Spam or Not? Instagram Accounts Examined

Classifying Accounts with machine learning

Jamie Lynn Lufrano

2020

Fake spammer accounts plague social media platforms, so I thought it would be interesting to explore how data science could be used to identify real and “fake” (spam) accounts based on publicly-available attributes for the accounts.

I used a dataset I found on Kraggle (‘datasets\_145755\_339960\_instagram.csv’) that contains 696 entries with 12 features of Instagram accounts. These accounts are classified as real (“fake” == 0) or fake (“fake” == 1).

## Instructions to Run Code

The included code (Instagram\_Data\_Analysis.py) should run in any IDE.

## Examining the Features

Descriptions of the features contained in this data set:

* pic: user has a profile picture (1) or not (0)
* user #:len: ratio of the number of numerical chars in username to its length
* full words: full name in word tokens
* full #:len: ratio of number of numerical chars in full name to its length
* full = user: are username and full name the same (1) or not (0)
* desc len: length of characters of description/bio
* URL: has external URL (1) or not (0)
* private: private (1) or not (0)
* posts: number of posts
* followers: number of followers
* follows: number of follows
* fake: class (0 real/genuine, 1 fake/spammer)

After importing the data from ‘datasets\_145755\_339960\_instagram.csv’ into a dataframe and simplifying its labels, I copied the real (“fake” == 0) and fake (“fake” == 1) class data into two separate dataframes to examine their features for any noticeable patterns.

I calculated the mean and standard deviation for each feature for both dataframes, as well as the original dataframe which contained all of the data (both classes), which I exported to CSV files (insta\_df\_0\_mn\_sd.csv, insta\_df\_1\_mn\_sd.csv, and insta\_df\_mn\_sd.csv). These calculations are summarized in the table on page 2.

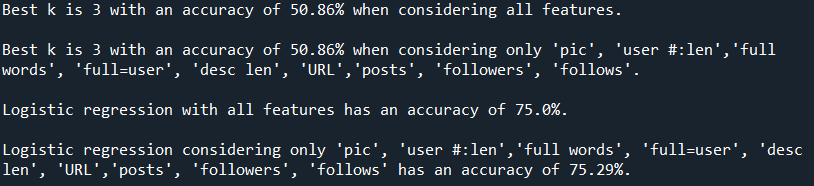
**Means and Standard Deviations for All Attributes for Real, Fake, and All Accounts**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| "fake" | "fake" == 0 | | "fake" == 1 | | All | |
| Feature | **μ (0)** | **σ (0)** | **μ (1)** | **σ (1)** | **μ (All)** | **σ (All)** |
| pic | 0.99 | 0.08 | 0.43 | 0.49 | 0.71 | 0.45 |
| user #:len | 0.04 | 0.1 | 0.29 | 0.24 | 0.17 | 0.22 |
| full words | 1.8 | 1.28 | 1.15 | 0.69 | 1.48 | 1.08 |
| full #:len | 0.01 | 0.04 | 0.08 | 0.19 | 0.04 | 0.14 |
| full=user | 0.01 | 0.08 | 0.07 | 0.25 | 0.04 | 0.19 |
| desc len | 41.88 | 43.67 | 4.94 | 19.64 | 23.41 | 38.57 |
| URL | 0.23 | 0.42 | 0 | 0 | 0.11 | 0.32 |
| private | 0.43 | 0.5 | 0.3 | 0.46 | 0.37 | 0.48 |
| posts | 197.92 | 516.51 | 8.57 | 26.3 | 103.24 | 377.76 |
| followers | 158157.2 | 1185914 | 142.62 | 384.27 | 79149.91 | 842281.8 |
| follows | 712.87 | 1073 | 397.3 | 944.16 | 555.09 | 1022.88 |

The means and standard deviations revealed some obvious attributes of real and fake accounts. The mean of 0.99 for the “pic” feature of real accounts with the low standard deviation of 0.08 suggest that these accounts almost always have a profile picture. The 0 mean and standard deviation for URL of fake accounts revealed that these accounts never had a URL.

There were several features that stood out to me as being unlikely to have a significant impact when predicting an account’s class. Real and fake accounts had similar means for whether an account was private or not. The ‘private’ attribute for real accounts had a mean of 0.43 with a standard deviation of 0.5, while fake accounts had a mean of 0.3 with a standard deviation of 0.46. Another attribute which seemed unlikely to significantly influence prediction was the ‘full=user’ attribute. Both real and fake accounts had low means for this attribute, with real accounts having a mean of 0.01 and fake accounts having a mean of 0.25. Therfore, a full name being the same as a username made an account more likely to be fake, but only marginally. Similarly, the low means for real and fake accounts for the ‘full #:len’ attribute of 0.01 and 0.19, respectively, suggest the number of numerical characters in the account’s full name will have a minimal impact on its classification.

In order to test my hypothesis about the ‘private’, ‘full=user’, and ‘full #:len’ attributes having limited impact in predicting the correct class for an Instagram account, I predicted the class using two different classifiers, k-NN and logistic regression, and checked their accuracy. For k-NN, k = [3, 5, 7, 9, 11] were checked for k and the best was determined to be k = 3. A random 50/50 training/testing split was used of the data for each classifier. As a control, the same 50/50 split was used to test the classifier with all features and reduced\_features\_1, which had the three aforementioned attributes removed. The results of this test are shown in the output screenshot on page 3.



The k-NN classifier produced the same accuracy when all features were used, as well as when the ‘private’, ‘full=user’, and ‘full #:len’ features were removed. This suggested to me these features were insignificant in predicting spam accounts. The reduced features resulted in a slightly higher accuracy than all features with logistic regression, supporting the theory that these three attributes aren’t useful in training a classifier. These results were consistent across numerous runs of the code with different testing/training splits.

I computed the correlation and plotted heatmaps for both the real and fake data in an attempt to see if correlations revealed anything about the features. These heatmaps are shown below (and saved as insta\_0\_heatmap\_red.jpg and insta\_1\_heatmap\_red.jpg). I used a reduced list of features (see reduced\_features\_1 on page 4). There is commented code showing the initial run for all features (and the heatmaps can be found in my zip file).

A close up of a keyboard

Description automatically generatedA picture containing text, crossword

Description automatically generated

I didn’t ascertain any useful info from the heat maps. The correlations I found were common sense and didn’t help me narrow down the most significant attributes. For example, in the genuine data (‘fake’ == 0), the strongest positive correlation is between ‘URL’ and ‘desc len,’ which makes sense since a URL would likely make a description longer than one which does not contain a URL. The fake accounts showed the strongest positive correlation between ‘followers’ and ‘follows.” It stands to reason that the fake accounts that are attempting to be perceived as more legitimate with increased follower/follow counts would increase both of these counts.

I also created pairwise plots for the features but commented out the code because I didn’t find them particularly helpful. I continued to narrow down the features to various combinations and expanded my testing to numerous classifiers. I have described the groups of attributes that were tested in my code below:

* ‘features’ – as a baseline, every classifier was tested using all of the attributes
* ‘reduced\_features\_1’ – all of the features except ‘private,’ ‘full=user’, and ‘full #:len’ as noted above
* ‘reduced\_features\_2’ – ‘pic’, ‘desc len', 'URL', 'posts', 'followers', and 'follows', which stood out as perhaps having a significant impact on account classification because their means either differed dramatically or had a value that suggested a characteristic of the vast majority of accounts in a specific class (fake == 0 or fake == 1)
* ‘reduced\_features\_3’ - 'desc len', 'posts', 'followers', and 'follows', which were the remaining attributes without the two Boolean features (‘pic’ and ‘URL’) in the hopes that training the classifiers with floats and integers would improve accuracy.
* ‘reduced\_features\_4’ – the three features which varied the most in values were used: 'posts', 'followers', and 'follows'
* ‘reduced\_features\_5’ – the list was further reduced to the 'followers' and 'follows' features in order to assess if prediction with higher accuracy was possible from only two features

## Testing Classifiers

My least accurate classifiers were K-NN using the best k (k = 3) and Logistic regression, both of which performed near or worse than a coin toss as shown in the two tables below:

**K-NN Using K = 3:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | 0 | 0 | 161 | 187 | 46.26% | 0.00% | 100.00% |
| **reduced\_features\_1** | 0 | 0 | 161 | 187 | 46.26% | 0.00% | 100.00% |
| **reduced\_features\_2** | 0 | 0 | 161 | 187 | 46.26% | 0.00% | 100.00% |
| **reduced\_features\_3** | 0 | 0 | 161 | 187 | 46.26% | 0.00% | 100.00% |
| **reduced\_features\_4** | 0 | 0 | 161 | 187 | 46.26% | 0.00% | 100.00% |
| **reduced\_features\_5** | 1 | 0 | 161 | 186 | 46.55% | 0.53% | 100.00% |

**Logistic Regression**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | 9 | 0 | 172 | 167 | 52.01% | 5.11% | 100.00% |
| **reduced\_features\_1** | 6 | 0 | 172 | 170 | 51.15% | 3.41% | 100.00% |
| **reduced\_features\_2** | 0 | 0 | 172 | 176 | 49.43% | 0.00% | 100.00% |
| **reduced\_features\_3** | 0 | 0 | 172 | 176 | 49.43% | 0.00% | 100.00% |
| **reduced\_features\_4** | 0 | 0 | 172 | 176 | 49.43% | 0.00% | 100.00% |
| **reduced\_features\_5** | 0 | 0 | 172 | 176 | 49.43% | 0.00% | 100.00% |

As shown in the preceding tables, both of these classifiers favored negative classifications with 100.00% specificity across all groupings of features. Feature groupings for K-NN had little impact with most having the same 46.26% accuracy, except for the smallest grouping (reduced\_features\_5), which produced a slightly higher accuracy of 46.26%. All features had the highest accuracy for logistic regression, which was only 52.01%. I was not only surprised at how poorly these classifiers performed but also how little difference the included data seemed to matter when applying these classifiers for this data set.

Like K-NN and logistic regression, a simple classifier I created (('full-user' < 0.25 or 'URL' != 0) and 'posts' > 15 accuracy and 'pic' == 1) performed similar at only 53.74% accuracy.

Naive Bayesian performed with a higher accuracy than K-NN and logistic regression with its highest accuracy reaching 69.25%, as shown in the table below.

**Naive Bayesian**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | 156 | 87 | 85 | 20 | 69.25% | 88.64% | 49.42% |
| **reduced\_features\_1** | 156 | 87 | 85 | 20 | 69.25% | 88.64% | 49.42% |
| **reduced\_features\_2** | 156 | 87 | 85 | 20 | 69.25% | 88.64% | 49.42% |
| **reduced\_features\_3** | 156 | 87 | 85 | 20 | 69.25% | 88.64% | 49.42% |
| **reduced\_features\_4** | 161 | 101 | 71 | 15 | 66.67% | 91.48% | 41.28% |
| **reduced\_features\_5** | 165 | 129 | 43 | 11 | 59.77% | 93.75% | 25.00% |

This classifier again surprised me by its lack of variability in performance between the groupings. The highest accuracy of 69.25% was shared by using all features as well as reduced\_features\_1, reduced\_features\_2, and reduced\_features\_3, so even as few as four features ('desc len', 'posts', 'followers', and 'follows') managed to produce the same result. Naive Bayesian had a far better sensitivity than was seen in the previously tested classifiers, with the highest accuracies also sharing a TPR of 88.64%. The feature groupings reduced\_features\_4 and reduced\_features\_5 had higher sensitivities of 91.48% and 93.75% respectively, but their accuracies dropped to 66.67% and 59.77%.

Decision Tree was one of my two strongest classifiers, as shown in the below table

**Decision Tree**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | 153 | 26 | 150 | 19 | 87.07% | 88.95% | 85.23% |
| **reduced\_features\_1** | 158 | 25 | 151 | 14 | 88.79% | 91.86% | 85.80% |
| **reduced\_features\_2** | 158 | 31 | 145 | 14 | 87.07% | 91.86% | 82.39% |
| **reduced\_features\_3** | 148 | 33 | 143 | 24 | 83.62% | 86.05% | 81.25% |
| **reduced\_features\_4** | 155 | 39 | 137 | 17 | 83.91% | 90.12% | 77.84% |
| **reduced\_features\_5** | 145 | 32 | 144 | 27 | 83.05% | 84.30% | 81.82% |

The reduced\_features\_1 grouping produced the highest accuracy predictions using Decision Tree, reaching an accuracy of 88.79% with 91.86% TPR and 85.80% TNR. All features and reduced\_features\_2 produced the second most accurate Decision Tree classifiers with 97.07% accuracy but differed in their sensitivities and specificities.

Random Forest produced some of my most accurate classifiers, as shown in the below table:

**Random Forest**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **(N, D)** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | (7, 5) | 155 | 8 | 169 | 16 | 93.10% | 90.64% | 95.48% |
| **reduced\_features\_1** | (9, 5) | 158 | 11 | 166 | 13 | 93.10% | 92.40% | 93.79% |
| **reduced\_features\_2** | (4, 5) | 152 | 16 | 161 | 19 | 89.94% | 88.89% | 90.96% |
| **reduced\_features\_3** | (6, 4) | 160 | 21 | 156 | 11 | 90.80% | 93.57% | 88.14% |
| **reduced\_features\_4** | (4, 5) | 147 | 16 | 161 | 24 | 88.51% | 85.96% | 90.96% |
| **reduced\_features\_5** | (3, 3) | 138 | 13 | 164 | 33 | 86.78% | 80.70% | 92.66% |

All features and reduced\_features\_1 tied for the highest accuracy of Random Forest and all my other classifiers, with an accuracy of 93.10% with Random Forest. They differed in their sensitivities and specificities with all features favoring negative predictions with a specificity of 95.48% and a sensitivity of 90.64%. Negative predictions were also favored by the Random Forest classifier produced with reduced\_features\_1 but its sensitivity and specificity were more comparable, at 92.40% and 93.79%, respectively

To further test my results, I created an experimental data set with the data from my own Instagram account, two other real accounts, and three spam accounts (‘experimental\_data.csv’), imported it into a dataframe from the CSV, and used it as the test data on my two highest classifiers (Decision Tree and Random Forest) trained on a 50/50 split of the original data. The size of the data set made it statistically insignificant but I was interested to see the classifications.

Decision Tree was less accurate for this experimental data, with reduced\_features\_5 producing the highest accuracy of 83.33%, as shown in the table below:

**Decision Tree for Experimental Data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | 0 | 1 | 2 | 3 | 33.33% | 0.00% | 66.67% |
| **reduced\_features\_1** | 2 | 1 | 2 | 1 | 66.67% | 66.67% | 66.67% |
| **reduced\_features\_2** | 2 | 1 | 2 | 1 | 66.67% | 66.67% | 66.67% |
| **reduced\_features\_3** | 2 | 1 | 2 | 1 | 66.67% | 66.67% | 66.67% |
| **reduced\_features\_4** | 2 | 1 | 2 | 1 | 66.67% | 66.67% | 66.67% |
| **reduced\_features\_5** | 2 | 0 | 3 | 1 | 83.33% | 66.67% | 100.00% |

Like my previous results, Random Forest proved to be the most accurate of the classifiers for the experimental data set, producing a 100.00% accuracy for all but one of the groupings, as shown in the table below.

**Random Forest for Experimental Data**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **(N, D)** | **TP** | **FP** | **TN** | **FN** | **accuracy** | **TPR** | **TNR** |
| **all** | (1, 2) | 3 | 0 | 3 | 0 | 100.00% | 100.00% | 100.00% |
| **reduced\_features\_1** | (3, 1) | 3 | 1 | 2 | 0 | 83.33% | 100.00% | 66.67% |
| **reduced\_features\_2** | (8, 4) | 3 | 0 | 0 | 3 | 100.00% | 100.00% | 100.00% |
| **reduced\_features\_3** | (7, 1) | 3 | 0 | 3 | 0 | 100.00% | 100.00% | 100.00% |
| **reduced\_features\_4** | (2, 1) | 3 | 0 | 3 | 0 | 100.00% | 100.00% | 100.00% |
| **reduced\_features\_5** | (3, 1) | 3 | 0 | 3 | 0 | 100.00% | 100.00% | 100.00% |

I also was intrigued that my feature grouping that tied or bested all features in my data set (reduced\_features\_1) tied for the second most accurate grouping in Decision Tree and was the least accurate grouping for Random Forest with my experimental data set.

## Results

reduced\_features\_1 tied or outperformed classifiers trained on all features, tying it for the highest accuracy of all the classifiers (Random Forest). This grouping had three fewer features than all (‘private’, ‘full=user’, and ‘full #:len’), which had been removed based on the mean and standard deviation data I calculated early in my project, suggesting my supposition that these features would be insignificant in classifying real and fake Instagram accounts was correct.

## Conclusion

Based on this data set, the smallest grouping of features that I would use to produce the most accurate results for classifying real and fake Instagram accounts would be: 'pic', 'user #:len', 'full words', 'desc len', 'URL', 'posts', 'followers', 'follows'.

# Works Cited

Bakhshandeh, B. (2019). *Instagram fake spammer genuine accounts*. Retrieved from Kraggle: https://www.kaggle.com/free4ever1/instagram-fake-spammer-genuine-accounts/data?select=train.csv