A Comparison of Various Network Features for Temporal Link Prediction in Twitter

Fred Mubang
fmubang@mail.usf.edu
University of South Florida
Tampa, FL

Abstract

In this work, I describe my temporal link prediction experiments with the social media platform Twitter. Specifically, I trained various Logistic Regression models with different combinations of network features, in order to examine which features help the most in temporal link prediction. I find that the edge weight time series features are the only features required for the temporal link prediction task. Furthermore, this edge only model was successfully able to outperform the baseline model in 3 different performance metrics. In this work I also performed experiments to determine the best number of time steps to use for creating input temporal features and I found that 2 time steps was the best for this particular Twitter dataset. Lastly, I show results from Logistic Regression coefficient analysis in order to understand which network features help the most with temporal link prediction.

Keywords Network Science, Temporal Link Prediction, Twitter, Binary Classification, Imbalanced Learning, Machine Learning, Logistic Regression

ACM Reference Format:
Fred Mubang. 2018. A Comparison of Various Network Features for Temporal Link Prediction in Twitter. In *Network Science Final Project '20*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

In the current day and age, the usage of social media has become more prevalant than ever. There are countless number of social media sites that are widely used by people all throughout the world. Twitter, is one of the most popular among them. As of the time of this writing, Twitter has 330 million monthly active users and 145 million daily active users [1]. Due to the wide usage of social media, many researchers have taken interest in further understanding the network dynamics of Twitter. In this work specifically, I am interested in understanding the dynamics of user interactions over time. To that end, I wanted to answer 3 questions:

1. Can a Machine Learning model be trained to perform temporal link prediction better than a *Persistence Model* baseline? The *Persistence Model* is the model in which the predictions for time step $t_{present} + 1$ are generated by simply outputting the ground truth from time step $t_{present}$.
2. What types of network features aid in the Twitter temporal link prediction task?
3. How many time steps of temporal input features is required for a model to successfully perform the temporal link prediction task? I define the number of input time steps as the *lookback factor*. Intuitively, it means, by how many time steps does a model need to "look back" in order to predict the link at time step $t_{present} + 1$?

In order to answer these questions I trained multiple Logistic Regression models, each with different combinations of network features and lookback factors. The lookback factors ranging from 1 to 4 were used. The different types of network features are as follows: Edge Weight, Child Outdegree [2], Parent Indegree [2], Child Betweenness Centrality [3], Child Page Rank [4], Child Closeness Centrality [3], Parent Betweenness Centrality [3],Parent Page Rank [4], and Parent Closeness Centrality [3].

In my experiments, I found that the edge weight feature type was the only feature required in order to successfully outperform the *Persistence Baseline*. Its AUC, Micro F1, and Macro F1 scores were 0.6106, 0.9253, and 0.552 respectively, while the baseline's scores were 0.5799, 0.8178, and 0.549. Furthermore, I found that a lookback factor of 2 was the best lookback factor out of the 4 that I tried. The granularity of time steps was daily, so this means, that for this particular Twitter dataset, in order to predict whether an edge exists at $t_{present} + 1$, you only need the edge weight values from time step $t_{present} + 1 - 1$ and $t_{present} + 1 - 2$ as features. Further in this work I will explain these experiments and results in more detail. The contributions of this paper are as follows:

Network Science Final Project '20, Dec 8, 2020, Tampa, FL
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

1. I trained multiple models with different combinations of network features and lookback factors and analyze these results to show how each combination of feature types and lookback factors affect temporal link prediction performance.
2. I show the Logistic Regression coefficients of several models in order to illustrate the importance of each network feature of the overall temporal link prediction task.

2 The Temporal Link Prediction Problem

2.1 Preliminaries

Let $G^{temporal}$ be a temporal graph. $G^{temporal}$ can be thought of as a sequence of static graph snapshots such that $G^{temporal} = \{G_1, G_2, ...G_{present}\}$. Each graph snapshot corresponds to the state of $G^{temporal}$ some time step t. Furthermore, each temporal graph, G_t can be thought of as a tuple of sets, (V_t, E_t), in which V_t and E_t are sets containing the nodes and edges, respectively, that are active in G_t. Using this information, I further define a global tuple of nodes and edges, called V, E. V contains every node, $u \in V$, across each graph in $G^{temporal}$, and E contains every edge, $e \in E$ across each in graph in $G^{temporal}$.

In Twitter, the nodes and edges correspond to users and user interactions, respectively. In this work, I define a user interaction is defined as the act of some user A retweeting some user B. If the user interaction is a self loop, that means that user A is either replying to his own tweet, or user A is posting a tweet.

The edge set E_t contains tuples of the form:

$$e_t = (u^{child}, u^{parent}, w_t(u^{child}, u^{parent})).$$

u^{child} and u^{parent} are the child and parent nodes, respectively. In Twitter, u^{child} is the user who is performing some action in response to some post made by u^{parent}. $w_t(u^{child}, u^{parent})$ represents the number of times u^{child} interacted with u^{parent}. For example, if u^{child} retweeted 3 posts made by u^{parent} at time step t, then $w_t(u^{child}, u^{parent}) = 3$.

2.2 Defining the Temporal Link Prediction Problem

Using this definition of a temporal graph, I can now define the temporal link prediction problem which is as follows. Let us say you are given a feature vector, $x(e, t_{initial}, t_{present})$. This vector contains temporal features related to some edge e, that span from some initial time step, $t_{initial}$ up to the present time step, $t_{present}$. Predict whether or not edge e will be active at future time step $t_{present} + 1$. In other words, predict if $w_{t_{present}}(u^{child}, u^{parent}) \geq 1$ or $w_{t_{present}}(u^{child}, u^{parent}) = 0$. Figure 1 is a diagram of the temporal link prediction pipeline. Temporal features related to an edge e are fed into a Machine Learning model. The model then predicts 1 if the edge e will be active at time step $t_{present+1}$, and 0 otherwise.

3 Background and Related Work

3.1 Temporal Link Prediction

There have been several previous works related to the temporal link prediction task. Some approaches, in a similar fashion to this work, utilize binary machine learning algorithms along with hand-crafted features in order to predict the links. For example. the authors of [5] use the momentum effect, homophily effect, and the common factor effect network statistics in order to build a logistic regression model for temporal link prediction in the social media platform, Sina Weibo.

Some other approaches utilize graph embedding using neural networks in order to perform temporal link prediction. The authors of [6] introduced *tNodeEmbed*, which is an embedding algorithm based on the static-network-based *node2vec* algorithm [7]. The authors of [8] introduced dyngraph2vec, which utilizes an LSTM autoencoder in order to embed nodes for link prediction.

3.2 Centrality Metrics

In this work, I utilize several network centrality metrics for the link prediction task. They are as follows

1. The node indegree and outdegree: These metrics capture how many incoming and outcoming edges a node has, respectively, [2].
2. Betweenness Centrality: This metric captures how many times a particular node is in the shortest path between two other nodes [3].
3. Closeness Centrality: This metric is the reciprocal of the sum of length of the shortest paths between the node and all other nodes in the graph [3]. Nodes that tend to be "closer" to all other nodes will have a higher value.
4. Page Rank: This metric measures how influential node is on a given network [4].

4 Data Information and Preprocessing

As previously mentioned, I used a Twitter dataset for my experiments. The tweets are related to China's OBOR (One Belt One Road) Initiative. The tweets span from April 10 to June 29, 2020. The time step granularity I used for my experiments was daily, so there were 81 total time steps. In order to avoid dealing with noisy data, I removed all nodes that were active in less than 10 time steps in the entire dataset. I was mainly interested in viewing the activity of the more active edges. I used Networkx [9] and Pandas [10] to create the temporal graph snapshots.

After this initial preprocessing step, the dataset contained 3,680 unique nodes, 7,067 static edges, and 31,728 temporal edges. Note that there are more temporal edges than static edges because an edge, e, can appear in multiple time steps. For example, if static edge, e appears in time steps t_1, t_2, and

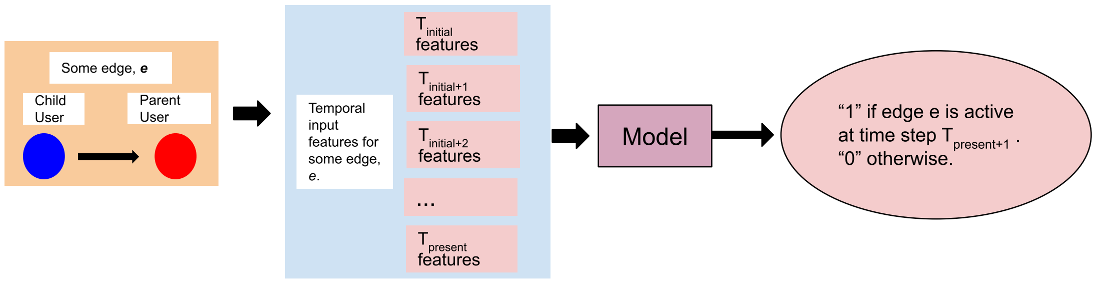

Figure 1. The pipeline for a temporal link prediction. Temporal features related to an edge, *e* are fed into a Machine Learning model.

t_3, there is 1 static edge (e), but 3 temporal edges, e_{t_1},e_{t_2}, and e_{t_3}.

5 Feature and Lookback Methodology

5.1 Feature Types

As previously mentioned, the different types of network features are as follows:

1. Edge Weight
2. Child Outdegree [2]
3. Parent Indegree [2]
4. Child Betweenness Centrality [3]
5. Child Page Rank [4]
6. Child Closeness Centrality [3]
7. Parent Betweenness Centrality [3]
8. Parent Page Rank [4]
9. Parent Closeness Centrality [3]

Different combinations of these feature types were used in order to construct *feature type sets*. Table 1 contains each category used. The intuition for each category is explained here.

1. E: This feature set contains the edge weight feature type only. I created this set to see if the edge weight history is all you need in order to predict if an edge will be active at a future time step or not.
2. All: This feature type set includes all 9 of the feature types.
3. No-E: This feature set contains the 9 feature types except the edge weight feature type. Note that the edge weight feature type is the feature type that is most correlated with the output value. Since the prediction task is to predict whether an edge will be active in a future time step, it stands to reason that the most relevant feature type would be the edge weight. I removed this feature type in this set to see if the model could still perform decently well using other network features besides this most relevant one.
4. EOI: This feature set just contains the edge weight, child outdegree, and parent indegree histories. I wanted to see if these 3 main feature types were all you need because intuitively, these 3 features are the most related to the prediction task.
5. No-EOI: This feature type set contains 6 feature feature types. It contains all the feature types minus the edge weight features (E), the child outdegree features (O), and the parent indegree features (I). The reason for creating this set was to see if a model could predict the future of an edge without features that more explicitly "show" the history of a given edge.

5.2 Use of Different Lookback Factors

In addition to different feature sets, I also wanted to see how many previous time steps of history were necessary in order for a model to properly predict the future of an edge weight. To that end, I tried 4 different lookback factors: 1, 2, 3, and 4.

5.3 Use of Multiple Datasets

Since there were 5 different feature sets, and 4 different lookback factors tried, I ended up training 20 different models, each with its own dataset.

6 Train and Test Methodology

6.1 Train and Test Splits

After creating the 20 data sets, the next step was to split each data set into train and test subsets. As previously mentioned, the train prediction period spanned from April 10 to June 15 2020 (67 time steps) and the test period spanned June 16 to June 29 2020 (14 days).

There were 74,580 training samples and 89,636 test samples. Since the task was a binary prediction task, I let class 1 represent a temporal edge sample that is active in the next time step, and I let class 0 represent a temporal edge sample that is not active in the next time step. In the training set, class 0 had 49,720 samples and class 1 had 24,860 samples. In order to avoid an imbalanced learning issue, I created 2 negative (class 0) samples for every positive (class 1) sample. That is why the ratio of positive to negative samples is 1:2. For the test set, the number of class 0 samples was 82,768 and

Feature Type Category	Feature Types Used	Number of Categories
EOI	Edge Weight, Child Outdegree, Parent Indegree	3
E	Edge Weight	1
No-EOI	Child Betweenness Centrality, Child Page Rank, Child Closeness Centrality, Parent Betweenness Centrality, Parent Page Rank, Parent Closeness Centrality	6
All	Edge Weight, Child Indegree, Child Betweenness Centrality, Child Page Rank, Child Closeness Centrality, Parent Outdegree, Parent Betweenness Centrality, Parent Page Rank, Parent Closeness Centrality	9
No-E	Child Indegree, Child Betweenness Centrality, Child Page Rank, Child Closeness Centrality, Parent Outdegree, Parent Betweenness Centrality, Parent Page Rank, Parent Closeness Centrality	8

Table 1. The feature type categories used in the link prediction experiments.

the number of class 1 samples was 6,868. No sampling was used for the test set. For data processing, I used Networkx [9], Pandas [10], and Numpy [11] to create the train/test sets. Furthermore, I used Scikit [12] for MinMax scaling and for the Logistic Regression algorithm.

6.2 Model and Metrics Used

Each of the 20 datasets was trained and tested with the Logistic Regression model, which is a common model for binary classification tasks. Performance was measured using the AUC, Micro F1, and Macro F1 metrics, which are common metrics for binary classification tasks.

6.3 F1 score

Let TP be the number of true positives. Let FP be the number of true positives, and let FN be the number of false negatives. Using this information, we can derive the F1 score as follows [13]:

$$F1 = 2 * \frac{TP}{2TP + FP + FN}$$

.

In this work we obtained both the Micro and Macro F1 scores. The Micro F1 score can be obtained by treating class 1 as the positive class, and class 0 as the negative class, and then plugging the true positives, false negatives, and false positives into the above formula [12]. The Macro F1 score involves calculating two F1 scores - the F1 score if class 0 is the positive class, and then the F1 score if class 1 is the positive class. These two scores are then averaged without weighting [12].

6.4 AUC Score

The AUC score provides an aggregate measure of performance across all classification thresholds in the ROC curve. Intuitively, this metric measures the following: What is the probability that a given model ranks a random positive sample more highly than a random negative sample. In other words, the AUC metric is more concerned with the relative separation of positive samples from negative samples.

7 Results and Discussion

7.1 Classification Results

Table 2 shows the results for each of the 20 Logistic Regression models. The *Ft. Category* column shows which feature set was used, and the *LB* column shows which lookback factor was used. The *Model Tag* column shows the name of each model.

As one can see in the table, the *LR-EOI-2* and *LR-E-2* models performed the best. The *LR-EOI-2* model had AUC, Micro F1, and Macro F1 scores of 0.6172, 0.9252, and 0.5496, respectively. The *LR-E-2* model had scores of 0.6106, 0.9253, and 0.552.

The *LR-EOI-2* model had the highest AUC of 0.6172 and the *LR-E-2* model had the best Micro and Micro F1 scores of 0.9253 and 0.552 respectively. The *LR-EOI-2* was the model that was trained with only the (1) edge weight, (2) child outdegree, and (3) parent indegree features. The *LR-E-2* model was the model trained with only edge weight features.

Overall, the *LR-E-2* was the best model because although it had a lower AUC than the *LR-EOI-2* model, it performed better in terms of Micro F1 and Macro F1. Furthermore, *LR-E-2* also managed to outperform the *Persistence Baseline* (the shifted) model. The *Persistence Baseline*'s scores were 0.5799, 0.8178, and 0.5489.

From the results I make the observation that the only features needed for edge classification are the edge weight features. Adding the child outdegree and parent indegree features only increased the AUC slightly, and it made the Micro F1 and Macro F1 decrease slightly. So, the main takeaway from this work is that, for this particular Twitter dataset at least, the only relevant feature type in determining the future of an edge is the edge's weight history. Secondly, I note that a lookback factor of 2 yielded the best results. This shows that that the best edge classification results can be

Logistic Regression Results

Model Tag	Ft. Category	LB	AUC	Micro F1	Macro F1
LR-EOI-2	EOI	2	***0.6172***	0.9252	0.5496
LR-E-2	E	2	0.6106	***0.9253***	***0.552***
LR-E-4	E	4	0.5864	0.9249	0.5199
LR-E-1	E	1	0.5864	0.9249	0.5199
LR-E-3	E	3	0.5864	0.9249	0.5199
LR-EOI-3	EOI	3	0.5861	0.9244	0.5248
LR-EOI-4	EOI	4	0.586	0.9248	0.523
LR-EOI-1	EOI	1	0.5859	0.9249	0.5229
Persistence_Baseline	n/a	n/a	0.5799	0.8178	0.5489
LR-all-2	all	2	0.5743	0.9176	0.5373
LR-all-2	No-EOI	2	0.5662	0.904	0.5071
LR-No-EOI-2	No-E	2	0.5558	0.9119	0.4991
LR-No-E-4	No-EOI	4	0.5441	0.9041	0.5059
LR-No-EOI-3	No-EOI	3	0.5424	0.9041	0.5059
LR-No-EOI-1	No-EOI	1	0.5411	0.9041	0.5055
LR-No-EOI-3	all	3	0.5365	0.9184	0.5214
LR-all-4	all	4	0.5361	0.9136	0.5239
LR-all-1	all	1	0.5352	0.9135	0.5193
LR-all-3	No-E	3	0.5265	0.9122	0.4969
LR-No-E-4	No-E	4	0.5248	0.9122	0.4963
LR-No-E-1	No-E	1	0.5223	0.9122	0.4957

Table 2. The Logistic Regression model results for each feature set and lookback factor combination.

obtained only using the past 2 time steps of edge history. The granularity of time steps used in these experiments was daily, so in other words, the past 2 days of edge history are all that is need for the temporal link prediction task.

7.2 Coefficient Analysis

In order to understand the role of each feature in the edge prediction task, I chose 2 models to analyze their coefficients. Model 1 is the LR-EOI-2 model. This is the model that was trained on the edge weight, outdegree, and parent indegree feature types. Model 2 is the LR-No-EOI-2 model. This is the Logistic Regression model that was trained on all 9 feature types except the edge weight, child outdegree, and parent indegree. It is the opposite of the LR-EOI-2 model. I chose this model to see what features helped with the link prediction task if there are no features directly related to the edge output value.

Table 3 shows the feature importances of the LR-EOI-2 model. The *ft* column shows each feature. The "ts" prefix tag in each feature refers to the time step for that particular feature. For example the feature *ts_2_cur_edge_weight* feature refers to the edge weight feature in time step 2.

The *coeff* column shows the raw value of the Logistic Regression coefficient. The *normed importance* column shows the value of each coefficient normalized from 0 to 1. Values closer to 1 indicate more importance relative to other features. Lower values closer to 0 indicate less importance.

As one can see in table 3, the most importance features are the edge weight features. The time step 2 edge weight feature had an importnace of 0.5, and the time step 1 edge weight had an importance of 0.42. This makes sense, because if the goal is to classify the edge at time step 3, it is reasonable to assume that the edge weight at time step 2 would be the most important feature, followed by the edge weight at time step 1. The more recent the an edge weight feature is, the more important it will be in classifying the future of the edge of interest.

The child and parent degree features had very low importances, closer to 0. This shows that the degree features are relatively less important in comparison to the edge weight features for the temporal link prediction task for this particular model.

The *LR-No-EOI-2* model had no edge weight, indegree, or outdegree features, so it had to rely on other features instead for the link prediction task. In this model, the child page rank features were the most important features. The time step 1 child page rank had an importance of 0.25, and the time step 2 page rank feature had an importance of 0.2257. In 3rd place was the time step 2parent betweenness centrality

LR-EOI -2 Feature Importances		
ft	**coeff**	**normed importance**
ts_2_cur_edge_weight	19.3534	0.5091
ts_1_cur_edge_weight	16.1477	0.4248
ts_2_user_outdegree	-0.8322	0.0219
ts_1_parent_indegree	-0.8086	0.0213
ts_1_user_outdegree	-0.6071	0.0160
ts_2_parent_indegree	0.2630	0.0069

Table 3. LR-EOI -2 Model Feature Importances

feature, and in 4th place was the time step 1 child closeness centrality feature.

It is surprising that the child page rank feature would be such an important feature for edge classification. The page rank score of a user indicates how influential a user is to a network [4]. My intuition is that if anything, the parent page rank feature would be more important because I imagine that an influential parent at a particular time step would be an indicator of a potential future edge forming involving that parent. However, according to table 3 the parent page rank features have importances only totaling up to 0.03, a very low number.

It makes sense that the parent betweenness centrality is a somewhat important feature (with 0.12 importance) . The betweenness centrality measures how many pairs of users a particular user acts as a "bridge" between [3]. So, if a parent has a high betweenness centrality, it makes sense that it would be active in more edges. This makes sense especially for Twitter because tweets have a cascade structure. In cascade structures, bridge users are very important for the dissemination of a tweet, because they can potentially cause many other users to see a tweet.

Lastly, it also makes sense that the child closeness centrality would have some importance as well (0.09). The closeness centrality measures the inverse sum of the shortest path between a user and all other users [3]. So, if a particular user is close to many users, that user will be active in more edges.

8 Future Work and Conclusion

In this work, I discussed how I used different network features to perform temporal link prediction. I was able to train several models that outperformed a Persistence Baseline model in terms of AUC, Micro F1, and Macro F1. I found that the edge weight feature type is the only feature type needed in order to perform the prediction task. Adding other network features either did not help by much, or they actually decreased performance accuracy. I also found that a lookback factor of 2 is all that's needed in order to make proper predictions.

I then analyzed the coefficients of several models. I found that in the model using edge weight, child indegree, and

LR-No-EOI-2 Feature Importances		
ft	**coeff**	**normed importance**
ts_1_child_page_rank	4.2822	0.2508
ts_2_child_page_rank	3.8535	0.2257
ts_2_parent_b_cen	2.1410	0.1254
ts_1_child_close_cen	1.6625	0.0974
ts_2_child_b_cen	-1.2692	0.0743
ts_2_child_close_cen	1.0890	0.0638
ts_2_parent_page_rank	0.9450	0.0554
ts_1_parent_b_cen	-0.5265	0.0308
ts_2_parent_close_cen	0.4510	0.0264
ts_1_parent_close_cen	0.3439	0.0201
ts_1_parent_page_rank	-0.3243	0.0190
ts_1_child_b_cen	0.1847	0.0108

Table 4. The feature importances of the LR-No-EOI-2 model.

parent outdegree, the edge weight features accounted for about 92% of the model's coefficient weights. For the model using all features except the edge weight, indegree, and outdegree, I found that child page rank was the most useful feature. Future work would involve further analyzing why the child page rank seems to be more useful than parent page rank in the prediction task. Furthermore, I would also try using more pair-wise features besides just edge weight, such as Common Neighbors, as well as similarity between the child and parent follower/followee sets. Perhaps these features could aid with temporal link prediction better than the centrality features that I used.

9 Acknowledgments

Thank you to DARPA, PNNL, and Leidos for providing the Twitter and data for these experiments.

References

[1] M. IQBAL. Twitter revenue and usage statistics (2020). [Online]. Available: https://www.businessofapps.com/data/twitter-statistics/

[2] L. C. Freeman, "Centrality in social networks conceptual clarification." in *Soc. Networks 1*, 1978, p. 215–239.

[3] ——, "A set of measures of centrality based on betweenness." in *Sociometry*, 1977, pp. 35–41.

[4] S. Brin and L. Page., "The anatomy of a large-scale hypertextual web search engine," pp. 30(1–7):107–117, 1998.

[5] D. H. Jing Zhou and H. Wang, "A dynamic logistic regression for network link prediction," p. 60, October 2016.

[6] K. R. Uriel Singer, Ido Guy, "Node embedding over temporal graphs," in *Proceedings of the 28th International Joint Conference on AI (IJCAI-19)*, August 2019.

[7] J. L. Aditya Grover, "node2vec: Scalable feature learning for networks," in *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July.

[8] A. C. Palash Goyal, Sujit Rokka Chhetri, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," in *Knowledge-Based Systems,Volume 187*, January 2020.

[9] J. L. Aditya Grover, "Aric a. hagberg, daniel a. schult and pieter j. swart," in *Proceedings of the 7th Python in Science Conference*, Aug 2008, pp. 11–15.

[10] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445. Austin, TX, 2010, pp. 51–56.

[11] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] G. K. Christen, P., "Quality and complexity measures for data linkage and deduplication," p. 127–151, 2007.