# A gender prediction study from census manuscript data

**Antonio Sasaki**
antonio.sasaki@ensae.fr

## 1 Introduction

The Socface project brings together researchers to analyze French census records from 1836 to 1936. They are using advanced technology to extract information from handwritten lists, but in certain censuses gender information may be missing. Thus, as this is essential for other analyses, our objective in this work is to use machine learning knowledge to develop a **gender prediction model** based on available data.

### 1.a Briefly explaining our main data

In our study, we used the databases `firstname_with_sex.csv` and `transcriptions_with_sex.csv`, which respectively represent a file of first names with their frequencies in each gender and a file containing manual and automatic transcriptions of each individual, along with their associated gender. As we will see, with the first file, we can easily estimate the probability of each individual's gender given their name.

It is simple to know that `prob = pd.read_csv("firstname_with_sex.csv")` is much easier to deal with than the other file because it is numerical data. As we had said before, it just tells us the number of times, `'male'('first_name')` and `'female'('first_name')`[1], that a `'first_name'` appears naming a man and a woman respectively. Therefore, it is very easy to know the chance of a specific name being in one group or another, just calculate `'prob_male'` and `'prob_female'` using this matrix transformation below

$$\begin{bmatrix} \texttt{'prob\_male'}(\texttt{'first\_name'}) \\ \\ \texttt{'prob\_female'}(\texttt{'first\_name'}) \end{bmatrix} = \begin{bmatrix} \dfrac{\texttt{'male'}(\texttt{'first\_name'})}{\texttt{'male'}(\texttt{'first\_name'}) + \texttt{'female'}(\texttt{'first\_name'})} \\ \\ \dfrac{\texttt{'female'}(\texttt{'first\_name'})}{\texttt{'male'}(\texttt{'first\_name'}) + \texttt{'female'}(\texttt{'first\_name'})} \end{bmatrix} \in \mathbb{R}^2$$

Now just add one of these percentages, male or female, to our main data frame `df` when there is an intersection between the names present in each file. It is worth specifying that initially we defined our data frame as `df = pd.read_csv("transcriptions_with_sex.csv")`. Before combining these odds with this data, we cleaned the `df` and tested it on machine learning models to understand the impact of probabilities on the prediction, you can check this in the code published on GitHub. As expected, among all the features we used, the ones that had the most impact were always those linked to the frequency of each gender.

## 2 Starting the data preprocessing

To begin, we checked that there are no duplicate individuals in `df`, using an identification code in the first column `abs(len(df["subject_line"].unique())-len(df)) = 0`. Without making any changes yet, it is possible to verify that our amount of data in this file is already relatively small, as `len(df)` $= 241$. Although many details are missing for some individuals in the manual transcription column, it typically includes the person's first name and last name, occupation, age or date of birth, city of residence, and position within the family. To discard possible information that does not seem to correlate with gender in the creation of our variables, we will just look at `'first_name'`, `'occupation'` and `'position'`.

### 2.a Cleaning data from our manual transcriptions

Our main objective now is to divide our text into keywords. It was possible for us to tokenize the above data using a selection structure together with the separator `split(' ')` and then deleting the empty rows in these columns from the entire dataframe `df`. After this transformation we obviously changed the size `len(df)` $= 189 < 241$ and now our amount of data, which was already not large, is even smaller. We were also interested in some items that also appear in the automatic transcriptions, such as profession, but the number of rows missing data `abs(len(df[df.profession.notnull()])-len(df))` $= 164$ makes an efficient analysis unfeasible. This is a common trade-off to be resolved in the preprocessing step.

Continuing our cleaning, after dividing the important words into columns, we used the `lower()` function on our strings to standardize them as they are in the other gender frequency file. Furthermore, it was also necessary to create a code to identify the lines written `'idem'` and replace them with the information directly above present in the same column. To finish, I manually changed some words that appeared recurrently in these new columns and which were actually abbreviations of other words that already existed there. In the tables below you can see all the changes we made to help with our analysis.

| Column name | Old name | New name |
|---|---|---|
| 'occupation' | 's.p' | 'sans' |
| 'occupation' | 'sp' | 'sans' |
| 'occupation' | 's.p.' | 'sans' |
| 'occupation' | 's' | 'sans' |
| 'occupation' | 'néant' | 'sans' |

**Table 1:** Update of some column names **1**

| Column name | Old name | New name |
|---|---|---|
| 'occupation' | 'cultiv' | 'cultivat' |
| 'occupation' | 'cult' | 'cultivat' |
| 'position' | 'ch.' | 'chef' |
| 'position' | 'domest.' | 'domestique' |
| 'position' | 'ép' | 'épouse' |

**Table 2:** Update of some column names **2**

Notice that we rewrited `'cultivat'` and `'cult'` to `'cultivat'`, which is another abbreviation for the profession of cultivator. However, in this specific case, I believed it would be more prudent to unify the abbreviations instead of writing the full name of the occupation. This is because, in French, words referring to profession vary grammatically depending on whether the worker is a man or a woman. For example, sometimes in the data this occupation is written as `'cultivatrice'` and other times as `'cultivateur'`.

## 3 Vectorizing our tokens using Word2vec

As we already know, the Word2vec method is a technique in natural language processing for obtaining vector representations of words. After the last part, we are now ready to accurately calculate the number of tokens to be vectorized by this algorithm. Now we can create a list with the updated data coming from the representative column values `df['first_name'].values`, `df['occupation'].values` and `df['position'].values`. With these lists called `first_name`, `occupation` and `position` we can see that `len(first_name)` $=$ `len(occupation)` $=$ `len(position)` $= 189 < 241$. In other words, since the beginning of the data manipulation process we have already lost 52 rows from our original data frame.

### 3.a Methods for the projection of high-dimensional points

When we trained our algorithm it transformed each token into a point within the real coordinate space of dimension 5, particularly within the space $[0,1]^5 \subseteq \mathbb{R}^5$. To be able to visualize the distribution of our words in space, we can use statistical tools like t-SNE and PCA to project them into some space of smaller dimension, such as $\mathbb{R}^2$ or $\mathbb{R}^3$. With lower dimensions, we can improve the interpretability of our result because distances and clusters in reduced-dimensional space can often reflect relationships between words.

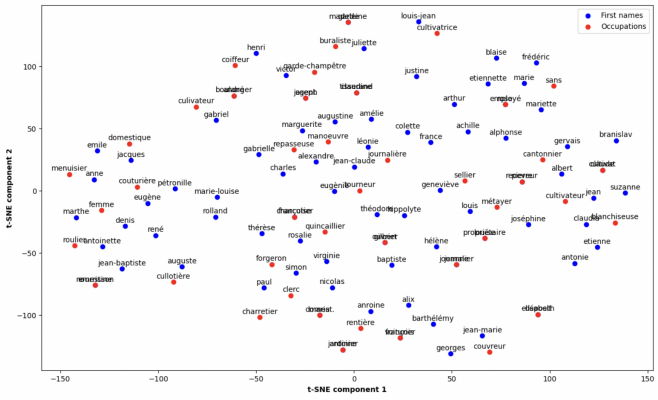


**Figure 1:** t-SNE for `'first_name'` and `'occupation'`

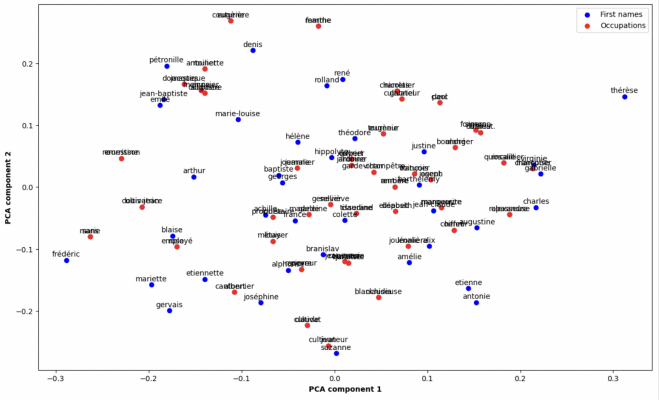


**Figure 2:** PCA for `'first_name'` and `'occupation'`

In the tables above, we projected our vectors using t-distributed stochastic neighbor embedding and principal component analysis respectively. In order to make the visualization more understandable, we only projected the vectors of the variables related to the `'first_name'` and `'occupation'`. The initial impression we get from looking at these two figures is that the data projected by t-SNE is more evenly distributed across the plane, while most of the PCA data is concentrated. However, no strong pattern or clustering correlating with semantic differences between the words could be distinctly identified.

This difference in the distribution of projected data after applying PCA and t-SNE can be explained by the characteristics and functioning of these methods. Since PCA focuses on the variance of the data, it tends to preserve the global structure of the data, leading to a more compact representation where data may appear more concentrated along the principal directions. On the other hand, t-SNE prioritizes the preservation of local structure, creating representations where similar points are closer than dissimilar ones, which can result in a more dispersed distribution of our transformed vectors as seen above.

# 4 Building our gender classifier

First I need to remember that we used the formula from the beginning to create a new column in the data frame `df` called `'prob_male'` with the probability that given the first name in each line, the person with that name identifies himself as male. To be mathematically precise, we are actually working with an estimator of this conditional probability built from data in the `prob` file. In any case, it is worth remembering that there are names that are in the main data frame but not in the list of estimated probabilities, so we must exclude these lines from the `df`. After deleting these lines we have `len(df['prob_male'].values)` $= 182$.

Thus, we have to construct a feature space of dimension 4 with the variables: `first_name`, `occupation`, `position` and `prob_male`. We can vectorize the set of true genders manually by assigning the integers $\{0, 1, 2\} \in \mathbb{Z}$ to the cases in which the individual is male, female and non-binary respectively. Now it is possible to notice that the low amount of data, mixed with the small proportion of non-binary people, makes this factor a key point in our performance, as it is very difficult to predict especially this gender.

## 4.a Choosing a suitable algorithm for our model

Several different machine learning models are possible for this kind of problem, such as k-NN, SVC, logistic regression and random forest. The k-nearest neighbors algorithm, although known for its simplicity and ease of implementation, is less effective at handling datasets with many features or imbalanced classes, as is the case with our context. On the other hand, the support vector classifier, while efficient in high-dimensional spaces, may also not be suitable for data with these disproportions. Logistic regressions are easily interpretable but they are limited in their ability to capture nonlinear relationships between features and gender. Therefore, we chose the random decision forest algorithm to help us build our classifier.

As mentioned, random forest appears to be robust regarding class disparity compared to other models, reducing overfitting and being capable to work with complex data. Its ability to combine multiple decision trees and utilize the average of their predictions contributes to a more stable and accurate performance, especially in datasets like ours, where balancing between different gender types can be challenging.

### 4.b Testing the model with hyper-parameters tuned

Before training the dataset, we need to calibrate a set of hyper-parameters to maximize the classification capacity of the random forest algorithm. Thus, we designed a hyper-parameter tuning program, using `GridSearchCV` from `sklearn.model_selection`, which contains crucial hyper-parameters such as the number of trees in the forest, the maximum depth of each tree, the minimum number of samples required to split an internal node and the minimum number of samples required at each leaf node. These variables mentioned are respectively called `n_estimators`, `max_depth`, `min_samples_split`, and `min_samples_leaf`.

| Hyper-parameter | Range tested | Result |
|---|---|---|
| `n_estimators` | [50, 100, 200] | 100 |
| `max_depth` | [**None**, 10, 20] | **None** |
| `min_samples_split` | [2, 5, 10] | 10 |
| `min_samples_leaf` | [1, 2, 4] | 2 |

**Table 3:** Hyper-parameters optimal results for our model

| Measure | Training | Testing |
|---|---|---|
| Recall | 0.99082 | 0.94520 |
| F1-score | 0.99129 | 0.92780 |
| Accuracy | 0.99082 | 0.94520 |
| ROC-AUC | 0.99922 | 0.89785 |

**Table 4:** Results of training and testing

The performance metrics revealed in the table provide insights into how well the model is performing on both the training and testing data. The recall highlights the proportion of positive instances that the model correctly identifies relative to the total actual positive instances. While the recall is high in both datasets, there is a slight decrease in the testing data, implying that the model may be failing to identify some positive cases with the same accuracy. The difference between F1-score training and testing values suggests a potential difficulty in generalizing the model to new data. The accuracy, indicating the proportion of correct predictions, is consistently high in both datasets, indicating good predictive ability.

On the other hand, the receiver operating characteristic curve, a measure of the model's discriminative ability, shows a significant discrepancy between the training and testing data, certainly implying a possible overfitting of the classifier to the training data. This could be happening due to that problem I mentioned about the discrepancy in balance between the different classes in our model. To better understand these metrics, let us visualize the ROC curve for each class and use the mean decrease in impurity algorithm to understand the impact that each feature has on the model's prediction. It is worth remembering that we vectorized the elements of three columns into 5-dimensional points and then we added `'prob_male'`.

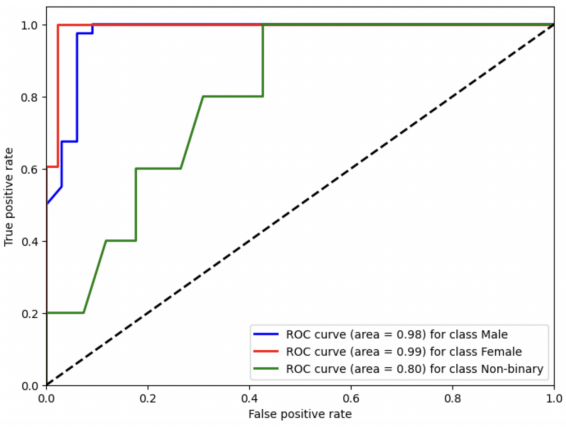


**Figure 3:** The ROC curves for each class

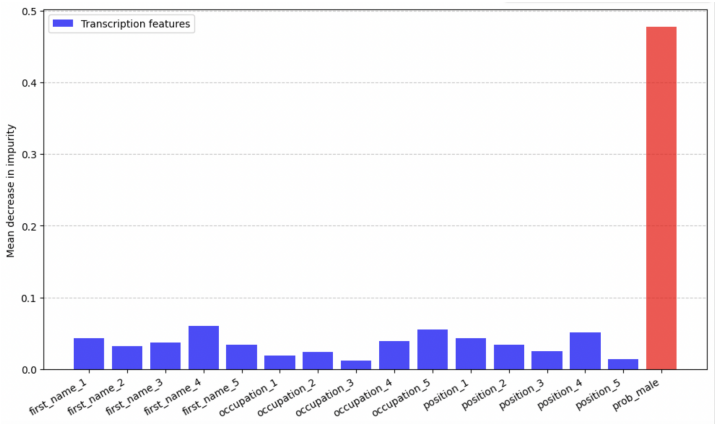


**Figure 4:** The mean decrease in impurity algorithm

### 4.c Some problems that could be improved

The classification issues of non-binary genders was already expected and these errors are easily verified using a confusion matrix or plotting the ROC curves as we did. Among the algorithms I mentioned earlier, the SVC from `scikit-learn`, you can specify the `class_weight` parameter to assign different weights to classes during model training. This allows for giving more importance to less represented classes, helping to address significant class imbalances and resulting in better classifier performance.

As you can also see, another difficulty is the very high dependence that our algorithm has on our probability estimator conditioned on the name. This behavior can be problematic due to concerns about overfitting. When this occurs, there is a tendency for the model to overly adjust to specific training data, rather than capturing general patterns that may extend to new data. This can result in a model that does not generalize well to unseen data, leading to inaccurate predictions. Furthermore, the dominance of a single variable can compromise the interpretability, as the model does not provide insights into the relevant characteristics of other data that drive predictions. This makes it difficult to understand how the algorithm makes its decisions and may limit its usefulness in practical scenarios.

Lastly, remember that we are working with a relatively small amount of data. In these dataset types, allocating a larger proportion of data to the testing set may be justified if there is a concern about the accuracy of model evaluation. However, this also means that we will have fewer data available for training the model, which may result in inferior performance, especially if the dataset is already limited in size. We particularly fixed 40% of the data for testing and 60% for training, which could perhaps be improved.

## 5 Conclusions and recommendations

As was shown, we managed to achieve approximately **94.52%** score for testing data and **99,08%** score for training data. In summary, while the model demonstrates satisfactory ability to correctly predict positive and negative instances, it is crucial to monitor differences between performance metrics to assess generalization ability and avoid overfitting. Overall, this is a good enough result for a 3-gender context. Moreover, to resolve the difficulties caused by the small size of our main dataset `df`, it would be pertinent in the future to look for new files with more data so that we can work with more information.

Concerning overfitting, it is important to further investigate the cause of our feature dominance and seek ways to mitigate these issues to ensure a more robust and interpretable model. To conclude, a good idea to improve the results might be to forego the random forest and search for algorithms where I can manually adjust the weight between different classes, to help our underrepresented non-binary gender.

### 5.a Appendix: the meaning of `'ambigu'` in the data frame

It is also possible to understand the meaning of `'ambigu'`, that appears in the gender column, not signifying non-binary gender but rather imprecision from the census collector in defining the individual's gender. In this case, we could discard this information and would have a prediction model with only two genders, which would be much more straightforward. It is highly likely that this type of model would be entirely based on the probability estimator, and its scoring and accuracy measures would easily reach 100%. For these reasons, it is more challenging to interpret these data as motivating the existence of a third gender that can be understood and identified by our classifier, which was the scenario we assumed.

## References

[1] Boillet, M., Tarride, S., Schneider, Y., Abadie, B., Kesztenbaum, L., & Kermorvant, C. (2024). *The Socface Project: Large-Scale Collection, Processing, and Analysis of a Century of French Censuses.* `arXiv:2404.18706 [cs.CV]`. Retrieved from `https://doi.org/10.48550/arXiv.2404.18706`.