Who is a Friend?

Determining whether or not two people are friends

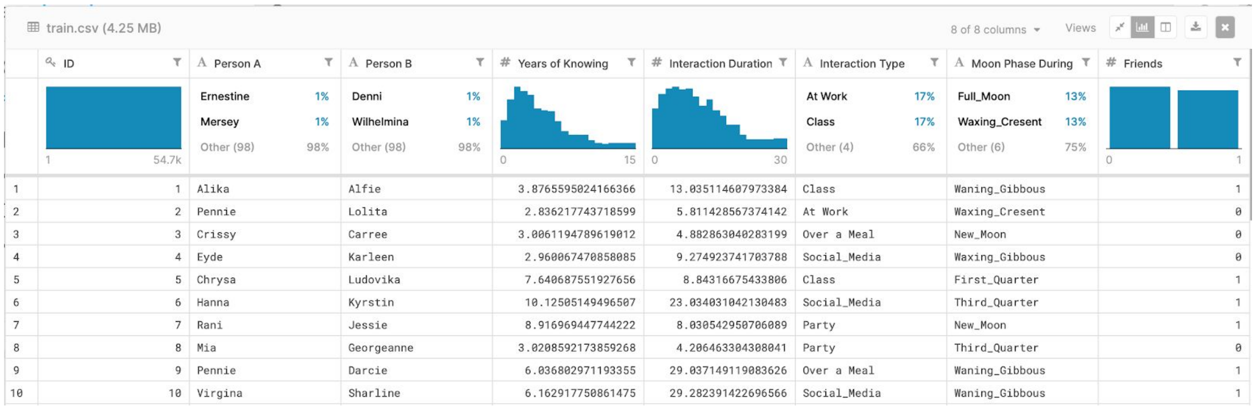
https://www.kaggle.com/c/whoisafriend/overview

Group 13
Grant Ferguson - gferguson@wpi.edu
Justin Kreiselman - jkreiselman@wpi.edu
Ken Morton - kjmorton@wpi.edu

Introduction

The goal of this competition is to predict whether or not two people are friends. We were given a set of data that recorded events from a group of people. While our feature space was limited, we were given some good information for each pair of people. The different features were ‘Person A’, ‘Person B’, ‘Years of Knowing’, ‘Interaction Duration’, ‘Interaction Type’, and ‘Moon Phase During.’ For a better idea of some sample data, we included a screenshot of the first 10 data points in our training data below. Then, for each ID (pair of two people), we would predict whether they were friends, denoted by 1, or not friends, 0. We are scored by the percentage of people we correctly predicted as friends.

In a time where it is very common to connect with people you meet on social media, we wanted to determine what factors influenced friendship. Was it simply the similarity of your names, or did it have to do with how long you knew the person? Did it matter how long you interacted with them or where the interaction took place? Does the phase of the moon truly make a friendship? These questions were interesting to us and we wanted to understand in more detail what exactly influenced friendships.


train.csv (4.25 MB)

8 of 8 columns

	ID	Person A	Person B	Years of Knowing	Interaction Duration	Interaction Type	Moon Phase During	Friends
	1 – 54.7k	Ernestine 1%, Mersey 1%, Other (98) 98%	Denni 1%, Wilhelmina 1%, Other (98) 98%	0 – 15	0 – 30	At Work 17%, Class 17%, Other (4) 66%	Full_Moon 13%, Waxing_Cresent 13%, Other (6) 75%	0 – 1
1	1	Alika	Alfie	3.8765595024166366	13.035114607973384	Class	Waning_Gibbous	1
2	2	Pennie	Lolita	2.836217743718599	5.811428567374142	At Work	Waxing_Cresent	0
3	3	Crissy	Carree	3.0061194789619012	4.882863040283199	Over a Meal	New_Moon	0
4	4	Eyde	Karleen	2.960067470858085	9.274923741703788	Social_Media	Waxing_Gibbous	0
5	5	Chrysa	Ludovika	7.640687551927656	8.84316675433806	Class	First_Quarter	1
6	6	Hanna	Kyrstin	10.12505149496507	23.034031042130483	Social_Media	Third_Quarter	1
7	7	Rani	Jessie	8.916969447744222	8.030542950706089	Party	New_Moon	1
8	8	Mia	Georgeanne	3.0208592173859268	4.206463304308041	Party	Third_Quarter	0
9	9	Pennie	Darcie	6.036802971193355	29.037149119083626	Over a Meal	Waning_Gibbous	1
10	10	Virgina	Sharline	6.162917750861475	29.282391422696566	Social_Media	Waning_Gibbous	1

Methods

For our baseline model we created a model that randomly returns 0 or 1 for each pair of friends. The results can be seen in the Random Baseline table below. The returned accuracy was ~50%, which is what we expected for having a 50/50 chance of guessing if two people are friends. With the baseline completed, we proceeded to develop our shallow models.

We created our shallow models using the sklearn package. We initially used a linear regression model because of its popularity as a starting point. For our initial attempt, we chose the features that we thought made the most impact on a friendship: Years of Knowing, Interaction duration and interaction type. We ran into errors when trying to run the model because of the string values for interaction type. To alleviate this issue, we tried to manually input float values between 0 and 1 representing how much we think each location can affect a friendship. We initially chose to ignore the names and moon phases because we believed that they did not have a meaningful effect on whether or not the pair are friends. Once we trained this model, we were only able to achieve a kaggle score of ~50%, nothing better than guessing. After doing some debugging, we noticed a small typo in our code where we were passing the wrong testing/training set to the model. After correcting this typo, our kaggle score increased to ~88%. However, when we calculated the accuracy and loss we obtained very poor results with 0.499 and 19.547 respectively. We are unsure as to why this happened. After speaking with the professor, we realized that the linear regression model was not the best model for our data, since we were predicting a binary outcome of 0 or 1. From this experiment we learned that even though the model can return a high accuracy it may not be the most optimal for the problem.

Our professor recommended we try a logistic regression model, something more suitable to our use case. He also recommended we use all of the features in the data set since there were only 6 features. We encoded our categorical data and included those features into the model using the sklearn LabelEncoder. The accuracy increased to ~92%, which led us to agree that logistic regression works better than linear regression for binary classification problems. We were curious to see which features had the most impact on the model, especially since we all laughed when we saw the moon phase as a data point.

To solve this problem, we used a class called RFE, or Recursive Feature Elimination. At a high level, RFE trains the model using the n important features. The importance feature is obtained using the coef_ attribute on our model after training, and the least important features are pruned from the current set of features. This process is recursively repeated on the pruned set until the desired number of features to select is reached. The feature importances can be seen in the Feature Importance Table below. Note that a value of +1 means the feature has a strong influence on ‘Friends’ or 1, while -1 means the feature has a strong influence on ‘Not Friends’ or 0.

As seen in table 3, we determined that years of knowing and interaction duration had the strongest influence on two people being friends. It was interesting to note that the person's name did not have much influence. One thing that surprised us is that the interaction type did not have much influence on determining friendship. We thought that people who met over a meal or at work might be more likely to be friends, but as this data shows, it doesn't have much importance.

Continuing to explore the relationship between features and accuracy, we created 5 models to see how the number of features affected the accuracy. The results can be seen in the Logistic Regression with RFE table below. We find it interesting how the data shows that the best calculated accuracy happened when two features were used and then decreased with three features and increased and level off with four features. We also find it interesting how the two features chosen were person b and the moon phase especially since we initially disregarded the moon phase when we first saw the training data. If we had to take anything from this model is that all data is important even if it seems completely unrelated. After creating these models we wanted to see how switching to a deep learning model would affect the results.

Our deep learning model was built with Keras. This was our first time using Keras, so we followed a tutorial on creating a simple binary classification neural network. For the first model, we used a single layer network with the sigmoid activation function and optimized the weights using Stochastic Gradient Descent and a binary_crossentropy loss function. We perform hyperparameter tuning on the batch size and epochs to help improve our accuracy. The model did not filter out any of the features. When we ran this model we achieved an accuracy of ~88% using 30 epochs and a batch size of 10. This result was on par with what we achieved using logarithmic regression. After we successfully built our first keras model we wanted to experiment more with it and see what would happen to the accuracy if we added more layers to our model. We created three more models using the same optimizations and hyperparameter tunings, but we had the models start with a relu activation function and on the last layer perform a sigmoid activation. The results can be seen in the Deep Learning Models without RFE table below, the accuracy seemed to increase as we went from 1 layer to 4 layers and the loss decreased significantly between layers 1 and 4. The best result can be seen with the four layer model which gave an accuracy of 91%. This could lead us to conclude that increasing the number of layers could improve the model, but we would need to keep in mind the fine line between high accuracy and overfitting. To stay consistent with our shallow models, we also performed RFE to see if limiting the number of features resulted in a similar performance compared to the logistic regression with RFE.

We then used RFE on the deep learning model to find the best feature set, and then used those features to train the neural network.. The results can be seen in the Deep Learning Models with RFE tables below. Similar with the shallow RFE models, they seem to prefer using two features, but chose different features to use. The best performing model is the one that uses 4 layers and the features: Interaction Duration, and Moon Phase During Interaction. This model achieved a score of 0.875 on kaggle and we calculate the accuracy to be 0.872. When comparing to the Logistic Regression model the deep learning appears to perform worse by a small amount. We found this a bit odd, since we would assume the neural network would give us a better result. This could be due to the different selected features between the models.

These are the models we created when challenging this Kaggle problem. If we had more time, we would've like to standardize the testing data to see what would happen if the features and hyperparameters were the same. In conclusion, we found that the practical aspect of this project was very helpful. It was a nice compliment to the theoretical focus of the class, and was fun to try some of the prebuilt models provided to us through sklearn and Keras.

Table of Results

Random Baseline		
Accuracy	Loss	Kaggle Score (Private)
0.502	17.208	0.499

Table 1: Training on a random baseline

Shallow Models			
Type	Accuracy	Loss	Kaggle Score (Private)
Linear Regression	0.499	19.547	.886
Logistic Regression	0.9155	0.1850	.917

Table 2: Training using shallow models

Feature Importance from Logistic Regression Model	
Feature	**Importance**
Years of Knowing	0.9457252294316647
Moon Phase During Interaction	-0.0026543058673065135
Interaction Type	-0.0069473988216924975
Person A	0.0002509868134093708
Interaction Duration	0.5226761112106989
Person B	0.0010581890516757307

Table 3: Feature Importance from Logistic Regression Model

Logistic Regression with RFE					
Number of Features	Features Used	Feature Ranking	Accuracy	Loss	Kaggle Score (Private)
1	Person B,	[3, 1, 5, 4, 6, 2]	0.87187831	0.280103477	0.87184
2	Person B, Moon Phase During Interaction	[2, 1, 4, 3, 5, 1]	0.91575560	0.185115532	0.91614
3	Person A, Person B, Moon Phase During Interaction	[1, 1, 3, 2, 4, 1]	0.91564591	0.185078542	0.91602
4	Person A, Person B, Interaction Duration, Moon Phase During Interaction	[1, 1, 2, 1, 3, 1]	0.91575560	0.185080354	0.91602
5	Person A, Person B, Years of Knowing, Interaction Duration, Moon Phase During Interaction	[1, 1, 1, 1, 2, 1]	0.91557278	0.185022380	0.91681

Table 4: Training using logistic regression and RFE

Deep Learning Models without RFE						
Activation Function	Epochs	Layer	Batch Size	Accuracy	Loss	Kaggle Score (Private)
Sigmoid	30	1	10	0.881	1.949	0.881
Relu + Sigmoid	30	2	20	0.906	0.203	0.908
Relu + Sigmoid	20	3	30	0.902	0.228	0.902
Relu + Sigmoid	20	4	30	0.916	0.184	0.916

Table 5: Training using a deep model without RFE

Deep Learning Models with RFE							
Activation Function	Filtered Features	Epochs	Layer	Batch Size	Accuracy	Loss	Kaggle Score (Private)
Sigmoid	Years of Knowing, Moon Phase During Interaction	20	1	30	0.868	0.280	0.868
Relu + Sigmoid	Years of Knowing, Interaction Type	30	2	30	0.874	0.267	0.874
Relu + Sigmoid	Person A, Person B	20	3	10	0.516	0.693	0.513
Relu + Sigmoid	Interaction Duration, Moon Phase During Interaction	30	4	30	0.872	0.239	0.875

Table 6: Training using a deep model with RFE

Conclusions

During our project we tried multiple techniques that had different pros and cons. We tried a few methods of feature encoding. Specifically, we focused on encoding the interaction type strings as a float between 0 and 1. Our first attempt at this was to manually assign a float value depending on how much we thought that the location had input on whether or not two people were friends. This was not the best approach, since we are manually tuning the variables based on what we feel is right. There isn't any math or science behind our reasoning, rather, experiences. This did not help our model as we were skewing the data to fit our thoughts of where friends would occur. This could've been helpful for creating some training data, but we did this for both the training and testing data so it would've negatively affected us down the line.

Eventually, we realized that encoding using the prebuilt libraries was a better approach to encoding categorical variables. We used the sklearn LabelEncoder to assign values to the following features: 'Person A', 'Person B', 'Interaction Type', and 'Moon Phase During Interaction'. While encoding categorical data didn't affect the accuracy of our model too much, it helped us to preprocess the data without adding personal bias to the data.

The technique we spent most of our time on was feature importance. Using this technique helped us to explore the data in which taught us that there is no stupid data. This was prevalent with our initial mocking of the moon phase data. When we saw it originally we scoffed at it and immediately ignored it, but when we got the feature importances especially with RFE it was some of the main features used by the models.

Using RFE had mixed results. It helped us and hindered us at the same time. There seemed to be no correlation between the features selected when running the models since it seems that the value of feature importance would change resulting in different features being used to train the model. The only way it seemed to help us was with showing a trend of how the number of features affected accuracy. We found that the model preferred to use two features and would result in a relatively high accuracy before decreasing and then increasing and levelling off. This would be interesting to explore more in depth especially with more available features.

All things considered, we found that the shallow models performed better on our data. Since we wanted to predict a binary classification of a simple ‘yes’ or ‘no’, we found the logistic regression model to be the best fit. It ran much faster than the deep models, and had the best kaggle score of all the models. It was also concise in its implementation, making it extensible if it were to be worked on in the future.

References

bhrt. “Starter KiT for ‘Who Is Your Friend?’” *Kaggle*, 7 Apr. 2020, https://kaggle.com/bhrt97/starter-kit-for-who-is-your-friend.

Brownlee, Jason. “How to Calculate Feature Importance With Python - Machine Learning Mastery.” *Machine Learning Mastery*, 29 Mar. 2020, https://machinelearningmastery.com/calculate-feature-importance-with-python/.

---. “How to One Hot Encode Sequence Data in Python - Machine Learning Mastery.” *Machine Learning Mastery*, 11 July 2017, https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/.

“Is There Any Way to Get Variable Importance with Keras?” *Stack Overflow*, https://stackoverflow.com/questions/44119207/is-there-any-way-to-get-variable-importance-with-keras. Accessed 12 May 2020.

Li, Lorraine. *Introduction to Multilayer Neural Networks with TensorFlow’s Keras API*. Towards Data Science, June 2019, https://towardsdatascience.com/introduction-to-multilayer-neural-networks-with-tensorflows-keras-api-abf4f813959.

marison. “My Approach for Who Is a Friend?” *Kaggle*, 8 Apr. 2020, https://kaggle.com/marison/my-approach-for-who-is-a-friend.

“Multinational List of Popular First Names and Surnames?” *Open Data Stack Exchange*, https://opendata.stackexchange.com/questions/46/multinational-list-of-popular-first-names-and-surnames. Accessed 12 May 2020.

Richards, Jason. *A Look into Feature Importance in Logistic Regression Models*. Towards Data Science, July 2019, https://towardsdatascience.com/a-look-into-feature-importance-in-logistic-regression-models-a4aa970f9b0f.

“Simple Binary Classification with Keras.” *Mc.ai*, 4 May 2019, https://mc.ai/simple-binary-classification-with-keras/.

sklearn.feature_selection.RFE — Scikit-Learn 0.22.2 Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html. Accessed 12 May 2020.

sklearn.preprocessing.LabelEncoder — Scikit-Learn 0.22.2 Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed 12 May 2020.

whoiskk. “Getting Started with WhoisaFriend?” *Kaggle*, 7 Apr. 2020, https://kaggle.com/whoiskk/getting-started-with-whoisafriend.