Companies classification using LLMs

Nikolaos Episkopos

Topics:

- SQL databases
- Exploratory Data Analysis (EDA)
- Data Manipulation
- Machine Learning

1. Introduction

The goal of this projectis to use existing information on some companies in order to extract some useful insights on the data features and their properties, and use these insights to train an Artificial Intelligence model to classify these companies based on their areas of business activity. Towards this goal we used information stored in the provided Database, provided as an SQLite database (.sqlite file), which contains two Tables of information on companies. Then we used Python with some external data-related modules like Pandas, Matplotlib, and scikit-learn to perform some exploratory data analysis, which can provide us with useful information on the data features.

For example, correlations between data features can be uncovered to suggest whether a feature can influence another and whether a feature correlates with companies' classification. Finally, given the insights we extracted during the EDA phase, we decided upon which data features to choose for training a ML classifier which can accurately predict the business categorization of a company based on said features.

2. Part 1 – SQL queries

We developed three SQL queries to retrieve the requested information from the SQLite database. First, we executed the queries to observe their outcome and verify their correctness. Afterwards, we run the equivalent EXPLAIN QUERY PLAN queries to obtain a high-level description of the plan that SQLite uses to perform the SQL queries. Finally, after observing both the queries and their corresponding plans, we decided to create two indexes to accelerate the query execution. These indexes were created for the "current employee estimate" and the "countries" columns. After both indexes were created, we re-run the EXPLAIN QUERY PLAN queries to determine whether these indexes use employed in the respective query plans or not. The result verified that both indexes were used in the queries and in fact all three queries used exactly one index in their plans.

3. Part 2 – Data Integration and Database Insertion

We used two different approaches to merge the two database tables.

Approach 1: We created a **view named "Company"** which was created through an SQL query which joined the two tables on the "CompanyName" column. This approach was the simplest and easiest to implement and did not significantly increase the database file size.

Approach 2: We used a combination of SQL queries and Python code using Pandas. Initially, we created two SQL queries which retrieved data from each table and stored it in a corresponding DataFrame. Then we used Pandas' merge function to perform DataFrame a left join on the "CompanyName" column, resulting in a single final DataFrame which contains all database information. Then, we stored this DataFrame back to the Database as a **new table**, under the name **"Company"** using yet another SQL statement.

Personally, I believe the **1st approach is much easier, faster and more resource efficient**. However, since the 2nd approach was implemented last, merged **"Company" was stored as a table** instead of a view.

After the merge was complete, we modified and executed one of the SQL queries presented in Part 1, to verify the correctness of the merge.

4. Part 3 – Exploratory Data Analysis (EDA)

Initially, we executed an SQL query to retrieve all database information from the Database, the result of which was stored in a DataFrame. Then we removed unnecessary columns, like a numerical column which serves as a unique row ID. Then, we renamed the columns, converting their names to lowercase and replacing whitespace characters with underscores.

	col_name	col_dtype	num_distinct_values	min_value	max_value	median_no_na	average_no_na	average_non_zero	null_present	nulls_num	non_nulls_num	distinct_values
0	company_name	object	5451314	N/A	N/A	N/A	N/A	N/A	NaN	0	5522803	{'independent consultant': 40, 'private practi...
1	website	object	5474762	N/A	N/A	N/A	N/A	N/A	NaN	0	5522803	{'nordalps.com': 12, 'play-cricket.com': 8, 'v...
2	year_founded	float64	232	1451.0	2103.0	2009.0	2001.904919	2001.904919	NaN	2175920	3346883	{2015.0: 219441, 2014.0: 218255, 2013.0: 20907...
3	industry	object	148	N/A	N/A	N/A	N/A	N/A	NaN	112775	5410028	{'information technology and services': 369512...
4	size_range	object	8	N/A	N/A	N/A	N/A	N/A	NaN	0	5522803	{'1 - 10': 4130680, '11 - 50': 988537, '51 - 2...
5	locality	object	90186	N/A	N/A	N/A	N/A	N/A	NaN	1693997	3828806	{'london, greater london, united kingdom': 690...
6	country	object	236	N/A	N/A	N/A	N/A	N/A	NaN	1566173	3956630	{'united states': 1767278, 'united kingdom': 3...
7	linkedin_url	object	5522803	N/A	N/A	N/A	N/A	N/A	NaN	0	5522803	{'linkedin.com/company/ibm': 1, 'linkedin.com/...
8	current_employee_estimate	int64	5189	0	274047	2	16.103707	16.103707	NaN	0	5522803	{1: 1776865, 0: 950542, 2: 789268, 3: 418838, ...
9	total_employee_estimate	int64	8223	1	716906	3	37.502976	37.502976	NaN	0	5522803	{1: 1740935, 2: 862262, 3: 491918, 4: 325942, ...
10	category	object	13	N/A	N/A	N/A	N/A	N/A	NaN	5449355	73448	{'Professional Services': 7279, 'Healthcare': ...
11	homepage_text	object	71397	N/A	N/A	N/A	N/A	N/A	NaN	5450023	72780	{' ': 174, ' Affordable, Relia...
12	h1	object	43915	N/A	N/A	N/A	N/A	N/A	NaN	5476420	46383	{'Home': 628, 'Welcome': 185, 'Affordable, Rel...
13	h2	object	50481	N/A	N/A	N/A	N/A	N/A	NaN	5469988	52815	{'Home': 77, 'Contact Us': 76, 'Contact Us#sep...
14	h3	object	41916	N/A	N/A	N/A	N/A	N/A	NaN	5478389	44414	{'Safe Payments By Adyen#sep#Fast Domain Trans...
15	nav_link_text	object	46250	N/A	N/A	N/A	N/A	N/A	NaN	5475006	47797	{'home,more': 155, ',(888) 694-6735,[email pro...
16	meta_keywords	object	22990	N/A	N/A	N/A	N/A	N/A	NaN	5499219	23584	{'web hosting, provider, php hosting,web hosti...
17	meta_description	object	65309	N/A	N/A	N/A	N/A	N/A	NaN	5456364	66439	{'See related links to what you are looking fo...

Afterwards, we replaced whitespace / empty strings with NaN, since they offer no useful information. Then we performed some missing values ratio analysis per dataset feature.

Feature	Missing values ratio
company_name	0.000000
website	0.000000
year_founded	0.393988
industry	0.020420
size_range	0.000000
locality	0.306728
country	0.283583
linkedin_url	0.000000
current_employee_estimate	0.000000
total_employee_estimate	0.000000
category	0.986701
homepage_text	0.986937
h1	0.991602
h2	0.990437
h3	0.991958
nav_link_text	0.991346
meta_keywords	0.995730
meta_description	0.987970

We can see that the ***"Category"*** feature which is the target / label / category feature contains meaningful values in less than 2% of its records, since the rest of the data points has missing values for the ***"Category"*** feature. This means that **less than 2%, i.e. 73,448 out of 5,522,803 the dataset records are useful** for keeping and actually using for training a Machine Learning model. Therefore, we were free to remove said points from the dataset and study features statistics on the reduced dataset. After this operation, the updated statistics looked like below.

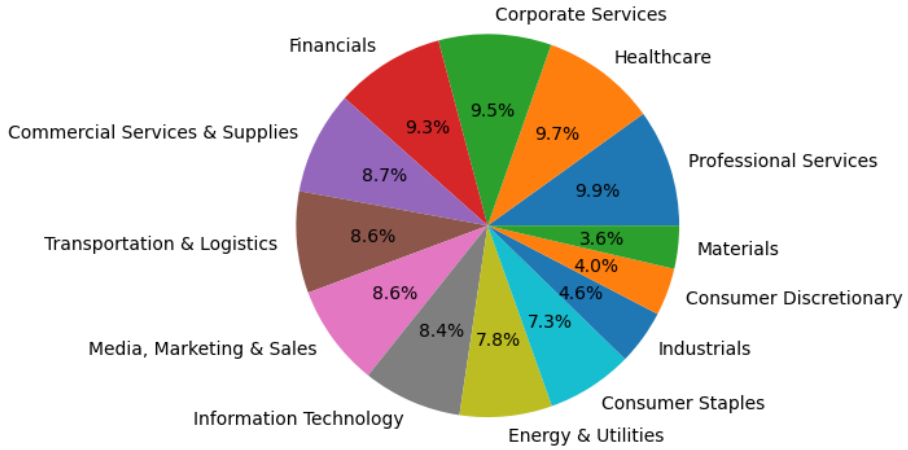
	col_name	col_dtype	num_distinct_values	min_value	max_value	median_no_na	average_no_na	average_non_zero	null_present	nulls_num	non_nulls_num	distinct_values
0	company_name	object	73448	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'cognizant technology solutions': 1, 'shareop...
1	website	object	73448	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'cognizant.com': 1, 'shareoptic.com': 1, 'mp-...
2	year_founded	float64	199	1804.0	2018.0	2005.0	1996.834138	1996.834138	NaN	25830	47618	{2010.0: 2201, 2014.0: 2122, 2012.0: 2070, 201...
3	industry	object	107	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'information technology and services': 3449, ...
4	size_range	object	8	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'1 - 10': 50783, '11 - 50': 15404, '51 - 200'...
5	locality	object	11012	N/A	N/A	N/A	N/A	N/A	NaN	1724	71724	{'london, greater london, united kingdom': 156...
6	country	object	7	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'united states': 48615, 'united kingdom': 115...
7	linkedin_url	object	73448	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'linkedin.com/company/cognizant': 1, 'linkedi...
8	current_employee_estimate	int64	908	0	122031	2	27.645926	27.645926	NaN	0	73448	{1: 19505, 2: 10226, 0: 8114, 3: 6155, 4: 4177...
9	total_employee_estimate	int64	1320	1	210020	4	59.457317	59.457317	NaN	0	73448	{1: 17381, 2: 9824, 3: 6270, 4: 4379, 5: 3241,...
10	category	object	13	N/A	N/A	N/A	N/A	N/A	NaN	0	73448	{'Professional Services': 7279, 'Healthcare': ...
11	homepage_text	object	71329	N/A	N/A	N/A	N/A	N/A	NaN	1301	72147	{' Affordable, Reliable Web Hosti...
12	h1	object	43915	N/A	N/A	N/A	N/A	N/A	NaN	27065	46383	{'Home': 628, 'Welcome': 185, 'Affordable, Rel...
13	h2	object	50481	N/A	N/A	N/A	N/A	N/A	NaN	20633	52815	{'Home': 77, 'Contact Us': 76, 'Contact Us#sep...
14	h3	object	41916	N/A	N/A	N/A	N/A	N/A	NaN	29034	44414	{'Safe Payments By Adyen#sep#Fast Domain Trans...
15	nav_link_text	object	46250	N/A	N/A	N/A	N/A	N/A	NaN	25651	47797	{'home,more': 155, ',(888) 694-6735,[email pro...
16	meta_keywords	object	22990	N/A	N/A	N/A	N/A	N/A	NaN	49864	23584	{'web hosting, provider, php hosting,web hosti...
17	meta_description	object	65309	N/A	N/A	N/A	N/A	N/A	NaN	7009	66439	{'See related links to what you are looking fo...

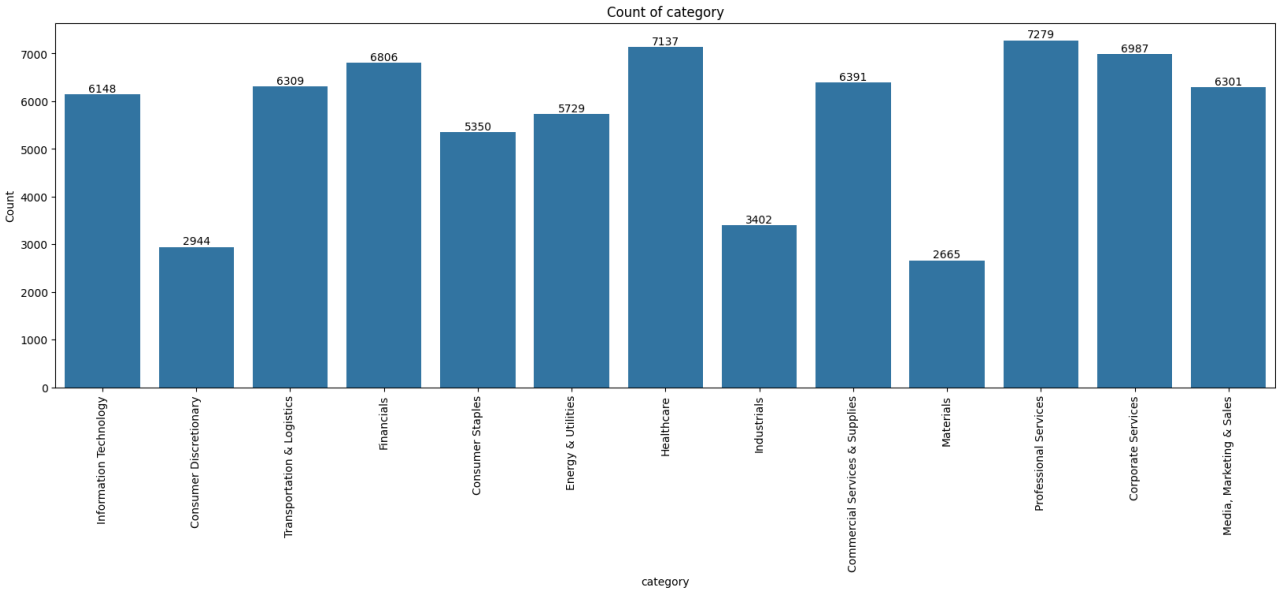
Feature	Missing values ratio
company_name	0.000000
website	0.000000
year_founded	0.351677
industry	0.000000
size_range	0.000000
locality	0.023472
country	0.000000
linkedin_url	0.000000
current_employee_estimate	0.000000
total_employee_estimate	0.000000
category	0.000000
homepage_text	0.017713
h1	0.368492
h2	0.280920
h3	0.395300
nav_link_text	0.349240
meta_keywords	0.678902
meta_description	0.095428

The “bad thing” that happened now is that **we had to give up on the 98.67% of the original data, i.e. 5,449,355 rows**. From this point on we used the reduced dataset for the rest of the EDA.

Target / category feature exploration

Afterwards, we performed some analysis on the ***"Category"*** feature, since this will be the classification target.



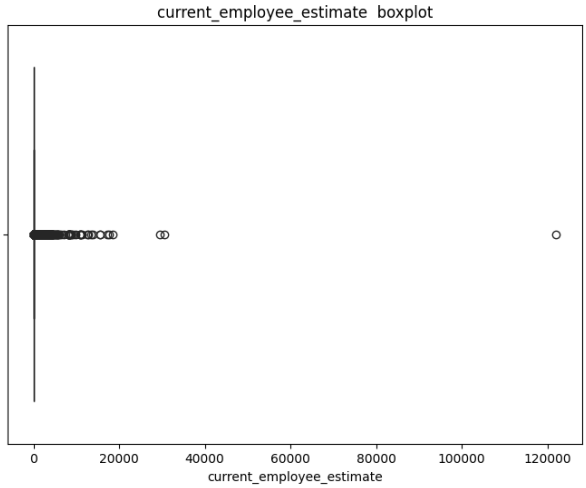


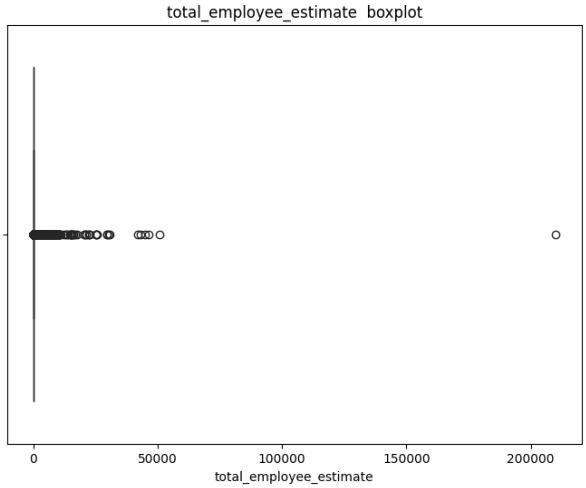
We can see that during the Machine Learning phase, we will have to deal with the class imbalance problem, since the Materials class consists only of 2,665 samples compared to the 7,279 samples of the Professional Services class.

Numeric features exploration

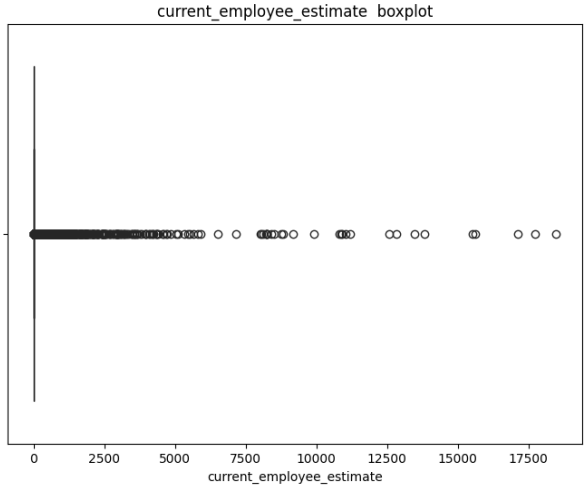
Afterwards, we performed some exploration on the numeric features of the dataset. We focused only on the numeric features whose percentage of missing values is lower than 20%, since 20% missing values is too high anyway and we might be missing on important information. After applying this constraint, the numerical features we considered were the ***"Current employee estimate"*** and the ***"Total employee estimate"***.

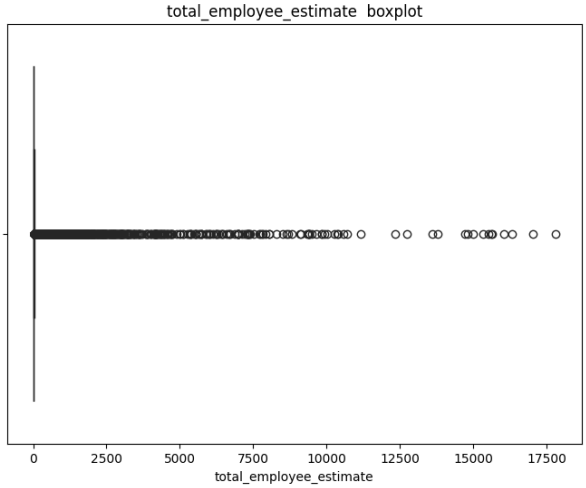
We used box plots to determine the concentration of values of these features, alongside their respective outliers.

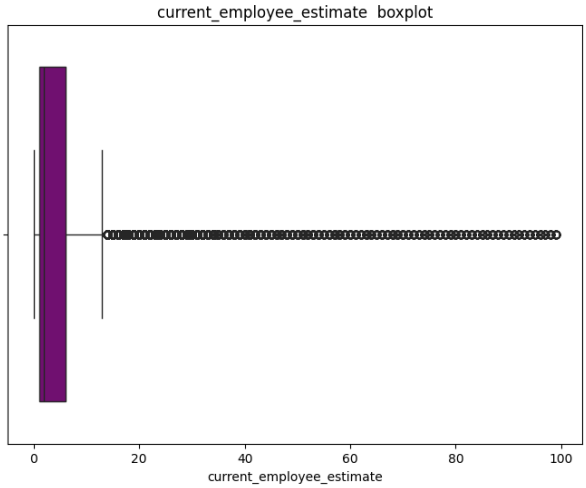


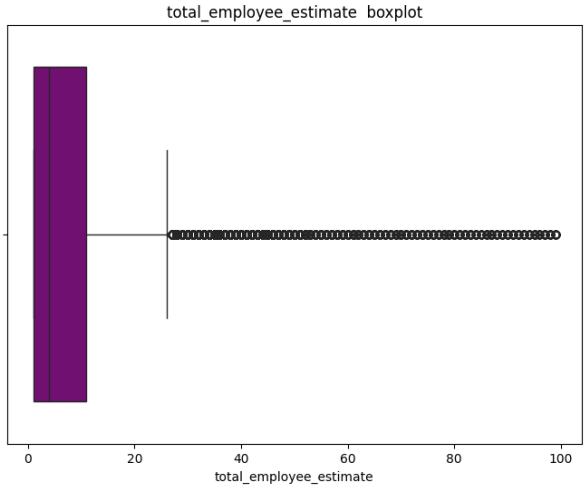


Since these plots do not provide us with any useful insight, we zoomed-in to acquire a clearer view.



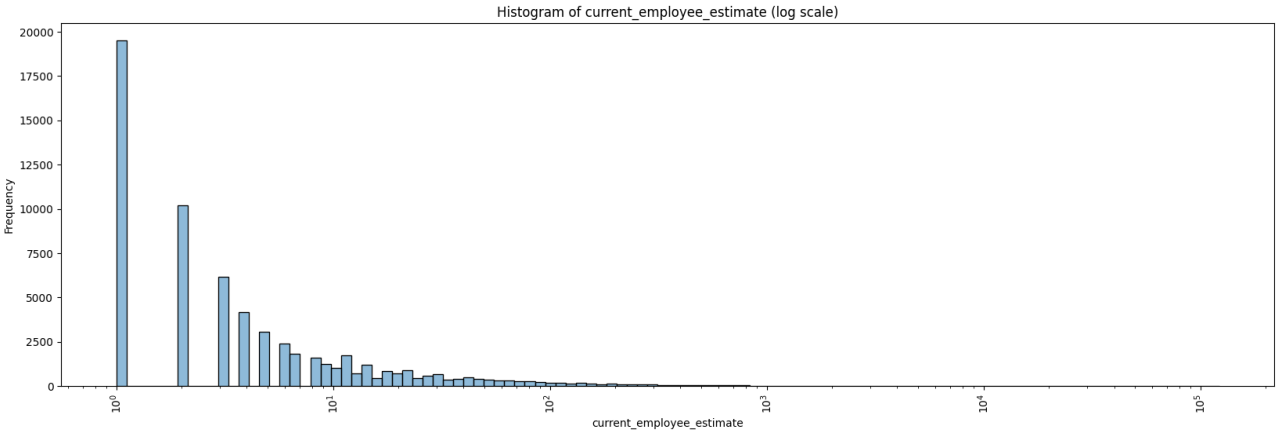


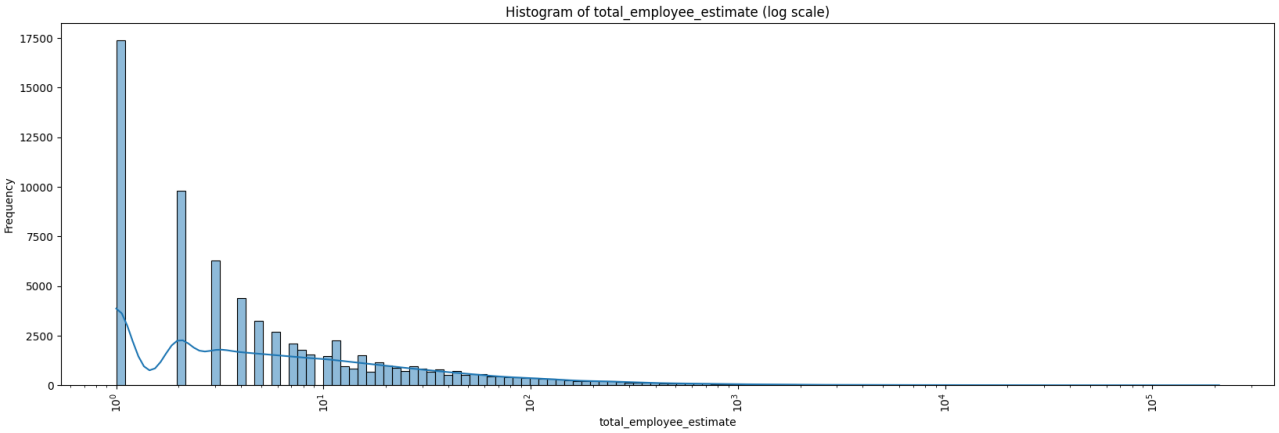


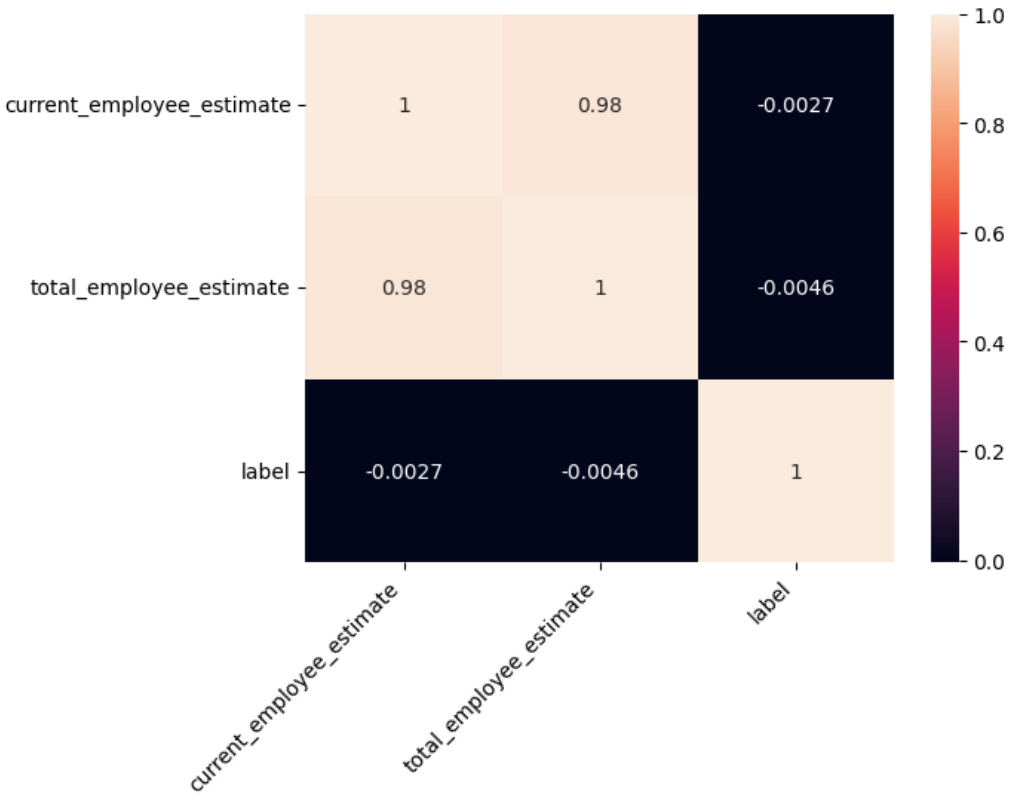


Finally, we can see that the vast majority of values in the numerical features is concentrated around zero. Our intuition tells us that these features are most likely highly correlated but due to the distribution of their values, they most likely are unsuitable in providing us with any useful insights.

This provides an better outlook of the distribution of values of the two numeric features.





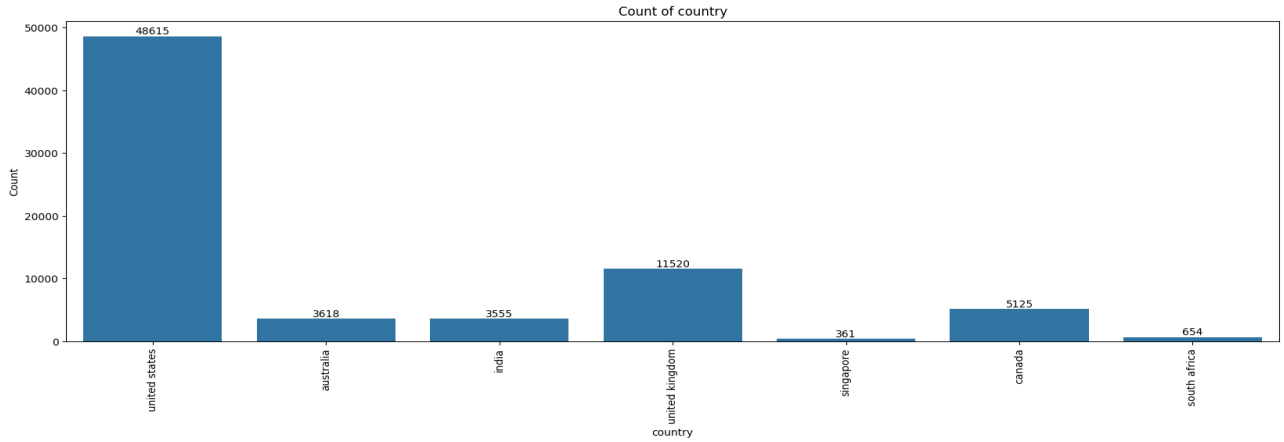


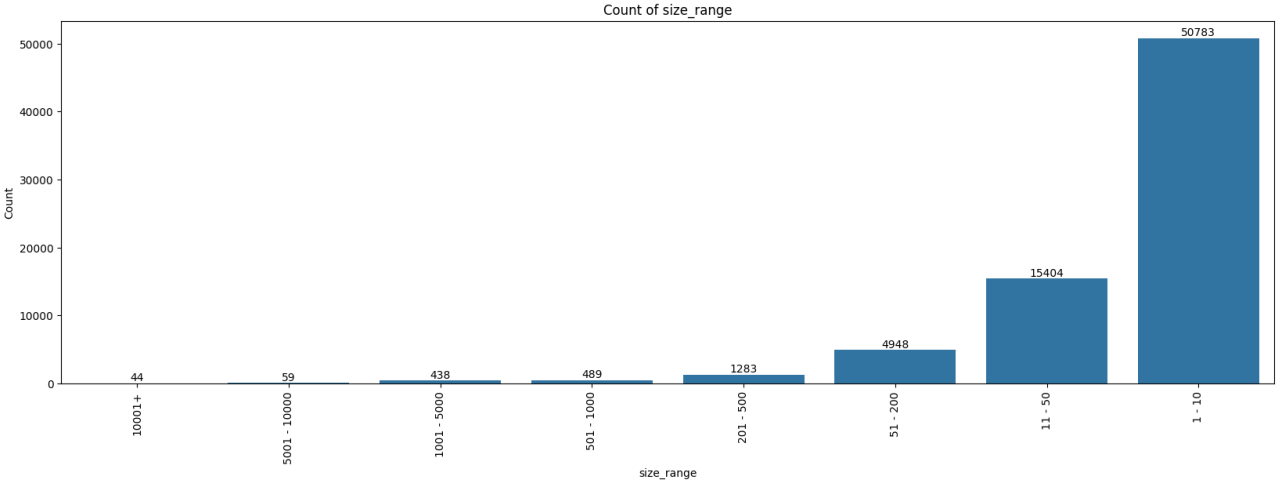
From the correlation matrix, we can also see that although these two numerical features are highly correlated, as expected, their correlation with the sample category is really poor.

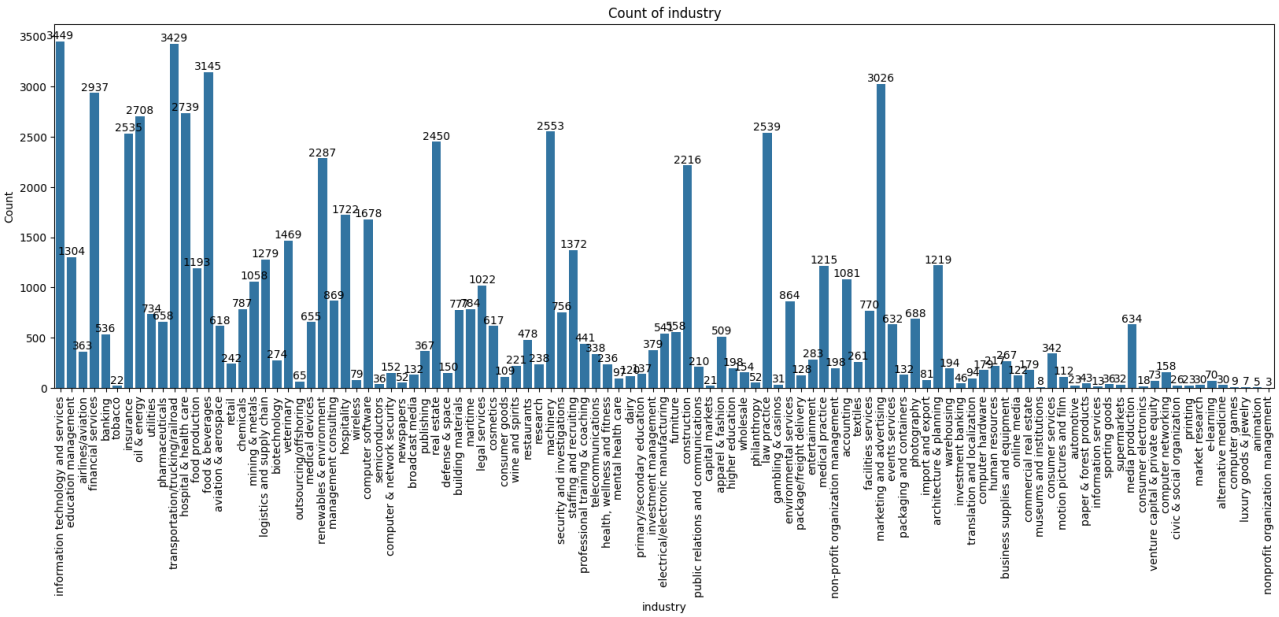
Considering all the above information, we decided against including these numerical features in training our classifier.

Categorical features exploration

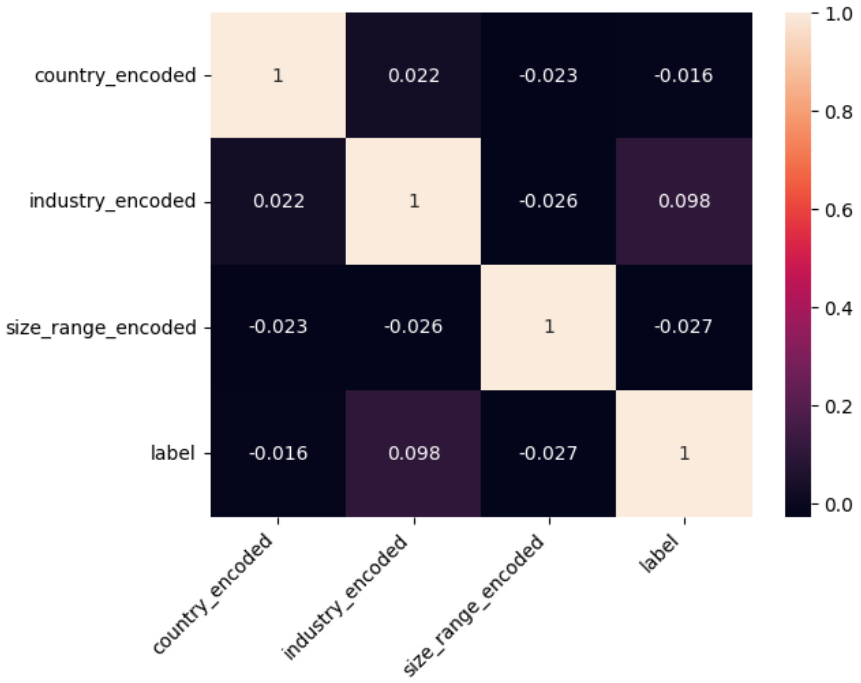
We performed a similar value analysis of the categorical features whose percentage of missing values is again lower than 20%. These features are ***"Country"***, ***"Industry"*** and ***"Size range"***.



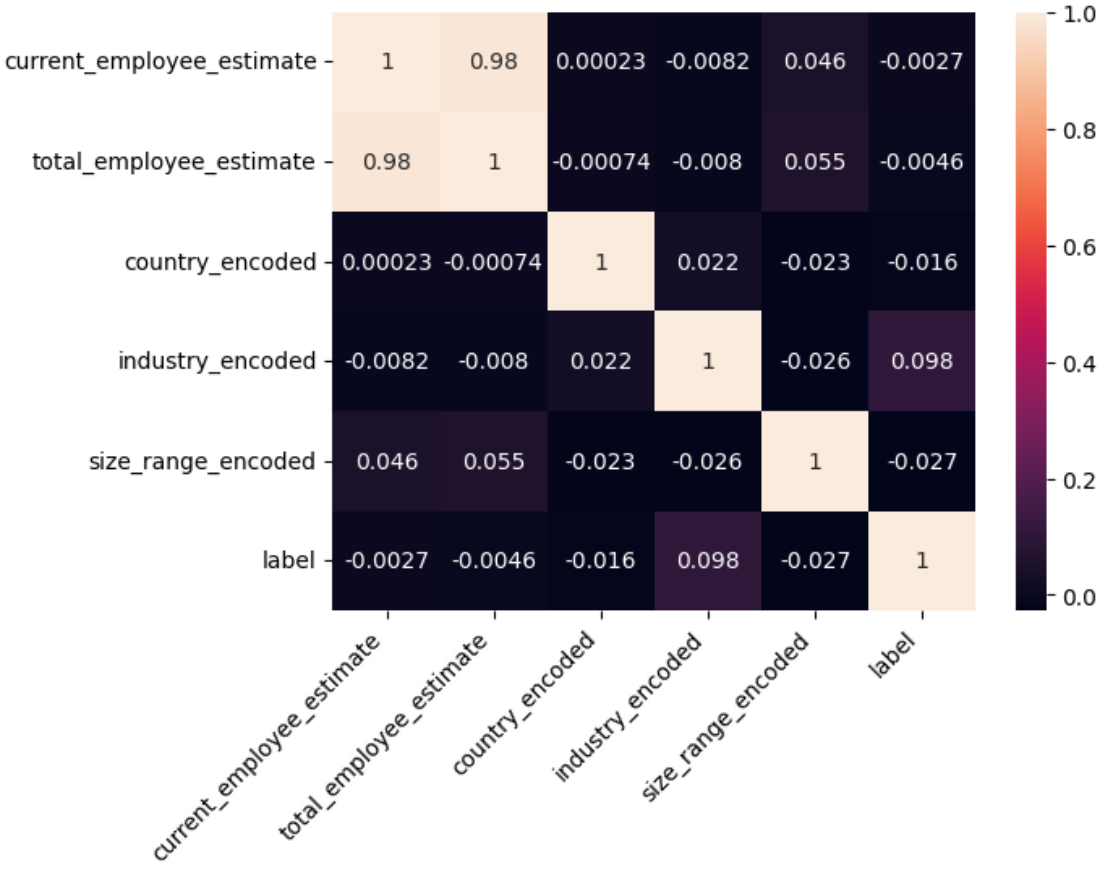




We can see that again, ***"Country"*** and ***"Size range"*** features have a very high concentration on a single value. However, ***"Industry"*** feature is a bit more diverse and may be good to consider for the classification task.



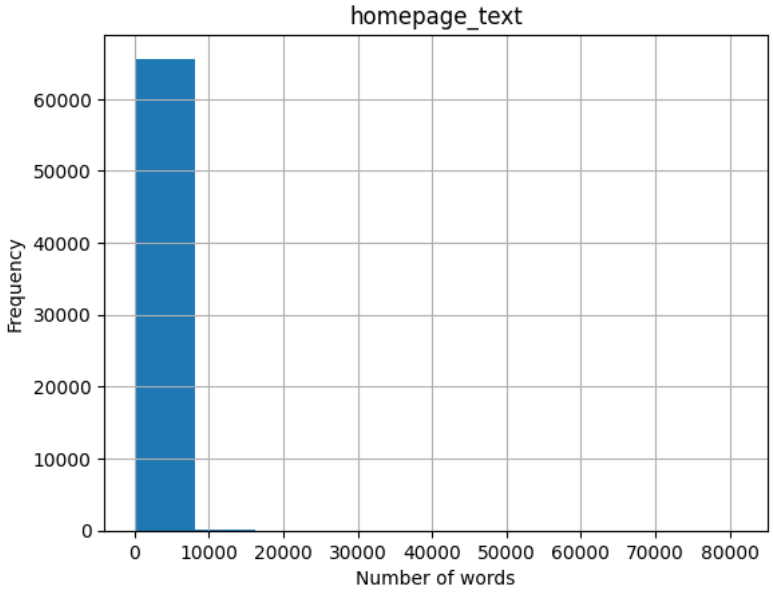
From the correlation matrix, no clear correlation is obvious between the label and the industry. However, since it originally contains text which was encoded into a label. Maybe the text information is useful for the classification task.

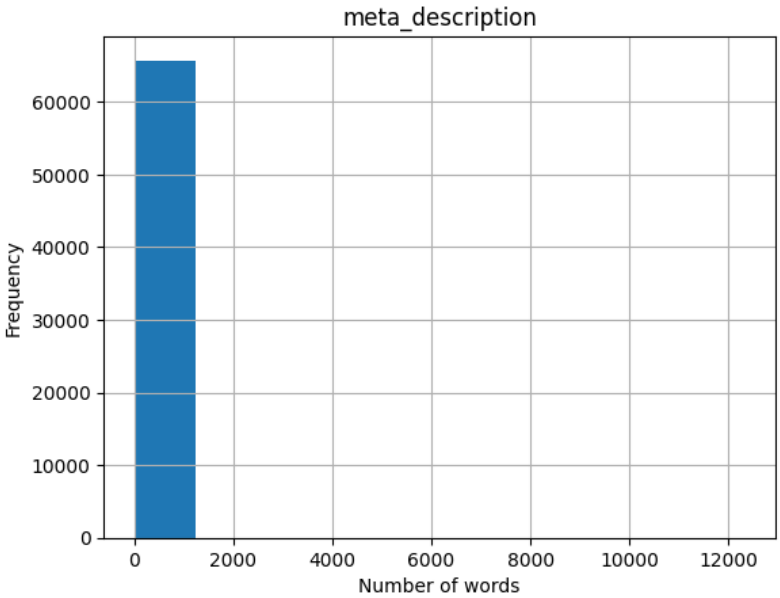


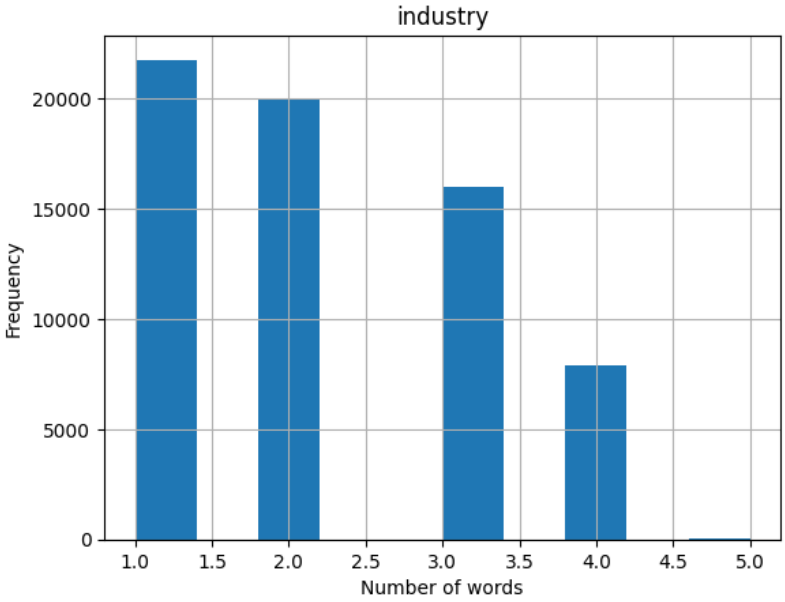
From the correlation matrix above, we can see that no obvious correlation between numerical and categorical data exists.

Text features exploration

We ended our EDA with the text features, in which we also include the ***"Industry"*** categorical feature to further analyze it and check if we could include it in the classification task. Again, we left our features whose percentage of missing values was 20% or more. So, we considered only the ***"Homepage text"*** and ***"Meta description"*** features.







We can see that for any of the three features, the most common word counts does not exceed 9,000 words, even when it comes to the homepage content.

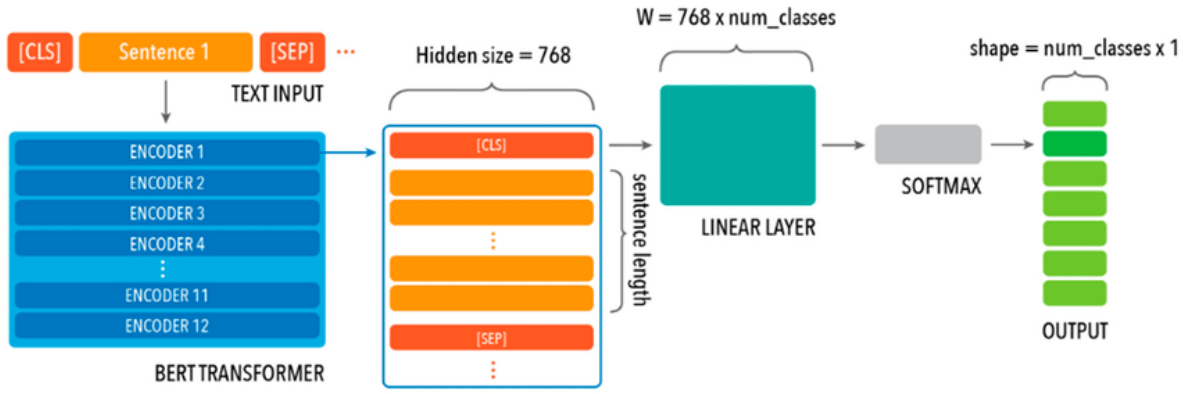
So, we proceeded to the classification task, considering only the ***"Industry"***, ***"Homepage text"*** and ***"Meta description"*** input text features.

Part 4 – Model Development

Since we needed to classify text data, we were in the ML field of "Natural Language Processing". This field has produced some State-of-the-Art Large Language Models (LLMs) which are very capable in comprehending human language and in uncovering hidden patterns, trends, and relationships within textual data that may not be apparent through traditional data analysis.

A family of publicly available LLM models is BERT. BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection. BERT is a Transformer successor which inherits its stacked bidirectional encoders. BERT is a bidirectional model meaning that it uses a bidirectional encoder to represent the input text in a high-dimensional space so that it can better capture word context and connections, due to the fact that the information is passed in both directions, which helps it to better understand the meaning of the text.

BERT is a pre-trained model, trained on massive amounts of text data, such as books, articles, and websites, and can be fine-tuned to match various NLP task. For the task at hand, we chose to use the Uncased BERT Base model. BERT will convert each text input into an embedding vector of 768 length. If the text input is longer than 768 words, then it is truncated. If it is shorter than 768 words, it is padded. This vector will be fed into a pre-trained neural network to obtain the output as one of the 13 classes.



The fine tuned model was designed with a dropout probability of 0.2, and with Early Stopping capability which was monitoring the validation loss. The dataset was split into a training set, which consisted of 85% of the data points, and a validation set which consisted of 15% of the data points. Both training and validation sets are class-balanced, i.e. they contain the same number of samples from each distinct label, to avoid overfitting during training. Before passing the text as input to the model, we performed some preprocessing, which consisted of the following steps:

1. Removing HTML tags (useful especially for the “Homepage text” if any HTML still exists).
2. Removing punctuations and numbers (since we will not perform sentiment analysis).
3. Removing single-character words (they do not offer any useful information).
4. Removing multiple spaces (preserve only a single space between words).
5. Removing redundant whitespaces at the beginning and at the end of the text.
6. When multiple text features are used as input, concatenate them into a single string, using the [SEP] token as a sentence separator between different texts in the final text.

We developed **3 distinct BERT models**:

1. **Model #1**: Using the ***“Industry”*** feature only.
2. **Model #2**: Using both ***“Industry”*** and ***“Meta description”*** features.
3. **Model #3**: Using all ***“Industry”***, ***“Meta description”*** and ***“Homepage text”*** features.

Part 5 – Model Evaluation

All models’ training was interrupted because of the use of Early Stopping Callback, since after some training steps the validation loss started to increase instead of drop. Early Stopping Callback was configured to restore the best model, if triggered. During the training phase, we evaluated our models against the evaluation text, which is class-balanced. After the training phase, we evaluated

our model against all available labeled samples which were not included in the training set, since at this point we do not care about class balancing. The evaluation metrics used are:

- Evaluation Loss
- Accuracy
- F1-score
- Precision
- Recall

Since the evaluation during training was performed on a small subset of data, which is included in the bigger evaluation set, we are only going to present the second (and bigger) evaluation results.

Model ID	Evaluation Loss	Accuracy	F1-score	Precision	Recall
#1	0.0005	0.9999	0.9999	0.9999	0.9999
#2	0.0006	0.9999	0.9999	0.9999	0.9999
#3	0.0012	0.9998	0.9998	0.9998	0.9998

Potential improvements:

1. More data is always welcome.
2. The ***"Homepage text"*** feature contains unnecessary information which was maintained after parsing the HTML code and which does not help with the classification task (or any other similar task) since it contains words and phrases like "Sign up", "Log in", "Cookies", "Javascript", "Privacy policy", etc..
3. Enrich the dataset with more adversarial data points.
4. Use **Large BERT** variation (or some other LLM) model to overcome the small input vector size limit.
5. Come up with additional and useful features for training.