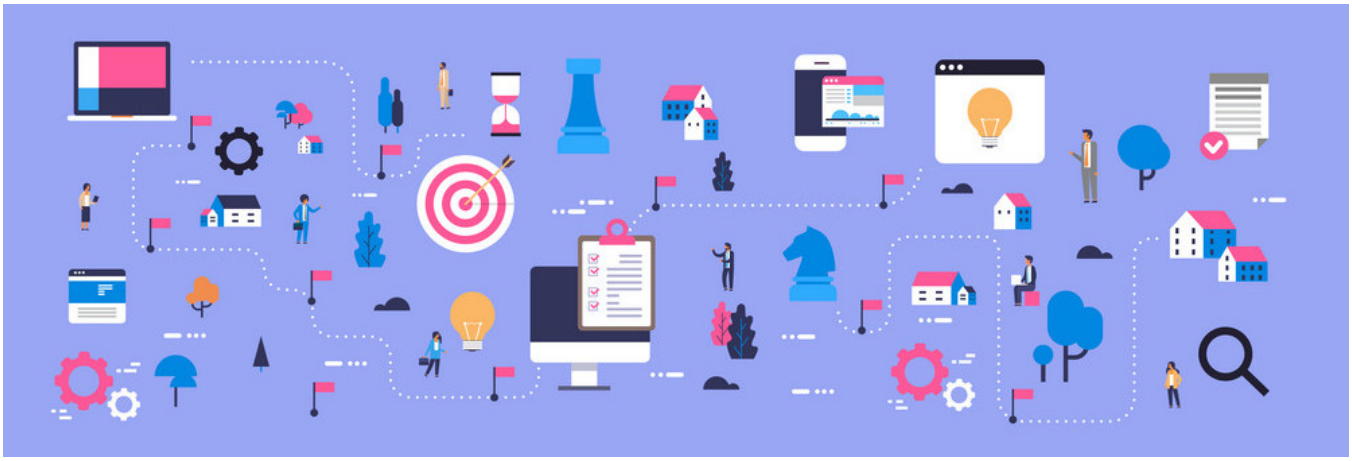
Facebook User Predictions - User09

Aditya Joshi
Matricule: 20114564
University of Montréal
aditya.joshi@umontreal.ca

Carolyne Pelletier
Matricule: 20143706
University of Montréal
carolyne.pelletier@umontreal.ca

Helgi Tómas Gíslason
Matricule: 20140669
University of Montréal
helgi.tomas.gislason@umontreal.ca

Lawrence Abdulnour
Matricule: 20019894
University of Montréal
lawrence.abdulnour@umontreal.ca



ABSTRACT

We propose and investigate gradient boosting decision tree methods in order to perform user profiling in social media. More specifically, we make use of this powerful model to predict gender, age and personality of facebook users and compare our results with standard baseline models. Furthermore, we examine different graph approaches for personality regression, such as DeepWalk and Node2Vec. Experiments with gradient boosting decision trees show improvements on all three tasks. User profiling in social media is at once an important tactic for applications that rely on personalisation, and a strong reminder that protected classes such as age and gender can be derived from social media data even if they are not explicitly stated in one's user profile. All code for this project can be found **here**.

KEYWORDS

datasets, facebook, social networks, gradient boosting decision trees, gradient boosting, random forests, neural networks, deepwalk, node2vec

IFT 6758. University of Montréal, Montréal, Québec, Canada

ACM Reference Format:
Aditya Joshi, Carolyne Pelletier, Helgi Tómas Gíslason, and Lawrence Abdulnour. 2019. Facebook User Predictions - User09. *IFT 6758. University of Montréal*. Montréal, Québec, Canada.

1 INTRODUCTION

User profiling has become a critical task in recent years in machine learning, as the identification of a user's interest domain has become valuable information, politically and economically. Indeed, a lot of research in the machine learning community evolve around user profiling and behavioral profiling techniques, which is particularly useful for recommendation systems.

Hence, in this project, we seek to contribute to this problem by performing profiling on three data sources, provided in a tabular format. More precisely, the goal of this project is to perform gender binary classification, age multi-class classification and personality regression using image, text and relation data. The image data is translated in namely, oxford features, which represent relevant facial point coordinates. In addition, the text data is extracted from the facebook user's status updates and transformed into word categories, whereas the relation data is extracted from page likes.

Deep learning methods particularly have been privileged to solve this kind of problem, due to their powerful nature. We, however, experimented with gradient boosting decision trees as they are robust methods for tabular or structured data, such as we have in our problem setting [2]. They also offer great performance and

execution speed. Our final model is thus achieved using the latter and we outperformed the baselines for all of the specified tasks.

2 METHODOLOGY

Our methodology involves predicting three main tasks. The first is a binary classification problem where we predict the gender of a user (female: 1, male: 0). The second is a multi-class classification problem in which we predict the age of a user within the following categories: “xx-24”, “25-34”, “35-49”, or “50-xx”. The third task is a personality prediction and involves providing a personality score between [1,5] for each of the five traits of the Big Five personality model: Openness to experience, Conscientiousness, Extroversion, Agreeableness, and Neuroticism.

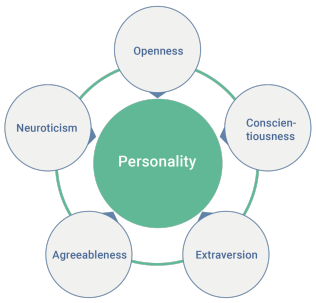


Figure 1: Traits of the Big Five personality model.

In order to achieve the mentionned predictions, we used several machine learning methods. What follows is a brief description of the ones we used.

A *neural network* is a network of artificial neurons. They are one of the fundamental machine learning models. They are inspired by biological neural networks.

Gradient boosting is a machine learning method which consists of an ensemble of weak prediction models.

Gradient boosting decision trees is the variant of gradient boosting where we specifically use decision trees for predictions.

Random forests are an ensemble learning method. They are a collection of decision trees. The predictions given are the collective vote or average of the output of said decision trees.

Relation v1 is the term we use for the manually feature engineered data set that we created from the relational data. See more information in section 3.4.

Vanilla nn is the term we use for the model we applied on the manually feature engineered data from section 3.4. This model was a simple three layer neural network which outputted log softmax probabilities for the age and gender tasks but unbound real values for the personality regression task using the relational v1 data.

The workings of the following two methods will be discussed in more detail in section 3.4.

DeepWalk is an approach for learning embeddings of the representations of nodes in a graph using random walks.

Node2Vec is a similar approach to DeepWalk except the randomness of the random walks can be controlled.

3 DATASETS AND METRICS

In this section we will describe the datasets and the evaluation measures used. In total, we have 9500 Facebook users with labels. The image data is composed of Oxford features with shape (7915, 66). The text data is made up of NRC features with shape (9500, 11) and LIWC features with shape (9500, 82). Finally the relation data is of shape (1671353, 2).

3.1 Profile data

When exploring the profile data, we observed quite a large class imbalance for the age labels (59.67% of users are 24 and under, 25.27% are of age 25 to 34, 11% of age 35 to 49 and 4.05% are 50 and older) as seen in Figure 2. This could cause issues with the accuracy of our predictions within subgroups. For the distribution of genders, We observed that the classes were relatively balanced (57.72% of users are females and 42.28% of users are males) as seen in Figure 3. Both genders seem to have similar distributions across different personality traits as seen with the Violin plots in Figure 4.

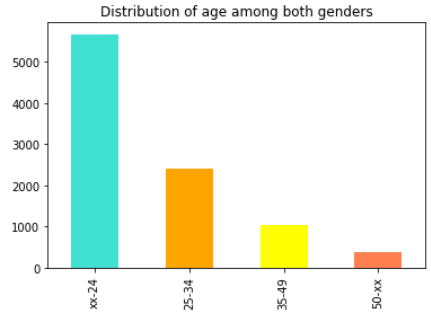


Figure 2: Distribution of age groups are not well balanced.

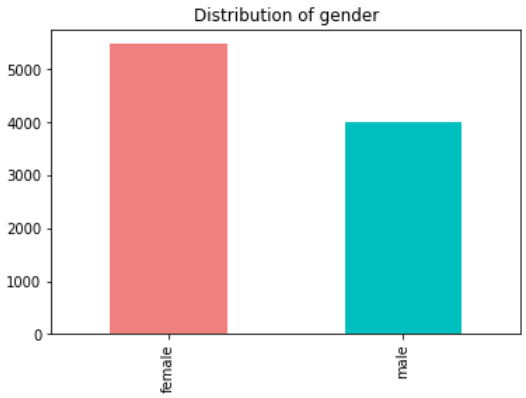


Figure 3: There is a less severe class imbalance in gender when compared to age.

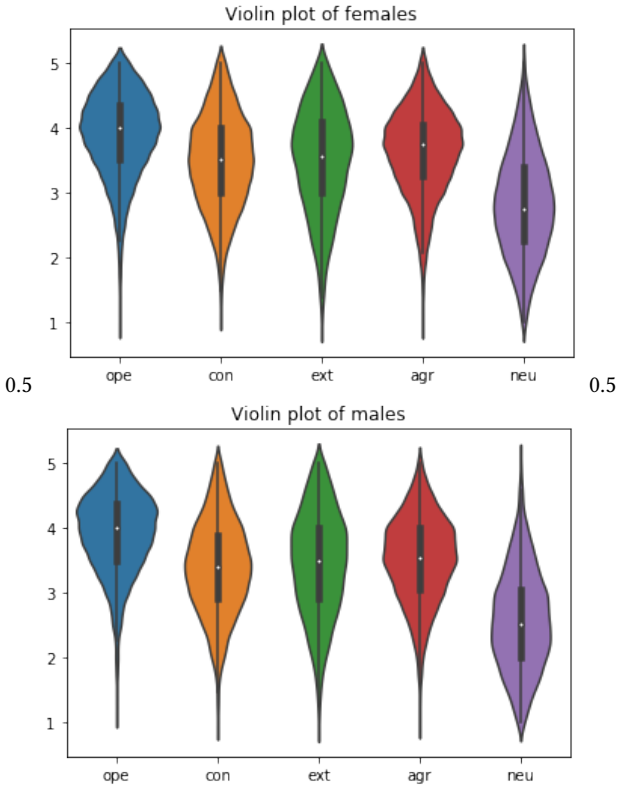


Figure 4: Violin plots which show that genders seem to have similar distributions across different personality traits.

3.2 Text features data

There were two different sets of text features: LIWC [9] and NRC [8]. LIWC features, which stands for Linguistic Inquiry and Word Count, is the gold standing in computerized text analysis. In this project, we used the 2015 framework of LIWC features which contains 82 different LIWC variables. The NRC Emoticon Lexicon on the other hand, helps describe the nuances of emotion and can be used to identify personality and contains 11 different variables.

In our data exploration phase for text features, we performed outlier detection via the *Isolation Forest* [6] algorithm from sklearn. It randomly "isolates" observations using recursive partitioning, and observations with a noticeably shorter path are likely to be outliers. What we observed is that outliers for LIWC features have extremes values as seen in Figure 5 and outliers in the NRC features tend to be ones containing a lot of zero values.

userId	WC	WPS	Sixltr	Dic	Numerals	funct	pronoun	ppron	i	...	Colon	SemiC	QMark	Exclam
3360a7f	18	18.00	44.44	55.56	0.00	27.78	0.00	0.00	0.00	...	22.22	0.00	3661.11	0.00
499d8e	45	22.50	26.67	84.44	0.00	62.22	22.22	13.33	13.33	...	0.00	0.00	0.00	0.00
17d8c5	186	26.57	9.68	84.41	1.08	51.08	27.42	26.34	12.37	...	3.23	0.00	2.15	5.38
cd6a44	157	22.43	20.38	83.44	0.64	29.94	10.19	6.37	2.55	...	3.82	0.64	1.91	8.92
ff05326	125	3.91	25.60	56.00	0.00	27.20	7.20	6.40	2.40	...	5.60	0.00	1.60	6.40

Figure 5: Examples of outliers for LIWC features.

userId	positive	negative	anger	anticipation	disgust	fear	joy	sadness	surprise	trust
17f376b	0.875000	0.125000	0.019231	0.134615	0.019231	0.019231	0.384615	0.019231	0.057692	0.346154
1461132	0.600000	0.400000	0.000000	0.000000	0.000000	0.200000	0.400000	0.000000	0.400000	0.000000
550a0cf	0.371429	0.628571	0.194805	0.038961	0.025974	0.194805	0.077922	0.207792	0.207792	0.051948
9e236e	0.666667	0.333333	0.000000	0.000000	0.000000	0.000000	0.500000	0.000000	0.000000	0.500000
1c7acfcf	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 6: Examples of outliers for NRC features.

In our best performing prediction models using text features, we used both LIWC and NRC features merged together to predict age and personality, and did not remove any outliers.

3.3 Image features data

As mentioned previously, the oxford features represented relevant facial point coordinates and facial characteristic measures, such as facial hair and the area containing the face.

We experimented with feature engineering by trying to find more expressive features, such as the euclidean distance between two facial analogous points. Unfortunately, we found that this prepossessing method did not improve our gender prediction and was not included in our final model. We did however notice that some users had multiple face ID's, probably indicating that more than one face was present in a profile picture. Thus, instead of randomly selecting a face ID, we proceeded to create a new feature expressing the face rectangle area and we selected the face ID that had the greatest area. This step increased our validation accuracy and was added to our final model for gender prediction.

Furthermore, we obtained our final gender prediction results by removing outliers using again *IsolationForest* [6]. The contamination parameter determines how many outliers are to be removed and a value of 0.10 was selected (meaning 10% of the training set, which consists technically of outliers, is removed).

Lastly, we also investigated on oxford features by trying different feature selection techniques. Some of what was tried include evaluating Chi-squared scores, feature importance and forward feature selection using gradient boosting decision trees (results of these can be found in the appendix A). Since gradient boosting decision trees already implicitly consider important features (on the top level splits), we did not use a subset of features prior to training our gender predictor. We did however obtain interesting and coherent results for all these methods. Indeed, facial hair features, such as mustache, beard and sideburns, were either always selected or had the highest scores for the purpose of gender classification.

3.4 Relational data

The relational data consisted of userids and pageids pairs. Such a pair indicated that a user of id userid liked the page of id pageid.

This data structure can not be directly digested by a machine learning algorithm. Therefore, we had to do perform feature engineering before feeding it to one. The following two sub-sections outline the two methods we experimented with to create new datasets from the original data.

3.4.1 Manual feature engineering. We created two features:

(1) Standardized number of likes a user gave.
(2) Standardized popularity of the pages that the user liked.

To construct the first feature, we counted how many pageids were associated with the given userid. We then standardized the values across all users so that the distribution of values would have zero mean and a unit variance. Standardization helps models learn.[5]

To construct the second feature, we started by calculating how many likes each page received in general from all users. Then, for each pageid that was associated with a userid, we summed those pages' total likes and based this value as the second feature. We standardized this feature as well like we did with the first.

We refer to this feature engineering in our code and this document as relation v1.

When we visualised these features against the properties we wanted to predict: age, gender and personality, we saw very similar graphs regardless of whether we plotted as a function of the first or the second feature. Due to this, let's only take some example as functions of the first feature:

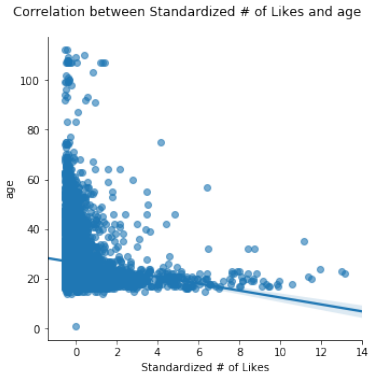


Figure 7: Age as a function of standardized # of likes

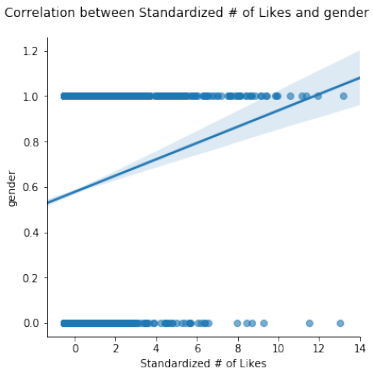


Figure 8: Gender as a function of standardized # of likes

Given these visualisations, it seems these two features we engineered might not be as good as we thought. Perhaps some learnings could be extracted out of these features for the age prediction task, but we were not so hopeful for the other two tasks.

Given this discovery, we also experimented with more advanced methods of creating features out of the relational data in the hope they could give us more promising results which we will discuss now.

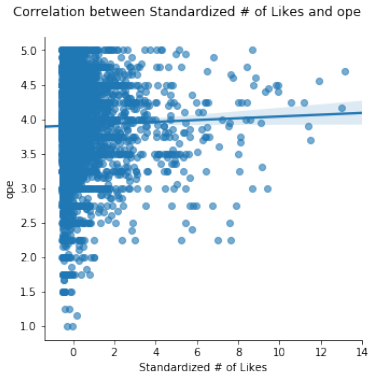


Figure 9: Openness as a function of standardized # of likes

3.4.2 Automatic embeddings engineering. Instead of thinking of the relational data as a collection of pairs, we figured we could instead interpret these pairs as a bipartite graph. Each node would either be a user or a page. Each user is linked to one page or more and vice versa. Using this interpretation, we investigated methods that could create embeddings from graphs, in a smarter and more thorough manner than we did with the manual feature engineering.

In particular, we experimented with DeepWalk[3] and Node2Vec[1]. Both of these methods strive to represent each node of a provided graph as an embedding. They do this by performing a predetermined amount of random walks of certain lengths from each node to other nodes through one of the nodes' links. Each walk generates a sequence of nodes visited which in turn will be treated as the equivalent of sentences. Each walk that originates from the same node will be used to encode information into the embedding of said node based on that node's context.

Node2Vec is an improvement to DeepWalk which is based on Word2Vec's skipgram model.[7] The skipgram model is what consumes the random walks as sentences and tries to generate similar embeddings for nodes that appear in a similar context. An example of the hyper-parameters found in Node2Vec and DeepWalk: how many random walks are performed from each node, how long each random walk should be, of how many dimensions the resulting embeddings vectors should be and what the window size should be. What follows are the particularities of the two methods.

DeepWalk does not provide any control on the nature of the random walks. For example, it will always be as likely to go from a node to another. On the other hand, Node2Vec provides us with two hyper-parameters to control the random walk: P and Q. Let's say we already went from node A to B. P controls the probability that we will go back to A in our next step while Q controls the probability we will go to a previously undiscovered part of the graph.

In the Results section, we will discuss results when training models using the features and embeddings constructed manually and automatically.

3.5 Model Selection and Evaluation Metrics

3.5.1 Cross Validation. We performed 5-fold cross validation to evaluate our algorithms and choose the best hyperparameters. All our final models used the gradient boosting approach, and we selected the three most important hyperparameters that would influence the outcome – `n_estimators`, `max_depth` and the `learning_rate`. `n_estimators` represents the number of iterations performed during the training phase (or equivalently, the number of trees used). `max_depth` is the maximum depth of a tree during each iteration. As you increase the number of estimators and max depth, you increase the capacity of your model. In order to prevent overfitting, we also tuned the `learning_rate` which in this particular case acts as a regularization hyperparameter; the higher the value the more complex hypotheses are penalized.

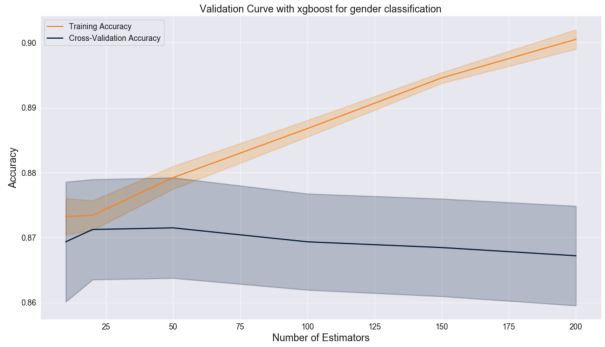


Figure 10: 5-fold cross validation on the number of estimators using gender prediction as an example.

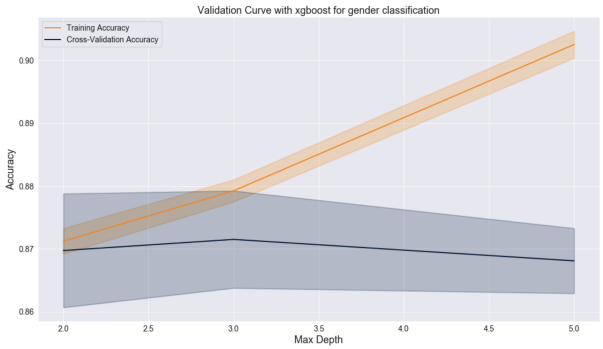


Figure 11: 5-fold cross validation on the maximum depth using gender prediction as an example.

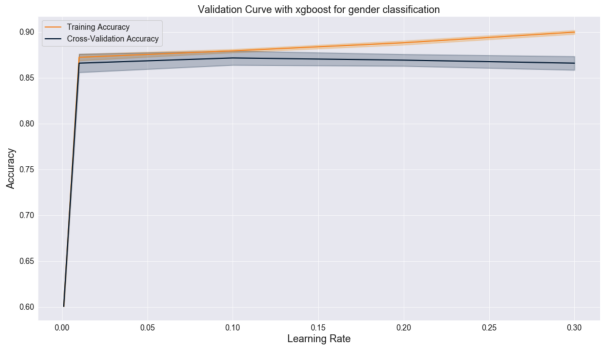


Figure 12: 5-fold cross validation on the learning rate using gender prediction as an example.

3.5.2 Evaluation Metrics. For the two classification tasks, we used the *accuracy* (fraction of the correct predictions) and for the regression tasks we used the root mean squared error as evaluation metrics.

4 RESULTS

What follows are results using the various machine learning methods identified in the Methodology section and the datasets in the Datasets and metrics section [1]

4.1 Dataset splits

In order to get a robust understanding of the strengths and weaknessses of the algorithms, we used the following data splits:

- Gradient boosting decision trees
  - Training: 70%, validation: 30%.
- Relational data models
  - Training: 80%, validation: 20%.

4.2 Gender Classification

4.2.1 Gradient boosting decision trees. Using our best models settings (selected using *cross-validation* as described in the previous section), we evaluated the performance on the held out validation set. We obtained an accuracy score of 87.53%.

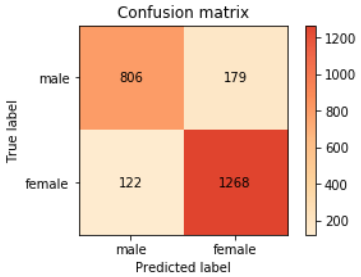


Figure 13: Confusion Matrix for Gender Prediction

[1] We did not perform a detailed analysis of models that didn't beat the baseline on one of the predictions tasks. We simply specify the performance in the tables below.

Model	Accuracy
baseline	59.10 %
gradient boosting + feature engineering	**89.14** %
gradient boosting decision trees	87.03 %
relation data v1 + vanilla nn	59.05 %

Table 1: Our various gender models and their accuracies

We observe that we confused more females for males (179 instances) than males for females (122 instances). The model predicts 1268 instances of females correctly as opposed to 806 male instances; this is because our dataset is imbalanced and includes more female examples as seen in Figure 3.

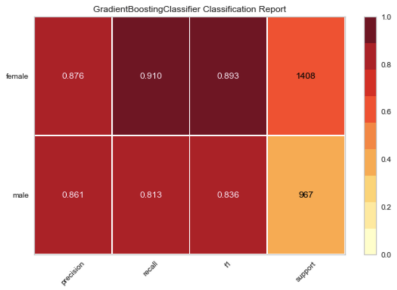


Figure 14: Classification Report for Gender Prediction

We see that the recall for males is considerably lower than for females. Again, this is a direct consequence of the class imbalance in the dataset as seen in Figure 3.

4.3 Age Classification

Using our best models settings (selected using *cross-validation* as described in the previous section), we evaluated the performance on the held out validation set. We obtained an accuracy score of 59.57%.

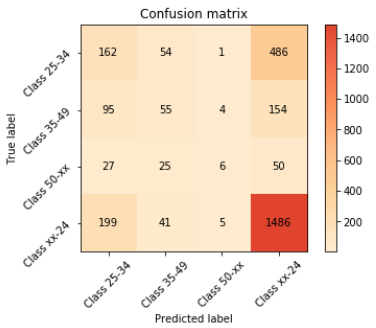


Figure 15: Confusion Matrix for Age Prediction

We observe that classes 25-34 and 25-34 are often confused. This could be because since they are neighbouring classes, there are not enough discriminatory features to distinguish the two classes. In a real world setting, we could take the following steps to further improve the performance:

Model	Accuracy
baseline	59.4 %
gradient boosting decision trees	**60.5** %
relation data v1 + vanilla nn	58.2 %
relation data + deepwalk + gradient boosting	59.4 % *
relation data + deepwalk + random forest	59.4 % *
relation data + node2vec + gradient boosting	59.4 % *
relation data + node2vec + random forest	59.4 % *

* Interestingly, regardless of which combination we used of deepwalk, node2vec, gradient boosting decision trees or random forests, we didn't manage to score anything higher than the baseline. We double checked our models, ran several different runs using different hyper-parameters for all four items without luck. We made sure to gather embeddings from all examples and then split the resulting embeddings into a training and a validation set to no avail.

Table 2: Our various age models and their accuracies

- Engineer features or collect more features that help us discriminate between the two classes.
- Depending on the use case, we could also benefit from merging the two groups.

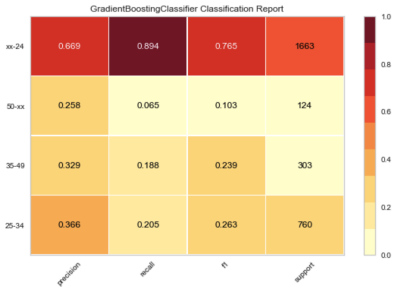


Figure 16: Classification report for Gender Prediction

We observe that the recall for the minority classes 25-34, 35-39 and 50-xx is quite low (<25%). For class 50-xx in particular we have a recall rate of almost 0%. This means we essentially failed to correctly identify instances from this class. The primary reason for the low recall rates as before is the class imbalance in the dataset as seen in Figure 2. The precision is also relatively low, even for the majority class of xx-24. This means that of the instances that were reported as being from xx-24, about 66% of them were correct and this implies a weak quality classifier. It hints to the fact that we do not have enough predictive power in the explanatory variables.

4.4 Personality Regression

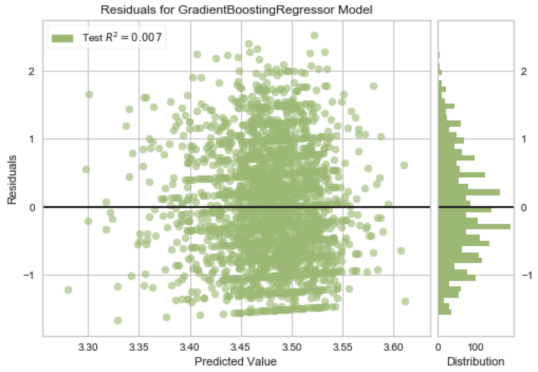


Figure 17: Residual Plot for Extroversion Regression

We observe that there is no obvious pattern in the residuals. They appear to be randomly dispersed. The R^2 value (0.007) is also quite low, implying that there is no linear pattern in our data. We also observe that the residuals are roughly normally distributed, indicating a well fitted model. We performed similar analyses on the other regression tasks and observed no discernible patterns.

Model	OPN	NEU	EXT	AGR	CON
baseline	0.652	0.798	0.788	0.665	0.734
gradient boosting decision trees	0.648	**0.796**	**0.786**	**0.662**	**0.726**
relation data v1 + vanilla nn	**0.634**	0.798	0.797	**0.662**	0.727

Table 3: Our various personality models and their error (RMSE)

5 CONCLUSION AND FUTURE WORK

In all three prediction tasks, the best performing model was gradient boosting decision trees. Moreover, the age multi-class classification task performed better with text features (LIWC and NRC features merged together) to make its predictions, as did the personality regression task. The gender binary classification task however performed better with image features (Oxford features). In future work, the SMOTE (Synthetic Minority Over-Sampling) technique [4] could be used to generate synthetic samples for the minority age and gender classes as seen in Figures 2 and 3. This technique finds the $k-$nearest neighbours and generate samples similar to them in order to combat the consequences of class imbalance.

In a time in which data privacy has become an important topic, this project points out the fact that valuable information such as age, gender, and personality can be inferred from someone's Facebook user profile, even if they do not explicitly share this information. This information is valuable to companies that build applications that rely on personalisation, such as recommender systems, but bring up important ethical issues related to user privacy and the disclosure (or lack thereof) of third party usage of Facebook user profile data.

REFERENCES

[1] Jure Leskovec Aditya Grover. 2016. *node2vec: Scalable Feature Learning for Networks.* Retrieved December 17th 2019 from https://arxiv.org/abs/1607.00653
[2] Jason Brownlee. 2016. *A Gentle Introduction to XGBoost for Applied Machine Learning.* Retrieved December 17th 2019 from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
[3] Steven Skiena Bryan Perozzi, Rami Al-Rfou. 2014. *DeepWalk: Online Learning of Social Representations.* Retrieved December 17th 2019 from https://arxiv.org/abs/1403.6652
[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (2002), 321–357.
[5] Swetha Lakshmanan. 2019. *How, When and Why Should You Normalize / Standardize / Rescale Your Data?* Retrieved December 17th 2019 from https://medium.com/@swethalakshmanan14/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff
[6] Eryk Lewinson. 2018. *Outlier Detection with Isolation Forest.* Retrieved December 17th 2019 from https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e
[7] Chris McCormick. 2016. *Word2Vec Tutorial - The Skip-Gram Model.* Retrieved December 18th 2019 from http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
[8] Saif M. Mohammad and Svetlana Kiritchenko. 2013. Using Nuances of Emotion to Identify Personality. *CoRR* abs/1309.6352 (2013). arXiv:1309.6352 http://arxiv.org/abs/1309.6352
[9] Boyd R.L.-Jordan K. Blackburn K. Pennebaker, J.W. 2015. The Development and Psychometric Properties of LIWC2015. *The University of Texas at Austin* (2015).

A APPENDIX: FEATURE SELECTION RESULTS

	Feature	Score
59	facialHair_beard	790.315948
58	facialHair_mustache	712.772793
60	facialHair_sideburns	577.982523
0	faceRectangle_width	13.883665
1	faceRectangle_height	12.763303
2	faceRectangle_left	2.487902
3	faceRectangle_top	2.374354
29	eyebrowRightOuter_y	2.144854
15	eyebrowLeftOuter_y	1.719007
14	eyebrowLeftOuter_x	1.695036

Figure 18: SelectKBest with Gradient Boosting Decision Trees

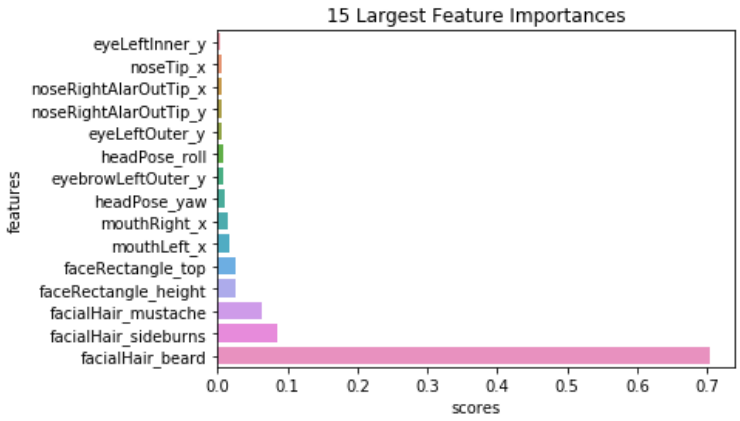


Figure 19: Feature Importance with Gradient Boosting Decision Trees

Facebook User Predictions Code Documentation

Aditya Joshi, Carolyne Pelletier, Helgi Tómas Gíslason, Lawrence Abdulnour
Computer Science, Université de Montréal, Montreal, Quebec

Abstract

What follows is a code documentation about the solution we submitted for the class IFT6758 - Data Science, Fall 2019. First, we present the code structure in Section 1 followed by a UML diagram of the code structure in Section 2.

1 Project structure

The entry point to our software is the bash script ift6758. The Python equivalent of that script is ift6758.py. You can use the Python script as your entry point to explore our code.

The solution is setup with the following structure:

- README.md
  - Please read this file to know how to work with our software. It contains instructions on how to:
    - Install dependencies.
    - Obtain predictions using our provided pre-trained models.
    - Creating new models by training our estimators.
    - Obtain predictions using those new models.
- environment.yaml
  - Please read README.md on how to use this file.
  - Contains all the conda dependencies needed to run our software.
- ift6758
  - A bash script one can use to obtain pre-trained predictions. Arguments:
  - *-i* or *–input_path*
    - Specifies what the path to the new_data should be.
  - *-o* or *–output_path*
    - Specifies what the outputted predictions path should be.
- ift6758.py
  - The Python script equivalent of the bash script mentioned above. Accepts the same arguments.
- datasets/
  - generate_synthetic_data.py
    - We used this script initially to generate some fake data that we could then feed into our algorithms to check if our code compiled and ran properly. Generates the folder /synthetic with the fake data.
- notebooks/
  - This folder contains most of the notebooks we used to explore our data.
- src/

  - This is our main code folder. It contains the following items:
  - predict.py
    * The Python script that we use to obtain predictions. Arguments:
    * *-i* or *–input_path*
      · Specifies what the path to the new_data should be.
    * *-o* or *–output_path*
      · Specifies what the outputted predictions path should be.
    * *-m* or *–model_path*
      · Specifies what model should be used to obtain predictions with.
  - train.py
    * The Python script that we use to train our estimators. Arguments:
    * *–data_path*
      · Specifies what the path to the new_data should be.
    * *–save_path*
      · Specifies what the outputted predictions path should be.
    * *–age_estimator*
      · Specifies which age_estimator we should use to train. See available options in the age_estimators variable in src/train.py.
    * *–gender_estimator*
      · Specifies which gender_estimator we should use to train. See available options in the gender_estimators variable in src/train.py.
    * *–personality_estimator*
      · Specifies which personality_estimator we should use to train. See available options in the personality_estimators variable in src/train.py.
  - config/
    * Contains hyper-parameters for the CNN model we experimented with.
  - constants/
    * Contains files with Python constants used throughout our code base.
  - data/
    * Contains files that we used to perform data transformation and serialization before feeding it to our algorithms.
  - estimators/
    * Contains all of the estimators that we wrote to solve the predictions tasks. Each estimator either derives from the base/age_estimator.py, base/gender_estimator.py or base/personality_estimator.py and is located in the corresponding subfolder: age, gender or personality.
    * estimator_utils.py contains some helper functions that we used for the DeepWalk, Node2Vec and Random Forest experiments.
    * fb_user_estimator.py keeps track of the three chosen estimators of each task, trains them and serves predictions.
  - evaluation/
    * Contains items used for evalutions.
  - networks/
    * Contains basic neural networks and convolutional neural networks we experimented with.
  - util/
    * Contains utilities for transforming the image, oxford features
    * Contains some helper functions for the global project
- synthetic/
  - This is a folder generated when code from generate_synthetic_data.py in datasets/
- test/
  - Uses the synthetic data in synthetic/ to check if our code compiled and ran properly.

2 UML Diagram

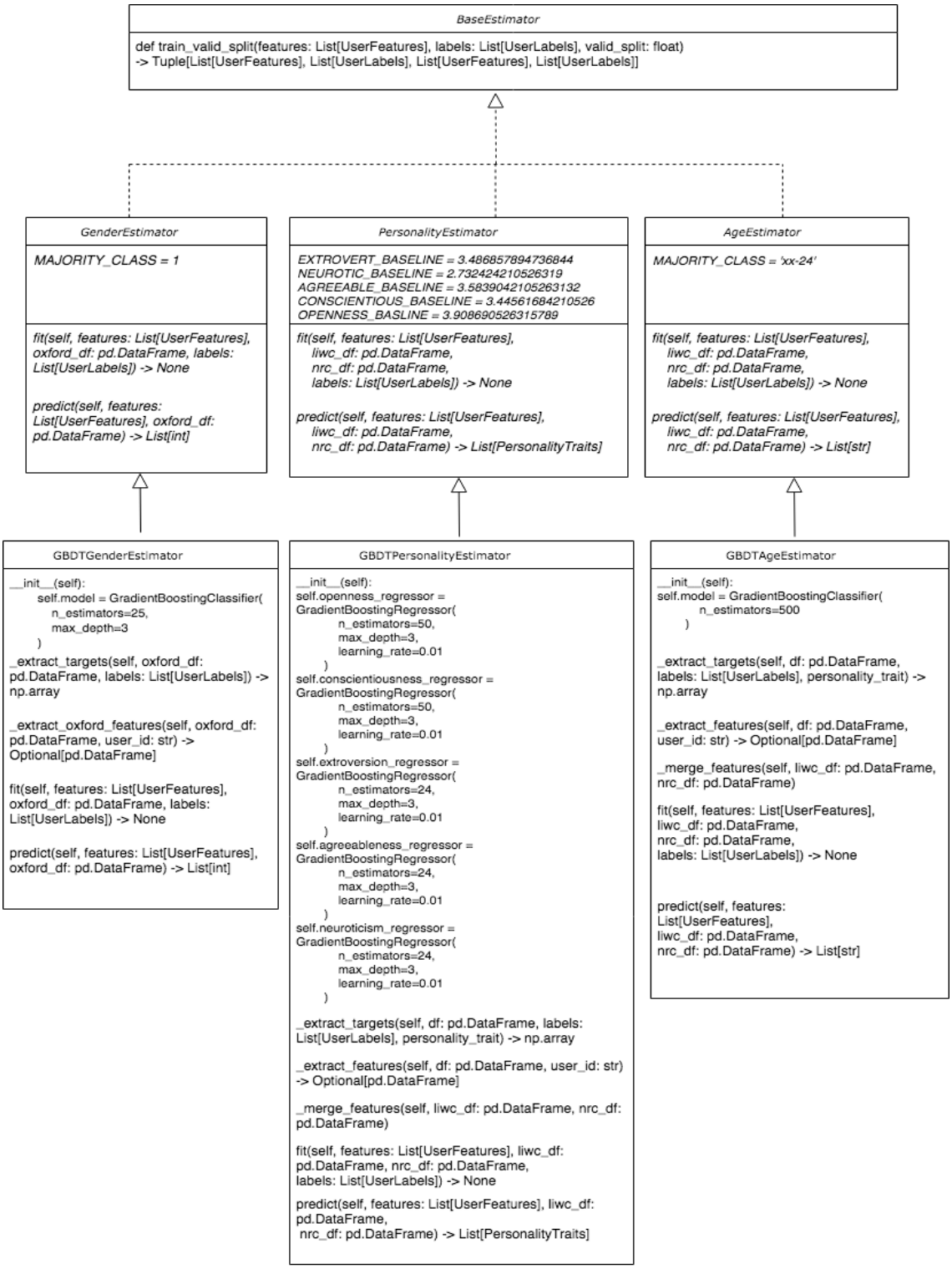


Figure 1: UML Diagram