Celebrity Profiling using Twitter Follower Feeds
Notebook for PAN at CLEF 2020

Abigail Hodge and Samantha Price

Northeastern University
hodge.ab@northeastern.edu, price.sam@northeastern.edu

Abstract This paper describes our approach to completing the Celebrity Profiling shared task set forth by PAN at CLEF 2020. We discuss the features selected (including part-of-speech tags, named entity types, and word vectors) as well as the logistic regression, random forest and support-vector models we tested for this task. The resulting confusion matrices and evaluation scores are provided.

1 Introduction

In this paper, we attempted to solve a natural language processing problem set forth by PAN as part of the organization's 2020 competition. At a basic level, the problem required a celebrity profiling ML model that could estimate the age, occupation, and gender of a given celebrity based upon the tweets of their Twitter followers [20]. More specifically, the test input was a list of JSON objects representing the tweets of followers, and the output was a list of JSON objects where each object contained the ID of a given celebrity, their predicted occupation (among a possible list of 'sports', 'performer', 'creator', and 'politics'), their predicted birth year (between 1940 to 1999), and their predicted gender ('male' or 'female'). The most unique aspect of this problem was that no information about a celebrity from their Twitter account or other sources was provided as input for the test dataset; the celebrities had to be profiled solely on the basis of their followers' tweets.

First, we discuss some related work done in this field, namely, PAN's 2019 celebrity profiling task, which required competitors to profile celebrities based on their own tweets. Next, we discuss our methodology for feature extraction and model building. Finally, we explain the results of our models for age, gender, and occupation. An earlier version of this paper was submitted for an academic project at Northeastern University [7].

2 Related Work

This task is, on the surface at least, very similar to the 2019 PAN Celebrity Classification task [19]. This task required competitors to classify celebrities by their birth year (between 1940 and 2012), gender (male, female, or non-binary), fame (rising, star, or superstar) and occupation (sports, performer, creator, politics, manager, science, professional, religious) based on their tweets. This task differed from the 2019 task in a few key ways. First, the fame classification task was not present. Second, the remaining three tasks had fewer categories. Birth year was restricted from 1940-1999, the non-binary class was no longer present for the gender category, and the manager, science, professional, and religious classes were no longer present for the occupation category. However, we also had significantly fewer data points to work with. The 2019 task had 48,335 celebrities to use for training. The 2020 task had only 1,920. Furthermore, for the 2020 task, we used data from the celebrity's followers, not the celebrities themselves.

For our feature extraction, we built off of work done by Argamon et al. [1]. They examined which features were generally most useful for anonymous authorship profiling, and had a good deal of success with POS tags. We also built off of our own success with word embeddings [6] for an age profiling task. This will be discussed in greater depth in the next section.

Generally, most of the submissions for the 2019 task seemed to find success using classical natural language processing and machine learning techniques. In fact, the three competitors in 2019 who attempted to use deep learning techniques reported that these techniques were not suited for the task. [19] Therefore, we chose to focus on models that do seem well suited: SVM, Logistic Regression, and Random Forest (discussed further in the Algorithms section). Our decisions about what algorithms to select were also influenced by our work on author and time period classification in another course [6].

3 Approach Description

3.1 Feature Extraction

For feature extraction, we decided to utilize features that have proven useful for author profiling problems in the past. Because we were not trying to profile the authors of the tweets, but rather a common person that all of these authors followed, we were required to make certain assumptions about the follower/followee relationship. We assumed that the followers of a celebrity might have similar interests to that celebrity, that is, a follower of a politician might post a lot about politics, a follower of a performer might post a lot about music/concerts, etc. We also assumed that celebrities might attract followers that are largely of a similar age and gender. Essentially, we decided to treat the aggregate group of tweets as though they were authored by the person we were trying to profile.

There are a few simple features that have proven highly effective for author profiling tasks, namely, stop words and part-of-speech (POS) tags. For example, Argamon et al. [1] found that men tend to use more determiners and prepositions, while women tend to use more pronouns, to the degree that these features are given significant weight in a machine learning model. Another effective feature appears to be n-grams [16] but we decided not to utilize n-grams as our dataset has a large vocabulary, and this would therefore require a lot of features to represent.

We had relative success with word embeddings with an author profiling task involving age, albeit on books rather than tweets [6]. Therefore, we decided to utilize them again for this project, averaging together the word embeddings for all words (in vocabulary) for a given celebrity's tweets. Finally, looking at last year's celebrity profiling task results, it appeared that occupation was generally the lowest scoring classifier [19] so we decided to add in features specifically to improve upon this classifier: named entity types. This was based on the logic that, for example, politicians would be more likely to talk about countries or organizations, creators would be more likely to talk about art, etc. However, it should be noted that adding named entity recognition to our pipeline significantly slowed down our feature extraction code—it took several minutes to run a single celebrity. However, we decided that the boost to classification was worth the extra runtime.

Ultimately, we decided on the following features: POS tags, stop-word count, named entity types, average word vectors, tweet length (in characters), number of links, number of hashtags, number of mentions, and number of emoji [10]. These were all normalized by total number of words in a celebrity's tweets (with the exception of word vectors and average tweet length, since those were already averages). Feature extraction of POS tags, stop words, NER types, and word vectors were done using the sPacy library [8]. Additional logic was executed using Numpy [12][17].

3.2 Algorithms

After extracting features, we decided upon three different machine learning algorithms to train on these features and compare the resulting metrics: logistic regression, random forest, and support-vector machine (SVM). The models were constructed through Sci-Kit Learn [13][3]. Each algorithm was implemented through three different models: one for occupation, one for gender, and one for birth year. Unique hyperparameters were defined for each model, and the optimized parameters were selected through 5-fold cross validation with scoring based on the macro f1 score (functionality also provided through Sci-Kit Learn [13][3]). Hyperparameter tuning resulted in higher f1 scores for all three chosen algorithms.

For the logistic regression model, the parameters chosen for tuning were the type of regularization (L1 or L2), the penalty on regularization (0.01, 0.1, or 1) and the type of solver (liblinear or saga). An article written by Qiao (2019) [15] inspired the choice of hyperparameters for tuning. The optimized parameters for occupation were (L2, 0.1, saga), the parameters for gender were (L2, 0.1, liblinear), and the parameters for birth year were (L1, 1, saga).

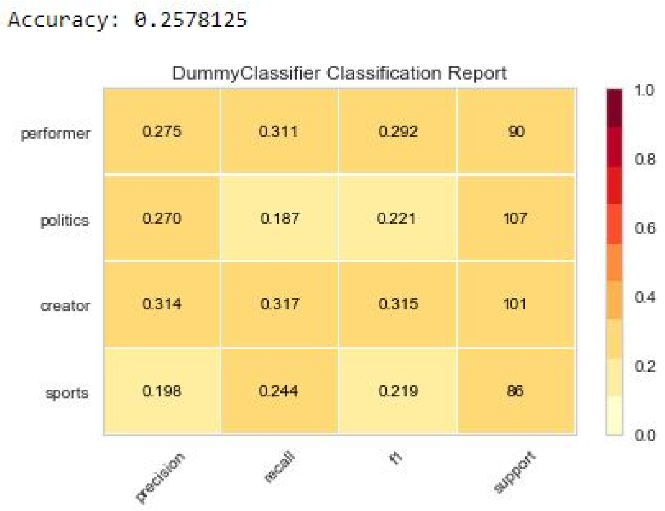
The selection of random forest and support-vector machine was inspired by the PAN 2019 celebrity profiling task, as these two models were proven to be successful [19]. The possible parameters for the random forest classifiers (based on Koehrsen, 2018 [11]) were number of estimators (50, 100, 500), max depth (None, 5, 10), and max features (auto or log2). Ultimately, the chosen parameters for training the random forest classifiers were (500, None, auto) for occupation, (500, None, auto) for gender, and (50, log2, 5) for birth year.

Finally, regularization penalty, (0.01, 0.1, 1) kernel type (linear, poly, rbf), and gamma value (0.1, 1, 10) (Fraj, 2018 [4]) were chosen as the adjustable hyperparameters for the support-vector machine model. The best parameters were determined to be (0.1, 0.1, linear) for occupation, (0.1, 0.1, linear) for gender, and (0.01, 0.1, linear) for birth year.

After running cross-validation and training all classifiers with the extracted features and optimized hyperparameters, metrics were determined for the different classifiers based on a section of the training data (20%) set aside for testing. Additionally, an alternative f1-score (besides the one from Sci-Kit Learn [3]) for birth year was calculated, as PAN [20] dictated in the task description that any predicted year within a specific range of years would be considered correct (true birthyear -m < predicted birthyear < true birthyear + m); this alternative f1-score took this window of error into account. Finally, PAN [20] also mentioned that submissions to the competition would be judged based upon a special "cRank" metric that combines the f1-scores for occupation, gender, and birth year. Thus, the cRanks for logistic regression, random forest, and SVM were also calculated in this project (results below).

4 Results

We defined baseline models using Sci-Kit Learn's DummyClassifier class to compare to our trained models [3]. Figure 1 displays the results from the occupation classification model, Figure 2 shows gender classification, and Figure 3 shows birth year classification. The classification report is a heat map that demonstrates the precision, recall, and, f1 score for every possible class; lighter colors indicate lower scores and darker colors indicate higher scores. Due to the large range of birth years as possible classes, a visualization of the metrics could not be produced, but a textual description is provided. All of the subsequent visualizations were constructed through Yellowbrick [2] and Matplotlib [9].

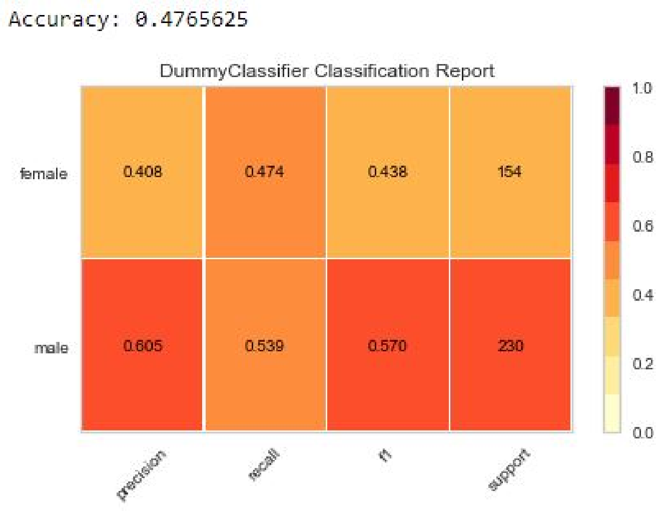


```
Accuracy: 0.2578125
```

DummyClassifier Classification Report

	precision	recall	f1	support
performer	0.275	0.311	0.292	90
politics	0.270	0.187	0.221	107
creator	0.314	0.317	0.315	101
sports	0.198	0.244	0.219	86

Figure 1: Metrics for Baseline Occupation Classifier

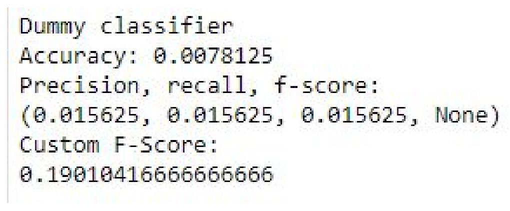


```
Accuracy: 0.4765625
```

DummyClassifier Classification Report

	precision	recall	f1	support
female	0.408	0.474	0.438	154
male	0.605	0.539	0.570	230

Figure 2: Metrics for Baseline Gender Classifier

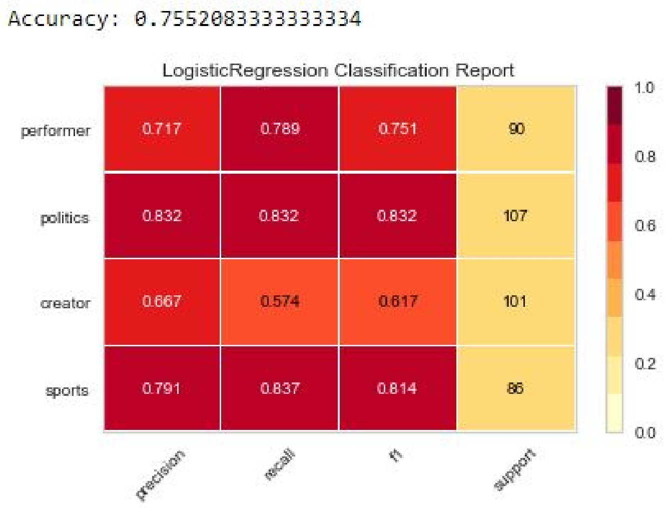


```
Dummy classifier
Accuracy: 0.0078125
Precision, recall, f-score:
(0.015625, 0.015625, 0.015625, None)
Custom F-Score:
0.19010416666666666
```

Figure 3: Metrics for Baseline Birth Year Classifier

4.1 Logistic Regression

This section contains classification reports and confusion matrices for the occupation and gender logistic regression classifiers (Figures 4-7), as well as a textual description of metrics for the birth year classifier. The confusion matrices are also heat maps, where a darker square indicates more predictions and a lighter square indicates fewer predictions. These classifiers were trained with the hyperparameters mentioned in Section 3.2. As can be seen in the figures, the logistic regression occupation classifier significantly outperformed the baseline occupation classifier. Its highest f1-score was 0.832 for the politics class.



```
Accuracy: 0.7552083333333334
```

LogisticRegression Classification Report

	precision	recall	f1	support
performer	0.717	0.789	0.751	90
politics	0.832	0.832	0.832	107
creator	0.667	0.574	0.617	101
sports	0.791	0.837	0.814	86

Figure 4: Metrics for Logistic Regression Occupation Classifier

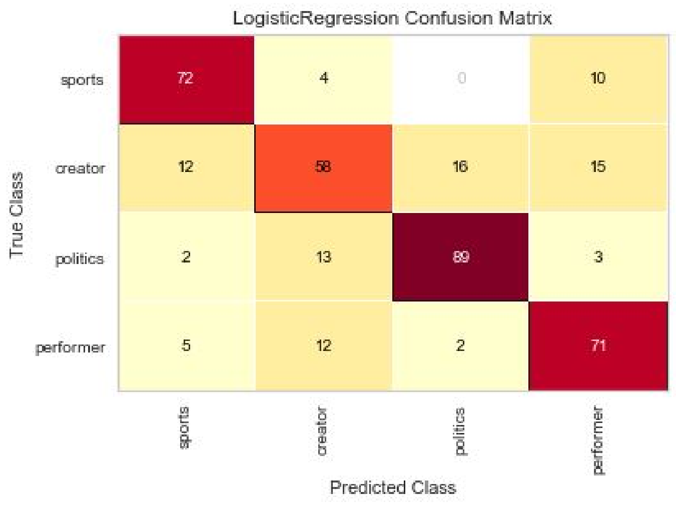


Figure 5: Confusion Matrix for Logistic Regression Occupation Classifier

The gender classifier was also more successful than the baseline, with a maximum f1-score of 0.792 (for the male label).

```
Accuracy: 0.7473958333333334
```

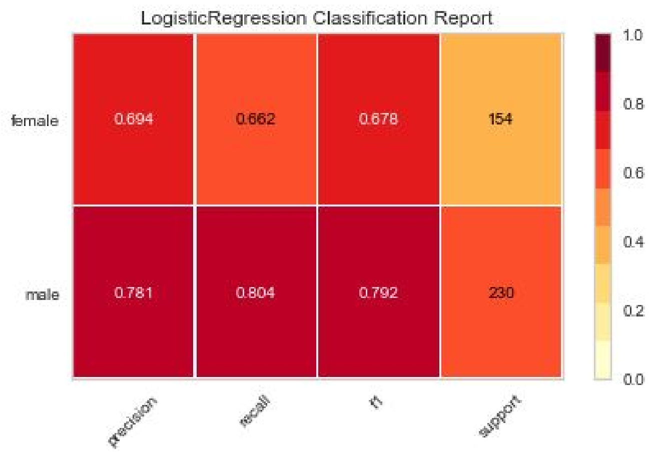


Figure 6: Metrics for Logistic Regression Gender Classifier

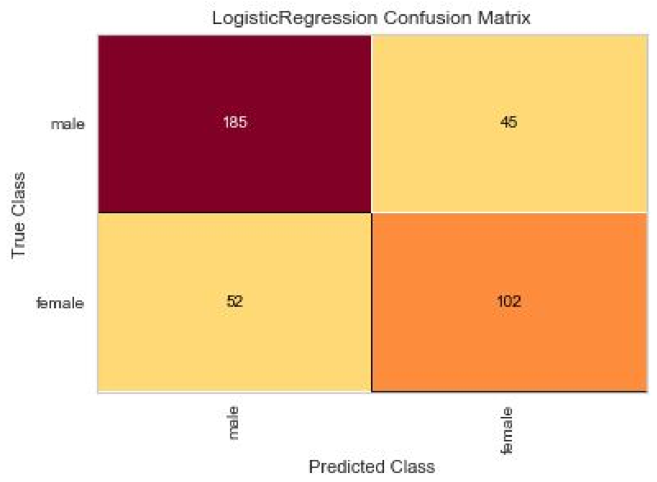
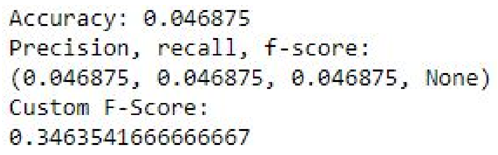


Figure 7: Confusion Matrix for Logistic Regression Gender Classifier

Finally, the birth year classifier showed some improvement compared to the baseline classifier, although the number of possible classes and unbalanced training set made this classifier difficult to train. Its custom f1-score (taking the aforementioned window of error into account) was 0.346.

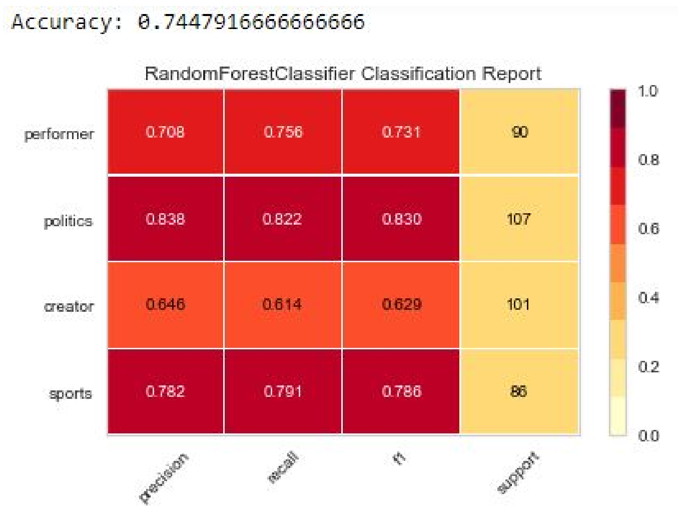


```
Accuracy: 0.046875
Precision, recall, f-score:
(0.046875, 0.046875, 0.046875, None)
Custom F-Score:
0.3463541666666667
```

Figure 8: Metrics for Logistic Regression Birth Year Classifier

4.2 Random Forest

Figures 9-13 display the metric information derived from the occupation, gender, and birth year random forest classifiers. In this case, the metrics are clearly superior to those from the baseline model. In comparison to the logistic regression occupation model, the random forest occupation model had a slightly worse accuracy, but higher maximum f1-score (0.830 for politics). The random forest gender classifier had a worse accuracy (0.70) and maximum f1-score (male label) than the logistic regression algorithm. The custom f1-score for birth year was basically the same for both algorithms, with a slight edge given to logistic regression.



```
Accuracy: 0.7447916666666666
```

Figure 9: Metrics for Random Forest Occupation Classifier

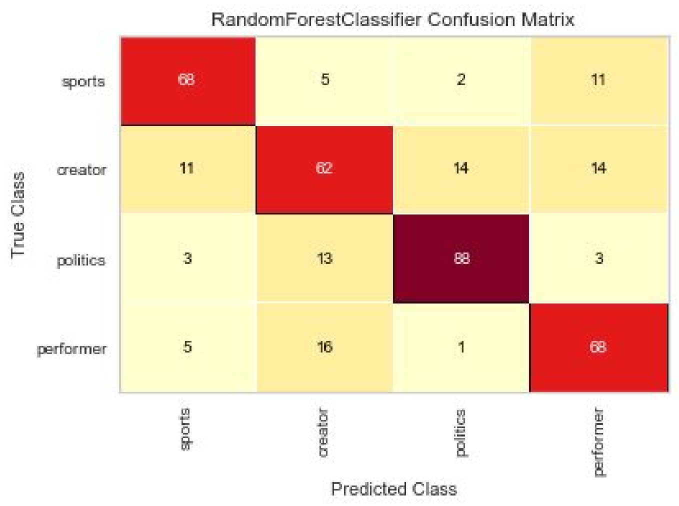


Figure 10: Confusion Matrix for Random Forest Occupation Classifier

```
Accuracy: 0.703125
```

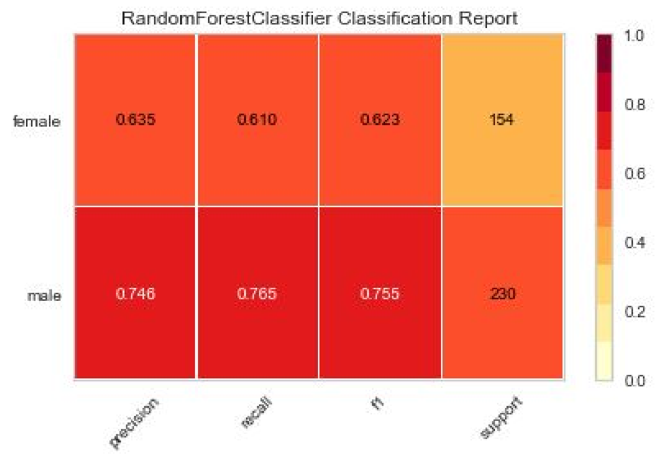


Figure 11: Metrics for Random Forest Gender Classifier

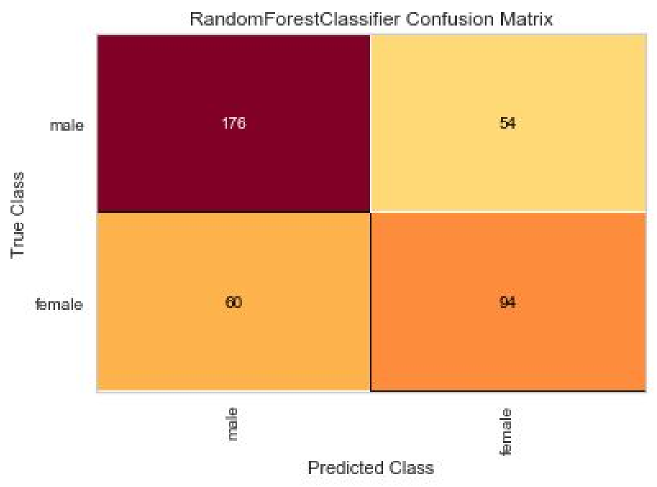
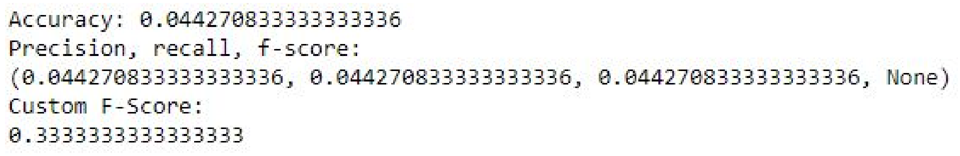


Figure 12: Confusion Matrix for Random Forest Gender Classifier



```
Accuracy: 0.044270833333333336
Precision, recall, f-score:
(0.044270833333333336, 0.044270833333333336, 0.044270833333333336, None)
Custom F-Score:
0.3333333333333333
```

Figure 13: Metrics for Random Forest Birth Year Classifier

4.3 Support-Vector Classifier

A support-vector algorithm (SVC) was the last one to be experimented with. Figures 17-21 display the results of this experimentation. Similar to the random forest classifier, no major improvements in metrics could be seen with the SVC, although it was still more successful than the baseline classifiers. The occupation classifier achieved a comparable accuracy to the random forest classifier, and its highest f1-score (0.798) was the lowest out of the three algorithms. The algorithm's performance with gender classification and birth year classification was very close to those of the random forest and logistic regression algorithms.

```
Accuracy: 0.7369791666666666
```

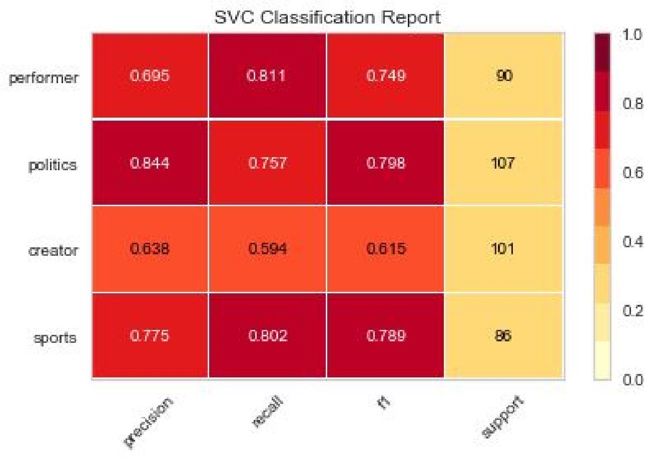
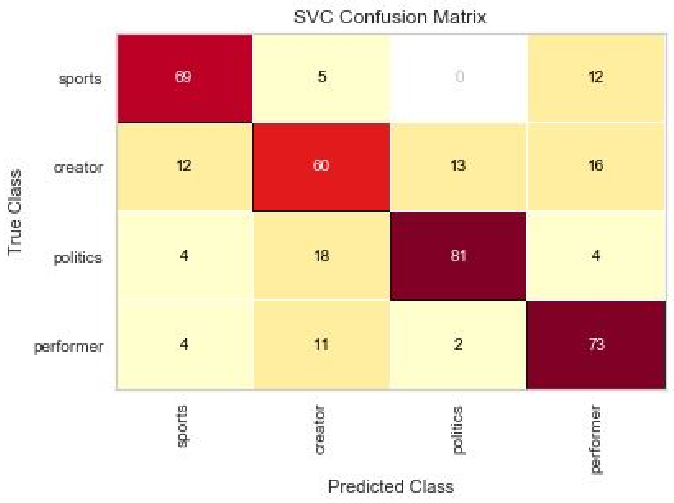
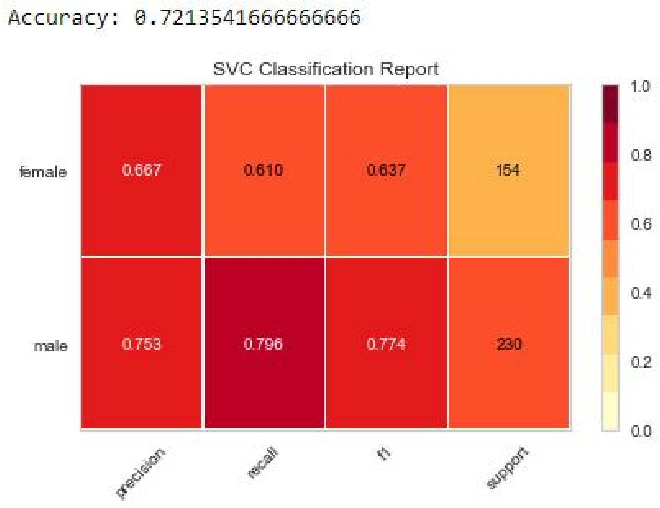


Figure 14: Metrics for SV Occupation Classifier



True Class \ Predicted Class	sports	creator	politics	performer
sports	69	5	0	12
creator	12	60	13	16
politics	4	18	81	4
performer	4	11	2	73

Figure 15: Confusion Matrix for SV Occupation Classifier



```
Accuracy: 0.7213541666666666
```

	precision	recall	f1	support
female	0.667	0.610	0.637	154
male	0.753	0.796	0.774	230

Figure 16: Metrics for SV Gender Classifier

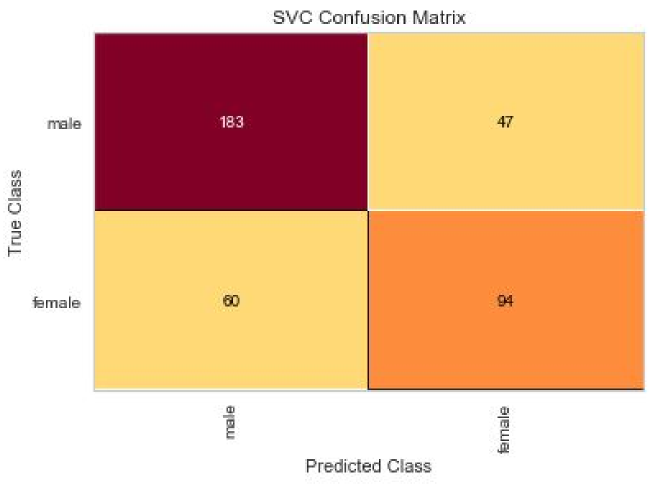
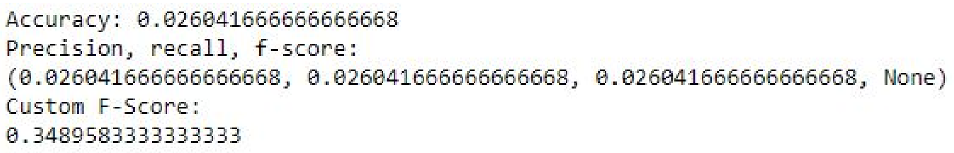


Figure 17: Confusion Matrix for SV Gender Classifier



```
Accuracy: 0.026041666666666668
Precision, recall, f-score:
(0.026041666666666668, 0.026041666666666668, 0.026041666666666668, None)
Custom F-Score:
0.3489583333333333
```

Figure 18: Metrics for SV Birth Year Classifier

4.4 Submission

Based on the results described above, it was difficult to choose which combination of models was most effective, as the scores for each category were similar. Table 1 displays the (rounded) f1-score that each classifier produced for each category. Ultimately, the final metric of evaluation was cRank [20]. The cRank value for logistic regression was 0.541, the value for random forest was 0.522, and the value for SVM was 0.535. This result, paired with the relative consistency of the metrics produced by the logistic regression models, indicated that a combination of three logistic regression models was the preferable algorithm to classify occupation, gender, and birth year. Thus, the software submitted to TIRA [14] contained three logistic regression models to predict labels (utilizing additional Python modules [18][5]) from the tweets of celebrity followers. The final results were: a c-rank of 0.577, a birth year f1-score of 0.432, a gender f1-score of 0.681, and an occupation f1-score of 0.707 [20].

	Occupation	Gender	Birth Year
Logistic Regression	0.754	0.735	0.346
Random Forest	0.744	0.689	0.333
Support-Vector	0.738	0.706	0.349

Table 1: F1 Scores per Category for every Classifier

5 Conclusions

Overall, the three ML algorithms that were chosen produced relatively equal results for the classification categories (occupation, gender, and birth year). The classifications of occupation and gender were the most successful, as the ultimate metrics of the trained models were clearly superior to those retrieved from the baseline models. Classifying birth year was the most complicated process and yielded smaller metrics due to the number of possible classes. Comparing the algorithms by cRank, logistic regression appeared to be the most effective. In the future we would like to experiment with a recurrent neural network to determine if it will offer better results than the algorithms we utilized.

References

1. Argamon, S., Koppel, M., Pennebaker, J., Schler, J.: Automatically profiling the author of an anonymous text. Commun. ACM **52**, 119–123 (02 2009). https://doi.org/10.1145/1461928.1461959
2. Bengfort, B., Bilbro, R., Danielsen, N., Gray, L., McIntyre, K., Roman, P., Poh, Z., et al.: Yellowbrick (2018). https://doi.org/10.5281/zenodo.1206264, http://www.scikit-yb.org/en/latest/

3. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
4. Fraj, M.B.: In depth: Parameter tuning for svc (2018), https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769
5. Grant, R.: ndjson 0.3.1. Python Module (2018), python ndjson support
6. Hodge, A., Huang, Z., Price, S.: Classification of time period and author age in fiction (2019), student project at Northeastern University
7. Hodge, A., Price, S.: Artificial intelligence final report: Celebrity profiling 2020 (2020), student project at Northeastern University
8. Honnibal, M., Montani, I.: spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. To appear (2017)
9. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
10. Kim, T., Wurster, K.: emoji 0.5.4. Python Module (2014), identifies emojis in text
11. Koehrsen, W.: Hyperparameter tuning the random forest in python (2018), https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
12. Oliphant, T.E.: A guide to NumPy, vol. 1. Trelgol Publishing USA (2006)
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
14. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World. Springer (Sep 2019)
15. Qiao, F.: Logistic regression model tuning with scikit-learn - part 1 (2019), https://towardsdatascience.com/logistic-regression-model-tuning-with-scikit-learn-part-1-425142e01af5
16. Rangel, F., Rosso, P., Chugur, I., Potthast, M., Trenkmann, M., Stein, B., Verhoeven, B., Daelemans, W.: Overview of the 2nd Author Profiling Task at PAN 2014. In: Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.) Working Notes Papers of the CLEF 2014 Evaluation Labs. CEUR-WS.org (Sep 2014), http://ceur-ws.org/Vol-1180/
17. Van Der Walt, S., Colbert, S.C., Varoquaux, G.: The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering **13**(2), 22 (2011)
18. Varoquaux, G.: Joblib 0.16.0. Python Module (2010), python parallel computing
19. Wiegmann, M., Stein, B., Potthast, M.: Celebrity Profiling. In: 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019). Association for Computational Linguistics (Jul 2019)
20. Wiegmann, M., Stein, B., Potthast, M.: Overview of the Celebrity Profiling Task at PAN 2020. In: Working Notes Papers of the CLEF 2020 Evaluation Labs. CLEF and CEUR-WS.org (Sep 2020)