Big Data Engineer

Test task

D. Makarevich, 20.11.2023

Sources & Goal

We have 3 datasets:

- facebook_dataset.csv
- google_dataset.csv
- website_dataset.csv

Those datasets contain basic information about companies (eg. company name, phone number, adress, category, etc) from different sources. We want to create a 4th dataset that joins the information in all of those 3 datasets with a high accuracy

Cleaning and data preparation

- Convert categories from string to list of 'category' items

eg. "Sports | Baseball" → ['sports', 'baseball']

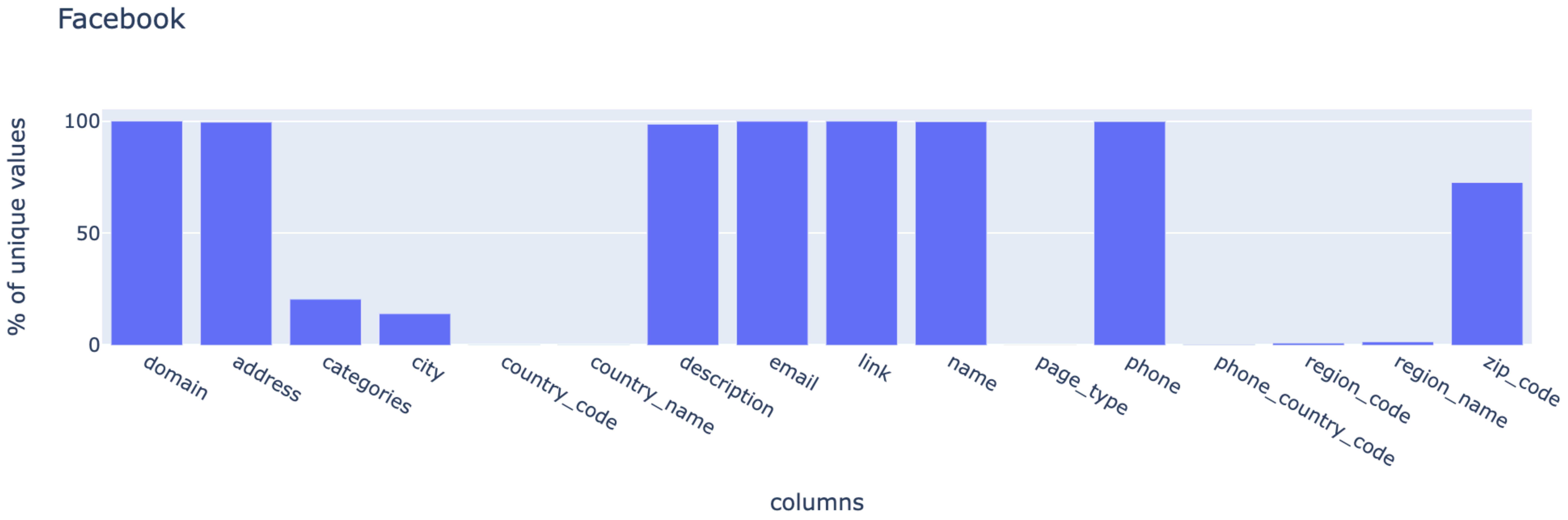
- Convert categories, adress, names, countries & regions to lowercase

eg. "England" → "england'

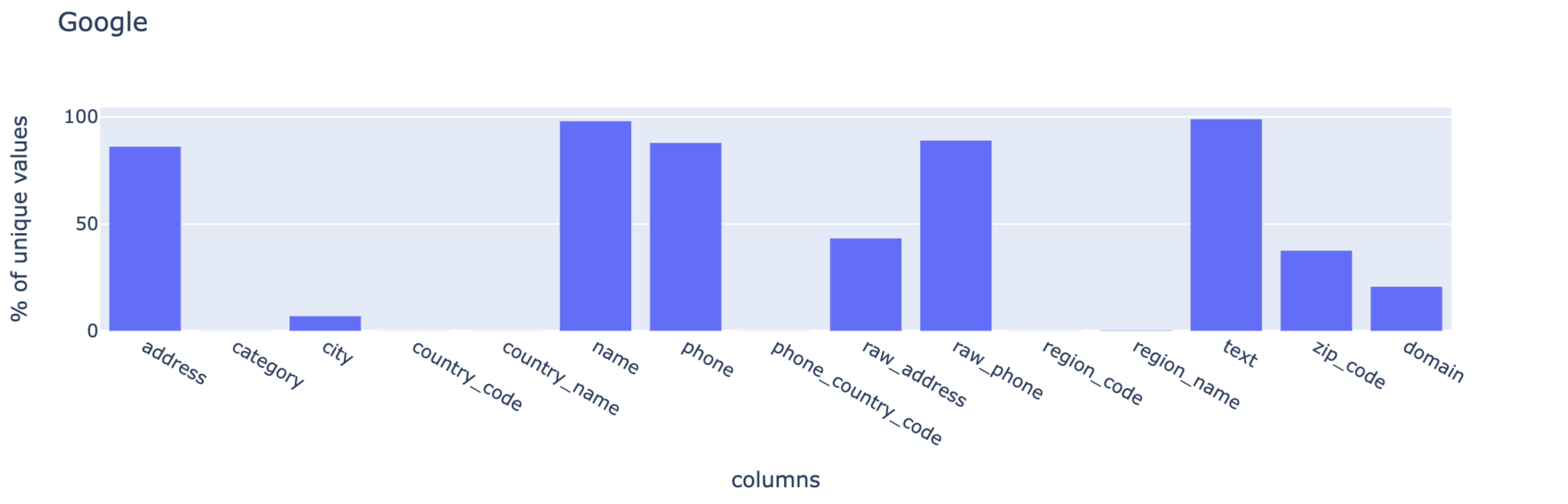
- Clean columns - delete symbols such as "+", "," , "-", "()", etc
- Clean company names - delete all sufixes such as 'Inc.' or 'Ltd.' and special characters
- Remove spaces in start and end of column values
- Bring all columns to the same name in difference datasets(ex. in website dataset ***site_name -> name***)
- Split and explode category column for Facebook dataset

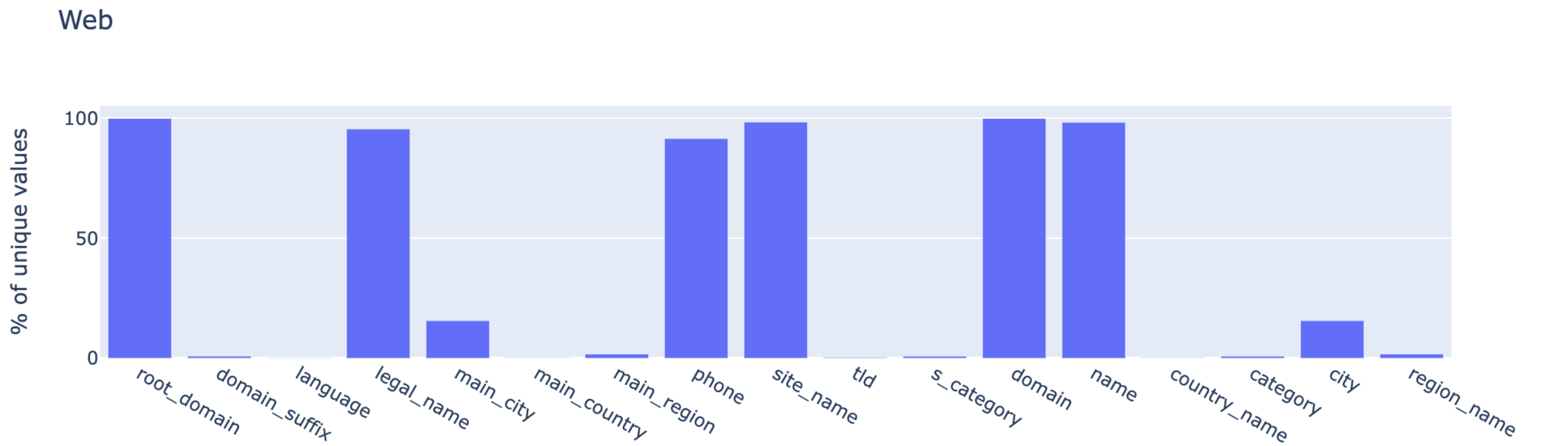
Analysis and Visualisation

Percentage of unique items in the FB dataset



Google and Website





What we have?

- Initially, my idea was to use **domain** as the key for the connection, but now we see that it is repeated many times for the Google dataset. Now we will use **name column**

Join datasets

We have 310k+ rows in ggDF and 63k rows in fbDF, but only **5991!** after join. Simple join is not suitable we are losing too much data

columns

```
[11]: #most are Category, Address(country, region...), Phone, Company names.
print(f"Google dataframe: {ggDF.count()}")
print(f"Facebook dataframe: {fbDF.count()}")
print(f"Inner join count: {ggDF.join(fbDF, on= ['category', 'country_name', 'region_name', 'phone', 'name']).count()}")
```

```
Google dataframe: 310854
Facebook dataframe: 63273

Inner join count: 5991
```

only 5991 rows from was joined, it bad result, we need investigate and improve it

How to Joins datasets?

fuzzymatcher & levenshtein

- If we can't join the data directly, we need to use either. an approximate match in the names, and *fuzzymatcher & levenshtein* can help us. *Fuzzymatcher* we can use for pandas data or *levenshtein* for spark data frames.

- Since our task is to get the most complete dataset without empty elements during join, we can take *facebook* and make a left join with W*ebsite* to it using *fuzzymatcher*(it doesn't make sense to use *Google* dataset at this stage, my tests have shown that performance drops very much)

```
Website names: 60934
Google names: 310854
Facebook names: 63273
Website unique names: 59365
Google unique names: 304712
Facebook unique names: 34853
```

Results after join

I tested and decided to filter rows where *best_match_score* > 0.1(*best_match_score* we get from *fuzzymatcher*) . After join we got 34712 rows, instead of 5991 in direct join!

```
]: joined_fb_wb[["best_match_score", "wb_name", "fb_name"]].head(10)
```

]:

	best_match_score	wb_name	fb_name
0	0.095517	chandler	chandler associates architecture
53	0.095517	chandler	chandler associates architecture
106	0.330651	aha scientific	aha scientific repair services
160	0.524375	healthability	healthability
161	0.524375	healthability	healthability
162	0.524375	healthability	healthability
163	0.707699	house of thunder	house of thunder
164	0.707699	house of thunder	house of thunder
165	0.239986	high irits distillery	broken irits distillery
166	0.839774	apex glass and mirror	apex glass and mirror

Join with Google dataset and filter

Now we join *Facebook&Website* dataset from previous step with *Google* dataset. Number of output rows after filtering by *best_match_score* is **32620**. Next step will be validate output

```
.9]: joined_fb_wb_gg = fuzzymatcher.fuzzy_left_join(filtered_fb_wb.drop(["best_match_score", "__id_left","__id_right"], axis=1),
     joined_fb_wb_gg.head()
```

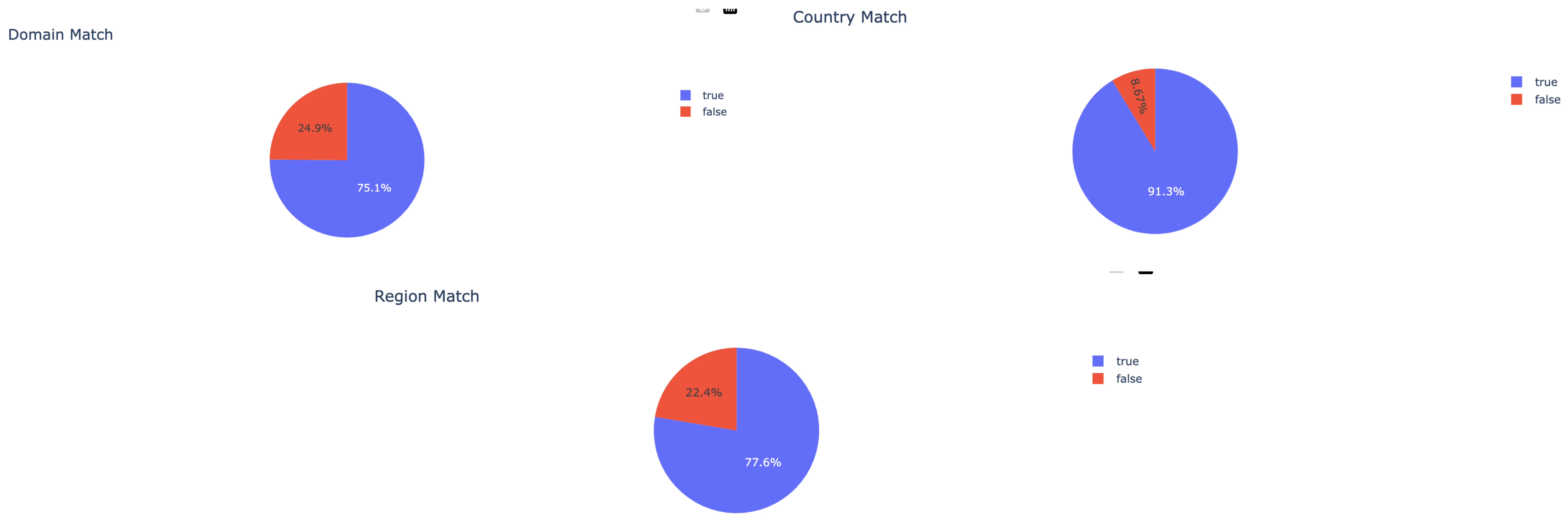
.9]:

	best_match_score	__id_left	__id_right	fb_name	fb_category	fb_phone	fb_domain	fb_country_name	fb_region_nar
0	1.294894	0_left	263059_right	fourth dimension orthodontics & craniofacial o...	clinics - surgeons & physicians	+19729472000	4dorthodontics.com	united states	tex
1	1.294894	1_left	263059_right	fourth dimension orthodontics & craniofacial o...	orthodontists	+19729472000	4dorthodontics.com	united states	tex
2	0.768896	2_left	21477_right	luc fontaine courtiers immobiliers re/max actif	travel agencies	+15142475055	lucfontaine.com	canada	queb
3	0.768896	3_left	21477_right	luc fontaine courtiers immobiliers re/max actif	real estate - agents & managers	+15142475055	lucfontaine.com	canada	queb
4	0.768896	4_left	21477_right	luc fontaine courtiers immobiliers re/max actif	real estate - agents & managers	+15142475055	lucfontaine.com	canada	queb

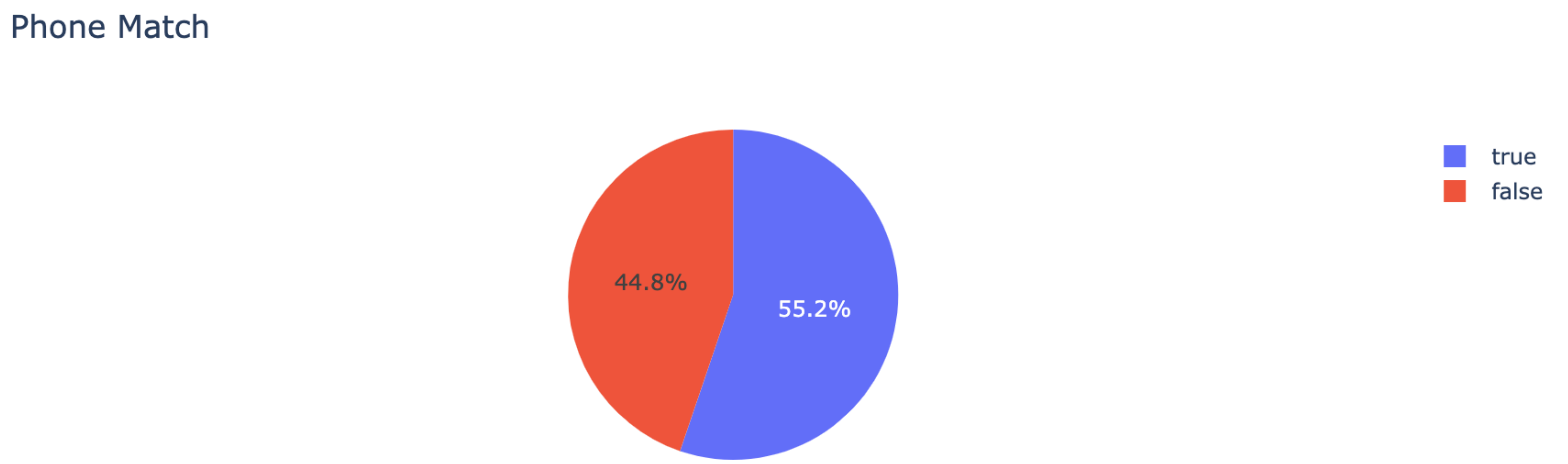
5 rows × 21 columns

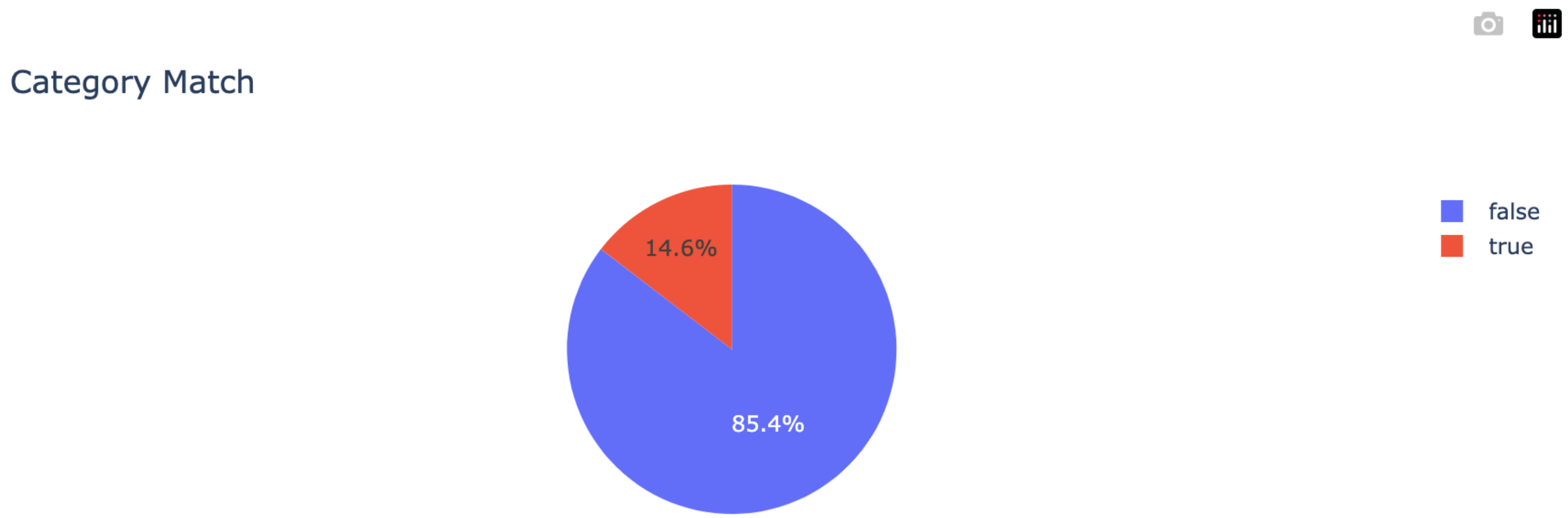
Validation and Visualisation

On these charts we can see how match Domain, Country and Region in final dataset



Phone and Category match





Why we not using levenshtein with PySpark

- Despite the fact that PySpark can use all the cores of my computer (there are 10 of them), it works much slower than *fuzzymatcher with pandas(~3mins in pandas vs ~20mins in spark)*

```python
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

merged = (fbDF_subset.join(wbDF_subset, levenshtein(fbDF_subset.fb_name, wbDF_subset.wb_name) < 6, "left")
 .select(fbDF_subset.fb_name, wbDF_subset.wb_name, levenshtein(fbDF_subset.fb_name, wbDF_subset.wb_name))
)
merged.show(10 ,False)
#merged.count()
```

```
+--------------------------------+-------------+-----------------------------+
|fb_name                         |wb_name      |levenshtein(fb_name, wb_name)|
+--------------------------------+-------------+-----------------------------+
|chandler associates architecture|NULL         |NULL                         |
|chandler associates architecture|NULL         |NULL                         |
|aha scientific repair services  |NULL         |NULL                         |
|healthability                   |healthbay    |5                            |
|healthability                   |health bloom |5                            |
|healthability                   |healthability|0                            |
|healthability                   |healthbay    |5                            |
|healthability                   |health bloom |5                            |
|healthability                   |healthability|0                            |
|healthability                   |healthbay    |5                            |
+--------------------------------+-------------+-----------------------------+
only showing top 10 rows
```

Questions and Answers

1. What column will you use to join?

   I decided use *name* column for join, this column is most suitable for the role of a unique key

2. If you have data conflicts once you join, which one do you believe?

   In cases where you are joining data from multiple sources, you might decide to prioritize one data source over another. You can choose to believe the data from the primary or most trusted source.

3. If you have very similar data, what information will you keep?

   When dealing with very similar data, the choice of what information to keep should consider factors such as data source reliability, data freshness, completeness, quality, user preferences, unique identifiers, and data context, with manual review as a possible resolution in cases of uncertainty.

Conclusion

- We analysed three datasets, cleaned them and joined them with
- Different approaches were used for the join and they were compared
- Not always the initially chosen tool is completely suitable for the task, in my case spark was unnecessary and it was possible to do with pandas, since we did not get any performance gain.
- We have received a dataset over which we can perform additional cleaning, validation, saturation and use in the further analysis

Thank you for your attention

- Makarevich Dmitry, Big Data Engineer