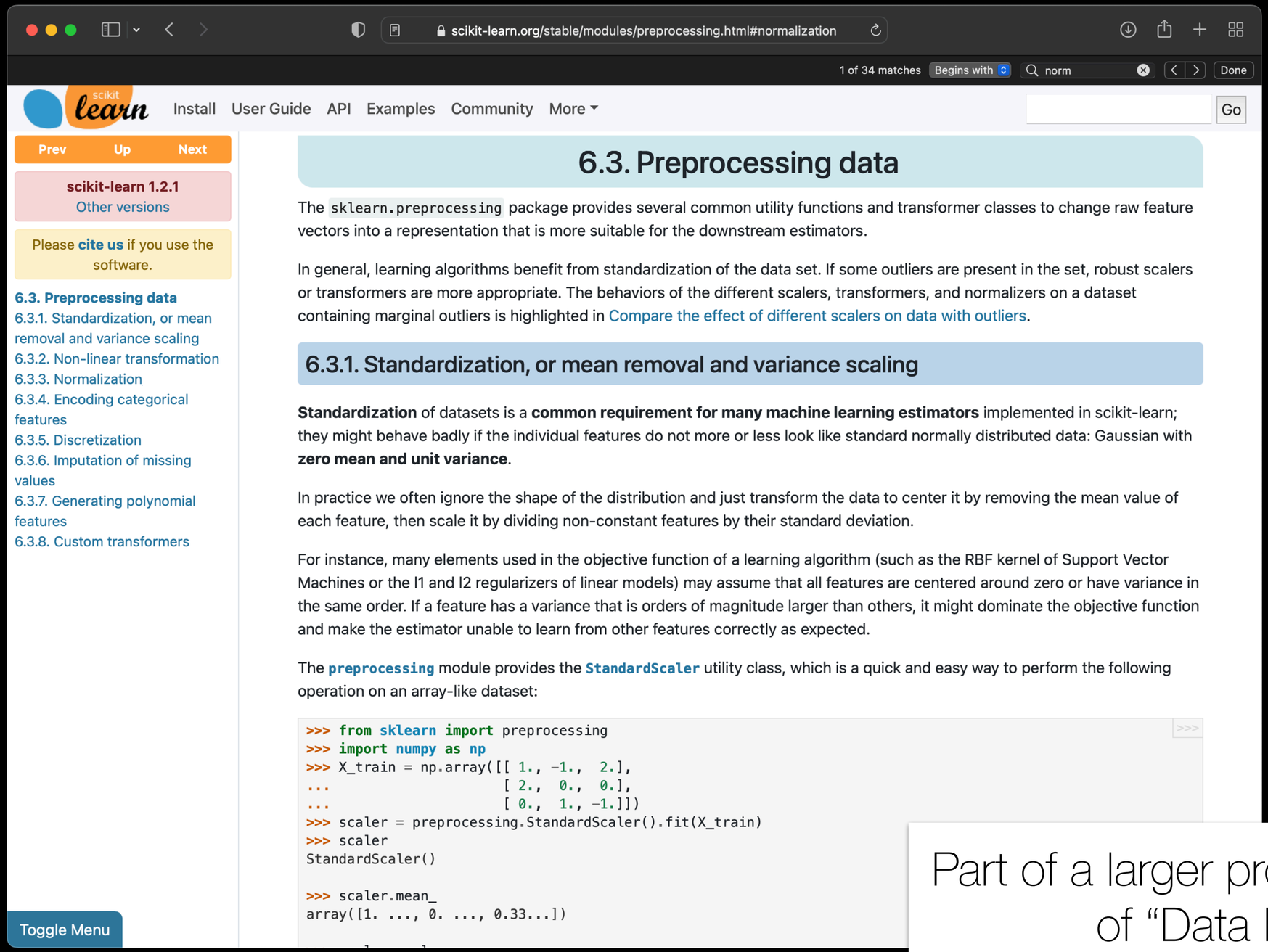


Can address the difference in scale among actors

Can also address differences in scale across columns



Data Normalization: scaling individual samples to have the same (often unit) norm

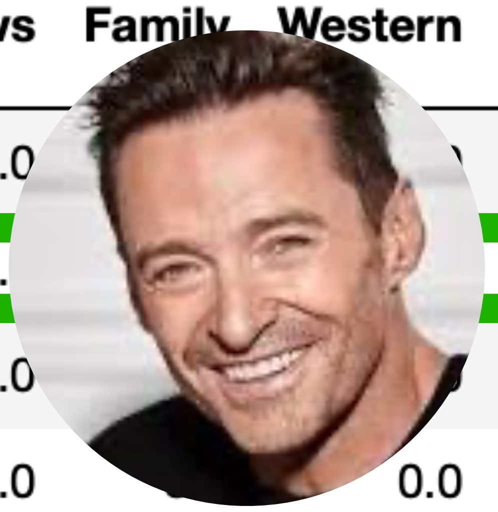


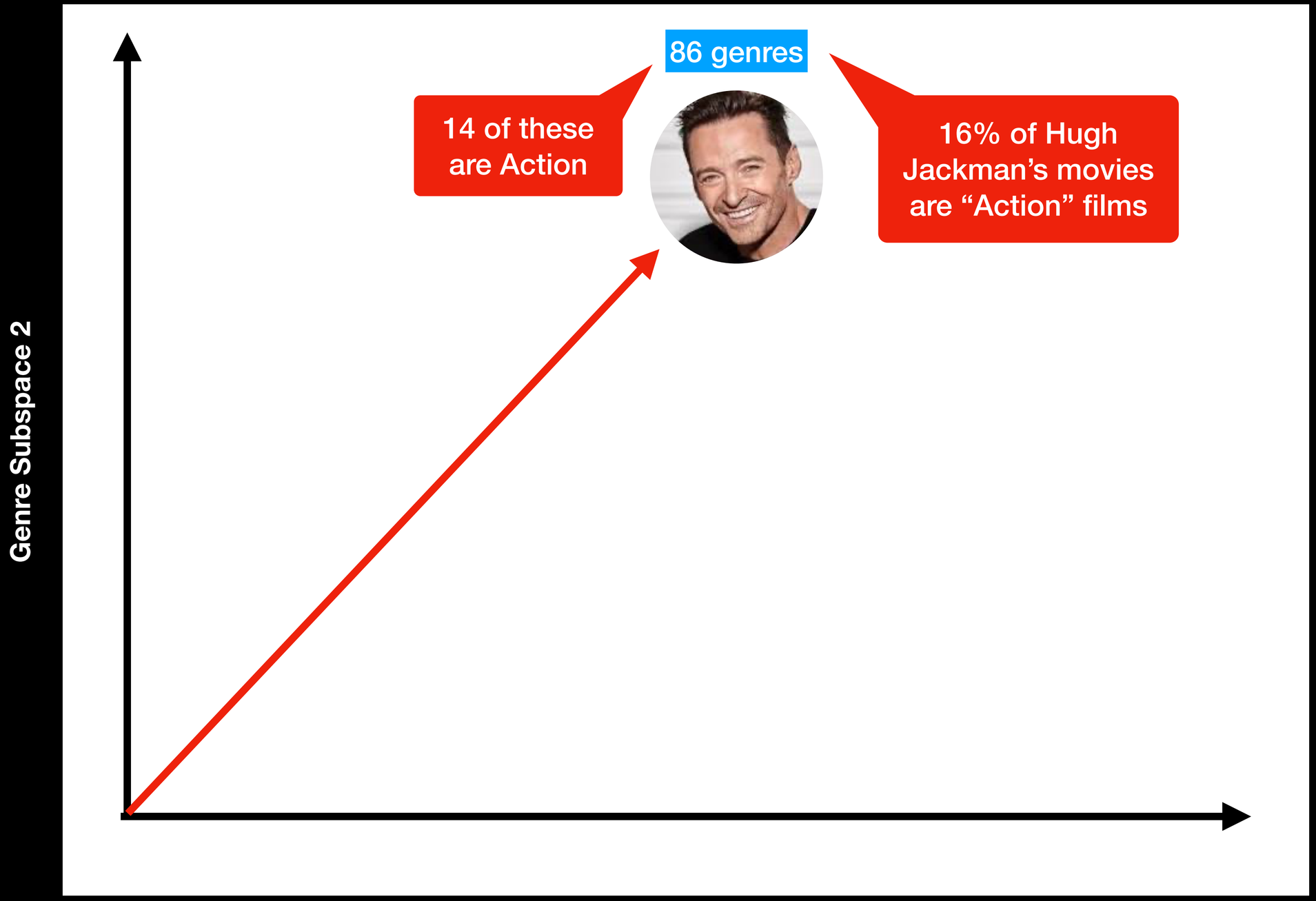
Part of a larger process concept of “Data Preprocessing”

In [19]:
```
actor_genre_df.head(20)
```
Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0	[illegible]	[illegible]	0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.	[illegible]	[illegible]	0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0	[illegible]	[illegible]	0.0
nm0005227	12.0	1.0	[illegible]	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0	[illegible]	0.0	0.0
[illegible]	[illegible]	[illegible]	[illegible]	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]	[illegible]	[illegible]	[illegible]	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]	[illegible]	[illegible]	[illegible]	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]	[illegible]	[illegible]	[illegible]	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
[illegible]2648	1.0	0.0	1.0	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
[illegible]	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]7	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
[illegible]0626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

What is an alternate way to interpret this data?

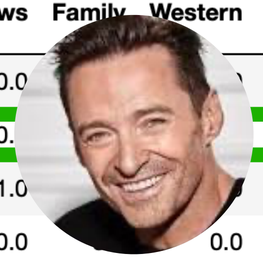


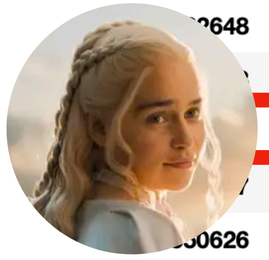

86 genres
14 of these are Action
16% of Hugh Jackman's movies are "Action" films
Genre Subspace 2
Genre Subspace 1

```
In [19]: actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0		0.0	0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.			0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0			0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0		0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
[illegible]648	1.0	0.0	1.0	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
[illegible]	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
[illegible]	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
[illegible]0626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								





Rescale the row to a consistent scale

In [58]:
```
df.divide(df.sum(axis=1), axis=0).head(10)
```

Out[58]:

$$\text{norm}_{row,col}(X) = \frac{|X_{row,col}|}{\sum_{j \in col} |X_{row,j}|}$$

	Comedy	Fantasy	Romance	Drama	Mystery	Thrill					...			N
nm0000212	0.250000	0.035714	0.214286	0.214286	0.035714	0.071429	0.035714	0.035714	0.071429	0.035714	...	0.000000	0.000000	0.0
nm0413168	0.081395	0.034884	0.058140	0.139535	0.058140	0.023256	0.162791	0.046512	0.069767	0.000000	...	0.000000	0.000000	0
nm0000630	0.112676	0.028169	0.084507	0.197183	0.028169	0.042254	0.056338	0.070423	0.014085	0.014085	...	0.042254	0.098592	0.0
nm0005227	0.400000	0.040000	0.080000	0.080000	0.000000	0.040000	0.040000	0.000000	0.000000	0.000000	...	0.040000	0.000000	0.040000	0.000
nm0864851	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.333333	0.000000	0.000000	0.000
nm0828288	0.119403	0.000000	0.000000	0.059701	0.044776	0.149254	0.014925	0.000000	0.044776	0.000000	...	0.522388	0.000000	0.000000	0.000
nm0933983	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.333333	0.000000	0.000000	0.000
nm0329491	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.333333	0.000000	0.000000	0.000
nm0000417	0.119048	0.023810	0.000000	0.190476	0.071429	0.095238	0.119048	0.000000	0.095238	0.000000	...	0.119048	0.000000	0.000000	0.000
nm0000603	0.062500	0.031250	0.062500	0.375000	0.062500	0.062500	0.000000	0.093750	0.062500	0.000000	...	0.031250	0.031250	0.000000	0.000

$$\text{norm}_{row,col}(X) = \frac{|X_{row,col}|}{\sum_{j \in col} |X_{row,j}|}$$

Referred to as the “L1 norm”

Interpretable as a probability

```
In [62]: df_norm.sum(axis=1).head(20)

Out[62]: nm0000212    1.0
         nm0413168    1.0
         nm0000630    1.0
         nm0005227    1.0
         nm0864851    1.0
         nm0828288    1.0
         nm0933983    1.0
         nm0329491    1.0
         nm0000417    1.0
         nm0000603    1.0
         nm0000457    1.0
         nm0452288    1.0
         nm0001002    1.0
         nm0001299    1.0
         nm0923529    1.0
         nm0936365    1.0
         nm0006763    1.0
         nm0007113    1.0
         nm0310173    1.0
         nm0412917    1.0
         dtype: float64
```

All actors' rows now sum to 1

In [68]:
```
#Gathering the genres for that actor
target_actor_ratings = df_norm.loc[target_actor_id]

#Generating distances from that actor to all the others
distances = scipy.spatial.distance.cdist(df_norm, [target_actor_ratings], metric="cosine")[:,0]

query_distances = list(zip(df.index, distances))
```

In [69]:
```
#Printing the top ten most similar actors to our target
for similar_actor_id, similar_genre_score in sorted(query_distances, key=lambda x: x[1], reverse=False)[:10]:
    print(similar_actor_id, actor_name_map[similar_actor_id], similar_genre_score, df.loc[similar_actor_id].sum())
```

```
nm0413168 Hugh Jackman 0.0 86.0
nm3592338 Emilia Clarke 0.0230383538713832 20.0
nm1663205 Sharlto Copley 0.03680695049665883 39.0
nm0000375 Robert Downey Jr. 0.04015740464249462 75.0
nm3772243 Theo James 0.04358329306959263 46.0
nm0262635 Chris Evans 0.04805589651668274 74.0
nm0881629 Karl Urban 0.05508543978437197 44.0
```

In [70]:
```
#Gathering the genres for that actor
target_actor_ratings = df_norm.loc[target_actor_id]

#Generating distances from that actor to all the others
distances = scipy.spatial.distance.cdist(df_norm, [target_actor_ratings], metric="euclidean")[:,0]

query_distances = list(zip(df.index, distances))
```

In [71]:
```
#Printing the top ten most similar actors to our target
for similar_actor_id, similar_genre_score in sorted(query_distances, key=lambda x: x[1], reverse=False)[:10]:
    print(similar_actor_id, actor_name_map[similar_actor_id], similar_genre_score, df.loc[similar_actor_id].sum())
```

```
nm0413168 Hugh Jackman 0.0 86.0
nm3592338 Emilia Clarke 0.07107305483691222 20.0
nm1663205 Sharlto Copley 0.08967075576182816 39.0
nm0000375 Robert Downey Jr. 0.09072324558361769 75.0
nm3772243 Theo James 0.09798239863947802 46.0
nm0881629 Karl Urban 0.10799068657869043 44.0
nm0262635 Chris Evans 0.10987243021473778 74.0
```

$$\text{L2 norm}_{row,col}(X) = \frac{x_{row,col}}{\sqrt{\sum_{j \in col}(x_{row,j})^2}}$$

Other norms as well

L2-norm ensures “unit” length

```
In [19]: actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Shor	Total income
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$156,000,000
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$100,000,000
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0	0.0	0.0	0.0	$100,000,000
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0	3.0	0.0	0.0	$100,000,000
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$100,000,000
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000+
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$70,000,000+
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$70,000,000
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$70,000,000
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$65,000,000
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0	$60,000,000
nm0462648	1.0	0.0	1.0	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	$59,000,000
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$55,000,000
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$50,000,000
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	$42,000,000
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0								

Will normalizing rows solve this problem?

```
In [19]: actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller							Western	Shor	Total income
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0							0.0	0.0	$156,000,000
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0							0.0	0.0	$100,000,000
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0							0.0	0.0	$100,000,000
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0	3.0	0.0	$100,000,000
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$100,000,000
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$75,000,000+
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$70,000,000+
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$70,000,000
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$70,000,000
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$65,000,000
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	$60,000,000
nm0462648	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	...	0.0	0.0	0.0	0.0	0.0	$59,000,000
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	$55,000,000
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	$50,000,000
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	$42,000,000
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0							
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0							
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0							

How can we transform this column into a more consistent range?

Rescale the column to a consistent scale

What portion of the maximum Total Income is this value?

What portion of the maximum Total Income is this value?

Total income
$156,000,000
$100,000,000
$100,000,000
$100,000,000
$100,000,000
$75,000,000+
$75,000,000
$75,000,000
$70,000,000+
$70,000,000
$70,000,000
$65,000,000
$60,000,000
$59,000,000
$55,000,000
$50,000,000
$42,000,000
$40,000,000+
$40,000,000
$40,000,000

min(

Total income
$156,000,000
$100,000,000
$100,000,000
$100,000,000
$100,000,000
$75,000,000+
$75,000,000
$75,000,000
$70,000,000+
$70,000,000
$70,000,000
$65,000,000
$60,000,000
$59,000,000
$55,000,000
$50,000,000
$42,000,000
$40,000,000+
$40,000,000
$40,000,000

)

max(

Total income
$156,000,000
$100,000,000
$100,000,000
$100,000,000
$100,000,000
$75,000,000+
$75,000,000
$75,000,000
$70,000,000+
$70,000,000
$70,000,000
$65,000,000
$60,000,000
$59,000,000
$55,000,000
$50,000,000
$42,000,000
$40,000,000+
$40,000,000
$40,000,000

)

Rescale this element:
(Element - min) / (max-min)

Total income
$156,000,000
$100,000,000
$100,000,000
$100,000,000
$100,000,000
$75,000,000+
$75,000,000
$75,000,000
$70,000,000+
$70,000,000
$70,000,000
$65,000,000
$60,000,000
$59,000,000
$55,000,000
$50,000,000
$42,000,000
$40,000,000+
$40,000,000
$40,000,000

```
In [19]: actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short	Total income
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$156,000,000
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$100,000,000
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0	0.0	0.0	0.0	$100,000,000
			3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0	3.0	0.0	0.0	$100,000,000
			0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$100,000,000
			0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000+
nm0940707	1.0	0.0		0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$75,000,000
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0								0.0	$70,000,000+
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0								0.0	$70,000,000
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0								0	$70,000,000
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0		$65,000,000
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0	$60,000,000
nm0462648	1.0	0.0	1.0	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	$59,000,000
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$55,000,000
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	$50,000,000
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	$42,000,000
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0								

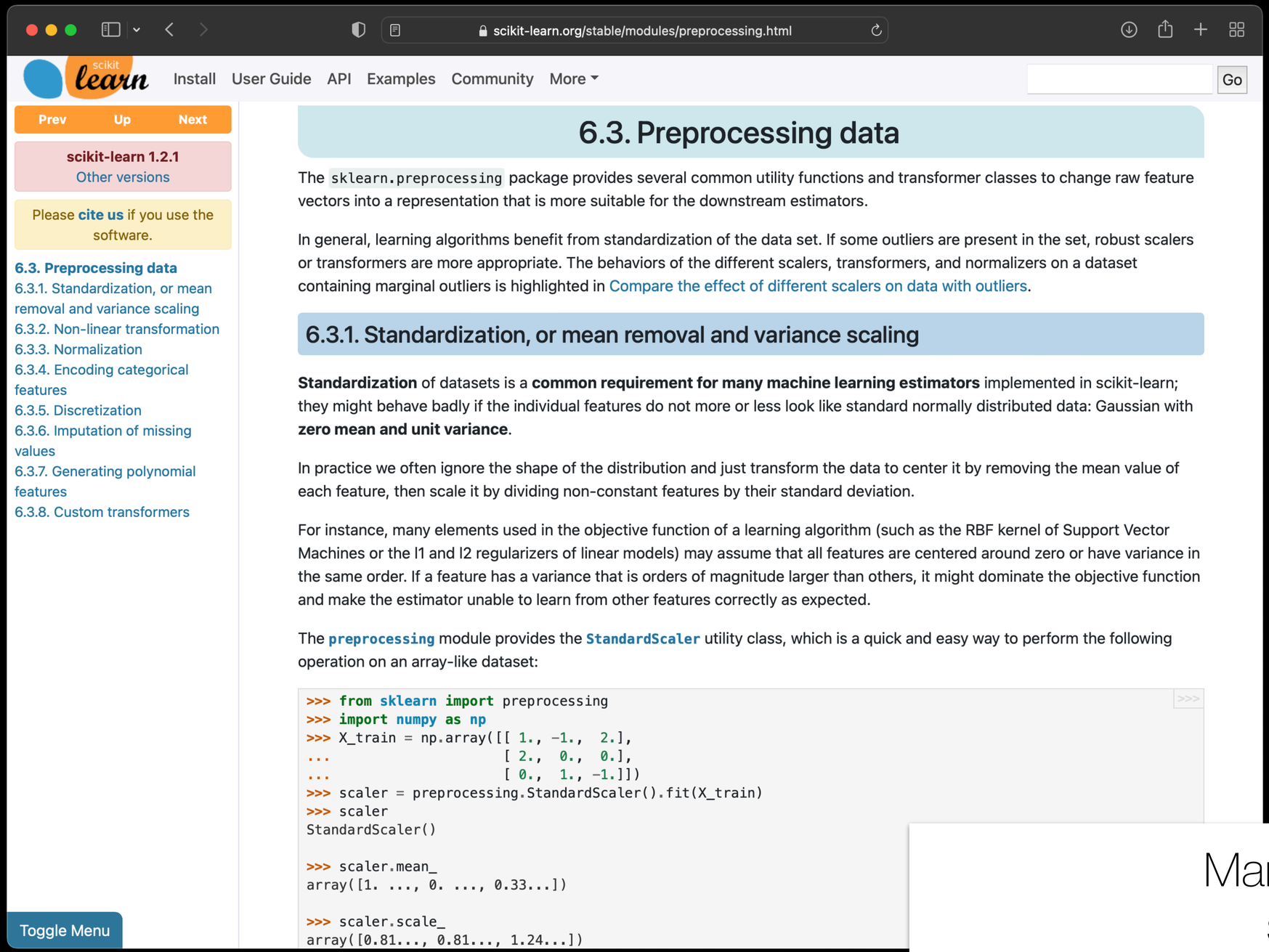
(x - min) / (max-min)
using min/max from Action

(x - min) / (max-min)
using min/max from
Total Income

Rescale every row this way

Minimum and maximum depend on current data

Why is this dependence problematic?


scikit-learn.org/stable/modules/preprocessing.html
scikit learn
Install
User Guide
API
Examples
Community
More
Go
Prev
Up
Next
scikit-learn 1.2.1
Other versions
Please cite us if you use the software.
6.3. Preprocessing data
6.3.1. Standardization, or mean removal and variance scaling
6.3.2. Non-linear transformation
6.3.3. Normalization
6.3.4. Encoding categorical features
6.3.5. Discretization
6.3.6. Imputation of missing values
6.3.7. Generating polynomial features
6.3.8. Custom transformers
Toggle Menu
6.3. Preprocessing data
The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.
In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate. The behaviors of the different scalers, transformers, and normalizers on a dataset containing marginal outliers is highlighted in Compare the effect of different scalers on data with outliers.
6.3.1. Standardization, or mean removal and variance scaling
Standardization of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.
In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.
For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the l1 and l2 regularizers of linear models) may assume that all features are centered around zero or have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.
The preprocessing module provides the StandardScaler utility class, which is a quick and easy way to perform the following operation on an array-like dataset:
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X_train = np.array([[1., -1., 2.],
... [2., 0., 0.],
... [0., 1., -1.]])
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> scaler
StandardScaler()
>>> scaler.mean_
array([1. ..., 0. ..., 0.33...])
>>> scaler.scale_
array([0.81..., 0.81..., 1.24...])
Many methods for standardization

This Week's Learning Objectives

Describe at least three ways data-hygiene problems may manifest

Define two kinds of "normalization" for numeric data

Convert categorical data to numeric data

This Week's Learning Objectives

Describe at least three ways data-hygiene problems may manifest

Define two kinds of "normalization" for numeric data

Convert categorical data to numeric data

How do you deal with categorical features?

movie1 = {Genre1, Genre2, Genre3}

movie2 = {Genre3, Genre4, Genre5}

d(movie1, movie2) = ?

d(movie1, movie2) = 1- Jaccard(m1,m2)

But what if we mix categorical and numeric features?

```
In [19]: actor_genre_df.head(20)
```

Out[19]:

	Comedy	Fantasy	Romance	Action	Crime	Adventure	Mystery	Thriller	Drama	Biography	...	Sport		News	Family	Western	Short
nm0000212	16.0	3.0	16.0	5.0	4.0	2.0	5.0	3.0	16.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0413168	8.0	3.0	6.0	14.0	6.0	11.0	5.0	2.0	13.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0000630	10.0	2.0	6.0	4.0	1.0	2.0	2.0	4.0	17.0	6.0	...	4.0	1.0	1.0	0.0	0.0	0.0
nm0005227	12.0	1.0	3.0	2.0	0.0	3.0	0.0	1.0	5.0	1.0	...	1.0	0.0	0.0	3.0	0.0	0.0
nm0697338	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm1300519	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0940707	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0625977	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0792032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0496571	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2868805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm2866192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0
nm0462648	1.0	0.0	1.0	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
nm0000953	6.0	0.0	0.0	1.0	3.0	0.0	0.0	2.0	9.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0001782	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0
nm0005077	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
nm0550626	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0177016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
nm0907480	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0								

Age

. . .

Already done something like this

Convert the set space into a binary vector, where 1 denotes that the element has that feature in its set

Called a “**one-hot** encoding”

What's the space of eye color?

Man, 37, 225lbs, blue eyes…

Person	Eye Color
Person_1	Blue
Person_2	Green
Person_3	Blue
Person_4	Brown

Person	Eye Color
Person_1	1
Person_2	2
Person_3	1
Person_4	3

What's the space of eye color?

Man, 37, 225lbs, blue eyes…

Person	Eye Color
Person_1	Blue
Person_2	Green
Person_3	Blue
Person_4	Brown

Blue = 1
Green = 2
Brown = 3

Implies people with blue eyes are "nearer" to those with green eyes than to those with brown eyes

blue eyes…

Person	Eye Color
Person_1	Blue
Person_2	Green
Person_3	Blue
Person_4	Brown

Blue = 1
Green = 2
Brown = 3

(Blue - green)^2 =
(1 - 2)^2 = 1 >
(blue - brown)^2 =
(1-3)^2 = 4

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

What's the space of eye color?

Man, 37, 225lbs, blue eyes…

Instead, create multiple columns of binaries, yes/no for every category

Man, 37, 225lbs, blue eyes…

Need to normalize columns

	Woman	Man	Transgender	Nonbinary	Age	Weight	Eye_Blue	Eye_Gr	Eye_Br
Cody	0	1	0	0	37	225	1	0	0

This Week's Learning Objectives

Describe at least three ways data-hygiene problems may manifest

Define two kinds of "normalization" for numeric data

Convert categorical data to numeric data

This Week's Learning Objectives

- Describe at least three ways data-hygiene problems may manifest
- Define two kinds of "normalization" for numeric data
- Convert categorical data to numeric data

Questions?

Prof. Cody Buntain | @codybuntain | cbuntain@umd.edu
Director, Information Ecosystems Lab