Man Vs The Digital Oracle

Mike Doan
Computer Science Department
Georgia State University
Atlanta, United States
mdoan4@student.gsu.edu

John Ericson
Computer Science Department
Georgia State University
Atlanta, United States
jericson@student.gsu.edu

Robert Tognoni
Computer Science Department
Georgia State University
Atlanta, United States
rtognoni1@student.gsu.edu

***Abstract*— This paper covers an assessment of twelve machine learning models using an OkCupid dataset of dating profiles created in 2015. This study aims to evaluate the performance of these models and find which is most effective at classifying a user's 'Star Sign Intensity'. 'Star Sign Intensity' is a composite feature based on self-reported OkCupid survey data representing one's affinity or interest in their zodiac sign. The performance of each model was evaluated based on metrics including accuracy, precision, recall, F1 score, and associated learning curves. Model selection, for each algorithm, compared three independent train/test splits (50-50, 70-30, and 80-20) before undergoing 10-fold cross-validation. The results of which were compared and the best models (by metrics) for each were selected by hand. The findings of this study do not necessarily support much in the way of predicting a human's interest in star signs in the given context but do supply valuable insights into the appropriate selection of machine learning models and algorithms for any application.**

Keywords*— *Astrology, Online Dating, Astrological Sign (AS), Star Sign Intensity (SSI), OkCupid (OKC), Machine Learning (ML)

I. Introduction & Project Description

In the past, analysis of the OkCupid (OKC) dataset has been used to study numerous interesting phenomena. Some are potentially unethical, such as finding the sexuality or race of an individual, etc. This project instead proposes something else: Is it possible to use common Machine Learning algorithms to classify a person's Astrological Sign (AS) based on their self-reported dating data (i.e., how does an individual's projected dating-ready self-compare to their sign)? In more educated circles (of humanity circa 2023), it is a given that there is no possible correlation between any aspect of a person and AS beyond mere coincidence [1]. Centuries of scientific progress and understanding have de-mystified the heavens and pushed humanity into an age of reason and proper discourse. From this perspective, it would not make sense to assume anything about personality based on one's birth month. Despite this, it is common practice in modern society to make decisions or insights on personality traits based on such factors [2]. Given that one in four people believe in 'mystical' astrology it is worth looking into real-life associations between personality traits and star signs [3]. Initial attempts to classify a user's AS were inconclusive as a result it was decided that more information could be gained by classifying based on a user's Star Sign Intensity (SSI) instead. This metric is based on more information provided in the survey that describes an individual's stated interest in astrology. For example, a user may report that their star sign matters a lot to them, it is fun to talk about, or holds no significance. This extracted field was simplified to a binary class of two possibilities, interest and lack of interest in astrology.

Analysis of the OkCupid dataset will be performed using traditional Machine Learning models to classify an individual's Star Sign Intensity based on their self-reported data. The project intends to compare a person's projected (reported) self to their Star Sign Intensity to decide if any correlations can be made e.g., income or marital status. Given the interesting but illogical existence of Astrology in the modern-day discourse, we have chosen to compare our base assumptions to the results from modern Machine Learning algorithms on this dataset.

II. Survey Of Related Work

The following list includes a variety of prior experiments, articles and scientific papers that are related to the OKC dataset or subject of Astrology in modern society.

[1] B. Resnick, "Researchers just released profile data on 70,000 OkCupid users without permission" Vox, May 12, 2016. [Online]. Available: https://www.vox.com/2016/5/12/11666116/70000-okcupid-users-data-release.

[2] E. Kirkegaard, "Intelligence and religiosity among dating site users" ResearchGate, December 2019. [Online]. Available: https://www.researchgate.net/publication/338125762_Intelligence_and_Religiosity_among_Dating_Site_Users.

[3] G. Suarez-Tangil, M. Edwards, C. Peersman, G. Stringhini, A. Rashid, M. Whitty, "Automatically Dismantling Online Dating Fraud", *2020 IEEE Transactions on Information and Security*

[4] C. van der Lee, T. van der Zanden, E. Krahmer, M. Mos, and A. Schouten, "Automatic identification of writers' intentions: Comparing different methods for predicting relationship goals in online dating profile texts" in *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, Nov. 2019, pp. 94–100. doi: 10.18653/v1/D19-5512.

[5] D. Boller, M. Lechner, and G. Okasa, *The Effect of Sport in Online Dating: Evidence from Causal Machine Learning.* 2021.

[6] I. Backus, "Predicting Gender from OkCupid profiles using ensemble methods" self-published, Mar. 22, 2018. [Online]. Available: https://raw.githubusercontent.com/wiki/ibackus/okcupid-gender-prediction/okcupid_gender_prediction.pdf.

[7] M. Campbell, "Investigating OkCupid profiles: Can we predict someone's city?" Medium, July 21, 2022. [Online]. Available: https://medium.com/@macrotentional/investigating-okcupid-profiles-can-we-predict-someones-city-31a4734e96dd.

III. Data Source

A. Background

Investigation of the infamous OkCupid dataset has been ongoing since its release in 2015. When made publicly available the release was considered somewhat controversial as it contained personal information of users who did not intentionally consent to the release. However, all information in this dataset is completely open to the public on the site and has no names, logins, birth dates, or other directly identifying features.

This project is based on a formatted version of the dataset obtained from kaggle.com holding ~60,000 entries and 31 attribute columns. After cleaning and filtering results down to users with a reported SSI, 14,506 entries remained for assessment.

Some entries have up to ten user-provided ‘essays’ holding personal thoughts and quotes. Since users selected prompts in any order and could enter any text, it was not possible to use the essays directly. Instead, these fields were combined into a simple metric of word length as a user’s typed word volume could be relevant to their SSI.

Some cases of nonresponse in the dataset can be considered valid data, as the act of withholding information for any reason could be similarly relevant to a description of an individual's personality as an honest response. However, for certain fields, this information was determined to be too vague to be interpreted and was discarded to reflect this.

B. Technical Description

a. Headers: age, status, sex, orientation, body_type, diet, drinks, drugs, education, ethnicity, height, income, job, last_online, location, offspring, pets, religion, sign, smokes, speaks, essay0, essay1, essay2, essay3, essay4, essay5, essay6, essay7, essay8, essay9
b. 59,949 raw entries
c. .csv format

IV. Key Algorithms & Technology

A. Algorithms

We applied the following Machine Learning algorithms to the OkCupid classification problem.

- Decision Tree
- Support Vector Machine (Linear Kernel)
- Support Vector Machine (RBF Kernel)
- Perceptron
- Multi-layer Perceptron
- Naive Bayes Classifier
- Logistic Regression
- Linear Regression
- Gradient Boosting
- Linear Ridge Regression
- K-Nearest Neighbors
- Passive Aggressive Classifier

B. Project Organization

- Google Colab was the primary organizational technology used for writing, training, and evaluating all algorithms and their associated code. A single Python .ipynb notebook has been be worked on collaboratively by all members of the group.
- A GitHub repository has been be created to hold the final notebook and all associated files.
- Google Docs, Slides and Sheets were used as needed to share data, writeups and create keynote presentations.

V. Project Execution

A. Preprocessing

All models used either partially or fully require all data points to be in numeric form, therefore all columns were converted into numeric data wherever applicable. Our assessment of the features in the data are as follows:

- ‘age’ and ‘income’ were already in numeric form and did not need any conversion.
- ‘height’: while ‘height’ was already in numeric form, it was decided that an additional column should be created that contains height adjusted for the difference in biological sex titled ‘height_data’.
- ‘sign’: originally holding both a user’s star sign and their attitude towards astrology, this was split into ‘sign_data’, which contains users’ astrological sign sorted by associated birthdate in the year, as well as sign_intensity, which is the label used for all our predictions.
- ‘religion_data’ was converted into numbers according to the index of the members in the following list: ['Catholicism', 'Christianity', 'Hinduism', 'Islam', 'Judaism', 'Buddhism', 'other', 'none', 'Agnosticism',

'Atheism',] sorted based on similarity of belief and a split between theists and atheists.

- 'religion_intensity' is sorted based on how serious the user's attitude towards their religion is: 'and laughing about it', 'and somewhat serious about it', 'but not too serious about it', 'and very serious about it']
- 'status_data' is sorted from most available to least available.
- 'sex_data' is encoded to 0 for male and 1 for female.
- 'height_data' is an auxiliary column that subtracts the height difference between the sexes in the state of California where most of our data points came from.
- 'orientation' is encoded as 0 for 'straight', 1 for 'bisexual', and 2 for 'gay'.
- 'body_type' sorts body types based roughly on body fat percentage into a numerical value ranging [0-9]
- 'diet_data' was converted into the index of their alphabetically sorted labels.
- 'drinks_data', 'smokes_data' and 'drugs_data' were all converted based on frequency of consumption.
- 'education_data' was sorted based on the general level of education reported by the user, ranging from those currently in primary and secondary education to those with a PhD.
- 'job_data' was converted into the index of their alphabetically sorted labels.
- 'last_online_data' was converted into the number of minutes since last online.
- 'speaks_data' was sorted and encoded based on whether someone was monolingual, bilingual, or a polyglot (3 languages or more)
- essays 0 through 9 were converted into the length of their essays, and 'essay_len' is the total length of all essays on the user's profile.

B. Feature selection

For training and predicting, all the original numeric data points as well as all converted numeric data points was used: *'age', 'height', 'income', 'sign_data', 'religion_data', 'religion_intensity', 'status_data', 'sex_data', 'height_data', 'orientation_data', 'body_type_data', 'diet_data' 'drinks_data', 'drugs_data', 'education_data', 'job_data', 'last_online_data', 'offspring_data', 'smokes_data', 'speaks_data', 'essay0_data', 'essay1_data', 'essay2_data', 'essay3_data', 'essay4_data', 'essay5_data', 'essay6_data', 'essay7_data', 'essay8_data', 'essay9_data', 'essay_len'.*

The column used for prediction purposes was 'sign_intensity'.

C. Pipeline

- Model initialization: All models were initialized with random_state = 1234 for reproducibility whenever possible.
- Data splits: The models were trained on three splits of the dataset in three ratios (50-50, 70-30, 80-20)
- Cross validation: For every split, every model is trained using 10-fold cross validation of the training set, from which the best model was selected.
- For classification models, we primarily used accuracy as the determining metric as our dataset is largely evenly split (47-53).
- From the best models chosen for every split, we choose the best model for every model type.
- Finally, from all the models we choose the single best performer.
- Notes on model evaluation: models that perform close to or worse than 0.53 (always guessing a single class) will be classified as poorly performing.

D. Notes on linear regression models

Linear regression models are typically used for predicting continuous numerical values, while predicting users' belief in star signs involves predicting categorical values. Therefore, using a linear regression model was not be appropriate, since the output of the model would not be limited to the range of possible categorical values.

VI. Model Evaluation

A. Decision Tree:

Overview: The decision tree model is a supervised learning classifier used to predict a target classification variable from numerical and categorical data. Using the ID3 (greedy heuristic) the decision tree is constructed to create the simplest possible tree (lowest entropy) from the provided features. The benefit of using a Decision Tree comes from its simplicity and the ability to visualize the model. Potential downsides include their inability to generalize well and their relative instability.

Overall results: Decision tree managed to fit on our dataset, given a small enough choice of max tree depth (the models we used had a max depth of 5), with minimal overfitting.

Best performers: Models trained on fold 0 of split 0, fold 5 on split 1 and fold 8 on split 2 performed the best with accuracy scores of 0.603, 0.634 and 0.628 on the validation set respectively. Of these three models, Model 8 trained on split 2 performed the best on the test set, with an accuracy of 0.62. Because there was only an exceedingly small drop of 1% between the cross validation and test evaluation, it can be concluded that the decision tree generalized well on our dataset.

Notes: The choice of max_depth = 5 was picked through experimentation, as any depth smaller than 5 did not have enough capacity to fit to the dataset, and any depth significantly bigger than 5 (for example 10 or 20), highly overfit and resulted in non-performant models.

B. *Perceptron:*

Overview: The perceptron is a supervised learning model designed to predict binary classifications of data. It is generally considered the simplest form of neural network. Using a linear classifier algorithm, it updates the weights of a multidimensional hyperplane to best fit (classify) the data. The benefit of the perceptron is its simplicity and scalability. The training cost is relatively low and as more data is added the model becomes more stable. Potential pitfalls of the perceptron include its poor generalization and inability to classify non-linearly separable data.

Overall results: Using the full dataset with all available columns, the perceptron does not fit to the data. The cause is most likely the pitfall of non-linearly separable data, as some of the columns were likely contradictory (data points with the same label might have opposite metrics on the same column/the same data point along multiple columns may have different labels). This resulted in models that essentially draw the decision boundary at random, aside from preferentially preferring the class with the slight majority in the dataset.

Best performers: Models trained on fold 7 of split 0, fold 5 of split 1 and fold 4 of split 2 were the best performers. Of these three models, model 7 trained on split 0 was the best performer, but this was largely due to a lucky random decision boundary and not a result of the hyperparameters chosen, or the specific split used to train the data. Because Perceptron performed poorly on both the training and validating data, it can be concluded that it did not generalize.

Notes: setting fit_intercept to False resulted in slightly better performing models, but overall, there was little to be done with a perceptron with non-linearly separable data.

C. *Naive Bayes:*

Overview: The Naive Bayes model is a supervised learning model that applies Bayes theorem to classify data. Naively using the assumption that all data are independent of one another, the algorithm is able to use conditional probability to predict the posterior probability of a predicted class: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, where P(A|B) is the posterior probability of event A given event B, P(A) is the probability of event A and P(B) is the probability of event B.

Naive Bayes does well in classification schemes due to its base assumption being typically true in most machine learning applications. If we know how the features of a dataset are dependent on each other, we likely don't need machine learning. This also allows Naive Bayes to be relatively fast compared to other algorithms, which makes it a good first choice to probe a dataset.

This same generalization can be viewed as a downside to the model, as it oversimplifies real-world classifications and results in a less accurate model than larger more complicated classifiers.

Overall Results: Naive Bayes was a poor performer when compared to other algorithms, though it did produce improved metrics compared to the Perceptron and Multilayer Perceptron.

Best Performers: Models trained on fold 2 of split 0, fold 3 of split 1, fold 2 of split 2 were the best performers, with accuracy scores on the validation set of 0.534, 0.578, and 0.564 respectively. Of these, the best performer on the test set was model 3 on split 1 with an accuracy score of 0.55. Because of the minimal reduction accuracy between the validation and test set, it can be concluded that the naïve Bayes classifier did generalize to the dataset.

D. *Linear Regression:*

Overview: The Linear Regression model is a supervised linear classifier that uses a least squares calculation to develop the model. This model is a continuous predictor model and does not classify, but rather predicts a value on a continuous spectrum.

Overall results: Because our prediction is a discrete classification, the accuracy, precision, and recall of the Linear Regression model is incoherent. As a result, linear regression metrics will be evaluated purely off RMSE.

Best performers: Models trained on fold 5 of split 0, fold 4 of split 1 and fold 6 of split 2 performed the best, with RMSE of 0.482, 0.484 and 0.481 respectively.

E. *Logistic Regression:*

Overview: Logistic regression is a classification model used to analyze the relationship between a dependent variable and one or more independent variables. It uses a logistic function to output a probability value between 0 and 1, which stands for the probability of the dependent variable being in one of two categories. The model estimates the parameters of the logistic function using a maximum likelihood estimation method, and these estimates are used to predict the probability of the dependent variable for new observations based on the input variables. It is commonly used in data analysis and machine learning applications for binary classification problems.

Overall results: Logistic regression performed slightly above average compared to other models in the set, being overcome by the likes of Decision Tree and Gradient Boosting. However, with a weighted average of 0.60 makes it stand out above models with averages closer to 0.55. As stated above, Logistic Regression is known to perform well with binary classification problems, which is represented in these results.

Best performers: Models trained on fold 0 of split 0, fold 4 of split 1, fold 8 of split 2 were the best performers, with accuracy scores on the validation set of 0.618, 0.621, and 0.613 respectively. Of these, the best performer on the test set was model 4 on split 1 with an accuracy score of 0.60. Because of the minimal reduction accuracy between the validation and test set, it can be concluded that the logistic regression classifier did generalize to the dataset.

Notes: An assay of the six available solvers was conducted for linear regression. These options were compared on the same tenfold cross validation across the same splits as the overall models in this project. Of lbfgs, liblinear, Newton-Cholesky, Newton-cg, sag, and saga options, Newton-Cholesky improved all scores by a marginal 0.0001. Overall, this difference is not significant, but the sag and saga solvers had lower F1 scores by around 0.01. In the end the Newton-Cholesky solver was used as it did produce a larger value. The official Scikit-learn documentation for the solver notes that it "is a good choice for n_samples >> n_features" [4], which is the case with this dataset.

F. *SVM with Linear Kernel*

Overview: Support Vector Machine (SVM) is a supervised learning algorithm that can be used for classification and regression analysis. SVM with a linear kernel is a variant of the SVM algorithm that uses a linear function to separate the data into different classes. This means that it is suitable for binary classification problems where the classes can be separated by a straight line or hyperplane. The SVM algorithm aims to find the hyperplane that maximizes the margin between the two classes, which is the distance between the hyperplane and the closest data points from each class. The linear kernel in SVM is used to calculate the dot product of the input data, which helps to transform the data into a higher dimensional space where it can be separated by a hyperplane. However, it may not perform well on datasets that are not linearly separable.

Overall results: SVM with Linear Kernel performed similarly to Logistic Regression with a modest score disadvantage. Overall, it is among the top performers of the group, but is not a standout like Decision Tree or Gradient Boosting.

Best performers: Models trained on fold 0 of split 0, fold 4 of split 1, fold 5 of split 2 were the best performers, with accuracy scores on the validation set of 0.612, 0.623, and 0.605 respectively. Of these, the best performer on the test set was model 4 on split 1 with an accuracy score of 0.59. Because of the minimal reduction accuracy between the validation and test set, it can be concluded that the SVM with linear kernel classifier did generalize to the dataset.

Notes: The LinearSVC implementation from Scikit-learn was used over the standard SVC model with parameter "kernel=linear". LinearSVC is "…like SVC with parameter kernel='linear'...it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples" [4]. This makes it potentially less comparable to SVM with RBF Kernel referenced in this paper, but the execution time was reduced from the scale of hours to as much as a half hour on this dataset, which allowed for comparisons for differing values of the regularization constant C. Overall a final value of 0.25 was chosen for C with a modest improvement of ~0.01 after comparing multiple SVMs with C values between 0.1 and 1.0. It should be noted that this could be considered insignificant overall, but it did result in an increase in score even with tenfold cross validation.

G. *SVM with RBF Kernel*

Overview: Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel is a variant of the SVM algorithm used for classification and regression analysis. Unlike the linear kernel, the RBF kernel maps the input data into an infinite-dimensional space, which allows it to capture more complex relationships between the data points. The RBF kernel computes the similarity between two data points based on their distance from each other. This similarity measure is then used to find the hyperplane that maximizes the margin between the two classes. SVM with an RBF kernel is widely used due to its ability to manage non-linearly separable data and its effectiveness in high-dimensional datasets. However, selecting the appropriate parameters for the RBF kernel can be challenging and may require some experimentation.

Overall results: The SVM with RBF Kernel was one of the poorer performers of the set and is comparable to the KNN and Naive Bayes classifiers. This is interesting as the SVM with Linear Kernel had significantly higher scores for every metric.

Best Performers: Models trained on fold 3 of split 0, fold 6 of split 1, fold 5 of split 2 were the best performers, with accuracy scores on the validation set of 0.534, 0.526, and 0.538 respectively. Of these, the best performer on the test set was model 3 on split 0 with an accuracy score of 0.51. Because the model did not perform any better than simply guessing a single class, it can be concluded that the model does not generalize to the dataset.

H. *Multi-layer Perceptron*

Overview: Multi-layer Perceptron (MLP) is a feedforward artificial neural network that is widely used in supervised learning tasks such as classification and regression analysis. MLP consists of multiple layers of neurons, each connected to the neurons in the previous and next layer through weighted connections. The input layer receives the input data, and the output layer produces the final output. The hidden layers between the input and output layers perform intermediate computations by applying a nonlinear activation function to the weighted sum of the inputs. MLP uses the backpropagation algorithm to adjust the weights of the connections between neurons during the training process to minimize the error between the predicted output and the actual output.

Overall results: Multi-layer Perceptron did not perform well on our dataset likely for the same reason that normal perceptrons did not work, that being contradictory data points and excess number of features.

Best performers: Models trained on fold 9 of split 0, fold 9 of split 1, fold 6 of split 2 were the best performers, with accuracy scores on the validation set of 0.545, 0.549, and 0.581 respectively. Of these, the best performer on the test set was model 6 on split 2, with an accuracy score of 0.53. Because MLP performed poorly on both the training and validating data, it can be concluded that it did not generalize.

Notes: While largely non-performant on the full dataset, in our preliminary testing we found that it actually was among one of the better performers with a significantly trimmed down dataset. This likely caused the data set to become more linearly separable, leading to better performing MLPs.

I. *Gradient Boosting*

Overview: Gradient Boosting is a machine learning model used for both regression and classification problems. It involves combining multiple weak learners into a strong learner, where each weak learner is trained to improve upon the errors made by the previous learner. The process begins with a single weak learner, and the subsequent learners are trained to minimize the residual errors between the predicted output and the actual output. The final prediction is a weighted sum of the predictions made by each weak learner.

Overall results: Gradient Boosting was one of the better performing models on our dataset and does not overfit with a max_depth of 4.

Best performers: Models trained on fold 4 of split 0, fold 9 of split 1 and fold 2 of split 2 were the best performers, with accuracy scores on the validation set of 0.632, 0.647 and 0.641 respectively. Of these, the best performer on the test set was the model 9 of split 1 with an accuracy of 0.62. Because there was a 2% drop in model performance between the validation and test set, it can be concluded that the gradient boosting classifier generalized to our dataset.

Notes: Because gradient boosting uses regression trees as the composite ‘weak models’, the same logic was used when selecting the max_depth. The loss, learning_rate, n_estimators, subsample and criterion were chosen to hasten training time without affecting model performance.

J. *Linear Ridge Regression*

Overview: Linear Ridge Regression is a linear regression algorithm that is used to model the relationship between a dependent variable and one or more independent variables. It adds a penalty term to the standard least-squares regression method to shrink the regression coefficients to zero. This penalty term helps to reduce the impact of multicollinearity in the input variables and prevent overfitting. By adding the penalty term, Ridge Regression reduces the variance of the estimates of the regression coefficients, which can lead to more stable and interpretable models.

Overall results: Linear Ridge Regression was one of the better performers and learning curves did not show signs of overfitting.

Best performers: Models trained on fold 0 of split 0, fold 4 of split 1 and fold 9 of split 2 were the best performers with validating accuracy scores of 0.618, 0.619 and 0.614 respectively. Of these, the best overall performer was model 4 of split 1, with a test accuracy of 0.60. Because there was a roughly 1% drop in model performance between the validating and test set, it can be concluded that linear ridge regression generalized to our dataset.

K. *K-Nearest Neighbors*

Overview: K-Nearest Neighbors (KNN) is a simple machine learning algorithm used for classification and regression tasks. It works by assigning a class label or a numerical value to a new data point based on the k closest training data points in the feature space. The k value is a hyperparameter that decides the number of neighbors to consider when making a prediction. For classification, the class label of the k-nearest neighbors is assigned to the new data point. For regression, the average or weighted average of the k-nearest neighbors is used as the predicted value. KNN does not make any assumptions about the distribution of the data and can be used for both linear and non-linear decision boundaries. However, the performance of KNN can be affected by the choice of k and the distance metric used to measure the similarity between data points.

Overall results: K-Nearest Neighbors did not show signs of fitting to our dataset according to the learning curves. This was likely due to the high dimensionality of our dataset. As the number of dimensions in the data increases, the curse of dimensionality becomes more severe. KNN assumes that similar points are close to each other in the feature space. However, in high-dimensional spaces, all points become equidistant, making the KNN algorithm ineffective. Other reasons such as feature scaling as well as outliers might have also played a significant role in the lacking performance of KNNs on our dataset.

Best performers: Models trained on fold 2 of split 0, fold 8 of split 1 and fold 3 of split 2 were the best, with accuracy scores on the validation set of 0.53, 0.542 and 0.535 respectively. Of these, the best overall performer was model 2 on split 0, with a test accuracy of 0.51. Because KNN performed poorly on both the training and validating data, it can be concluded that it did not generalize.

L. *Passive Aggressive Classifier*

Overview: Passive-Aggressive (PA) Classifier is a linear algorithm used in online learning and text classification. It minimizes a hinge loss function while subject to a regularization constraint. The algorithm updates its model parameters in an online manner, switching between passive and aggressive updates based on the model's accuracy. The PA algorithm is efficient and can manage large-scale datasets with high-dimensional feature spaces.

Overall results: PA Classifier did not fit well to our dataset. This was likely due to the non-linear separability of our dataset. In addition, PA classifiers are particularly sensitive to the selection of features, and our decision to choose all the available columns likely included many irrelevant and contradictory data points, leading to lackluster performance.

Best performers: Models trained on fold 7 of split 0, fold 9 of split 1 and fold 6 of split 2 were the best performers, with a validating accuracy score of 0.532, 0.551 and 0.531. Of these, the best overall performer was model 9 of split 1, with a test accuracy score of 0.53. Because PAC performed poorly on both the training and validating data, it can be concluded that it did not generalize.

VII. Conclusions

A. *Best models*

Based on the model metrics on the test set, there were 2 clear best performers: decision tree and gradient boosting, with test accuracy, precision, and recall scores all around 61-62%. However, as the decision tree classifier was a vastly more interpretable model, we concluded that it was the best overall model in predicting a user's attitude towards their star sign.

B. *Sampling:*

Based on the results, it is difficult to say which test/train split performed the best overall since the performance of the models varies across the splits. However, we can see that the models generally perform slightly better on Split 2 (80/20) compared to Split 0 (50/50), with slightly higher F1-scores for most of the models.

One possible explanation that does not apply to all the models is that a relative training set size provided the models with more examples to learn from.

In addition, a larger training set size would necessarily result in a larger variety in the data, thereby reducing the changes of models' overfitting to the training set, though none of our models' showed signs of overfitting regardless of the train/test split.

C. *Generalization*

Judging from the models that successfully fit to our data (decision tree, naive bayes, logistic regression, SVM with linear kernel, gradient boosting, and ridge regression), the models that generalized best were the decision tree models, as they performed similarly well on the validating set as on the test set. This showed that it did not simply memorize the data set or met a lucky split but captured an underlying pattern behind a user's dating profile and their attitude towards their zodiac sign.

D. *What could be done differentl?*

The single largest improvement to all our models would have been better feature selection. Many models, such as perceptron, MLP, KNN and PAC failed to fit to our dataset due to an overabundant number of features, which likely caused the data to become non-linearly separable and contributed to an increase in the dimensionality of our dataset without necessarily providing any extra actual information.

For model specific changes, we could have also spent more time on hyperparameter tuning, as models such as MLP and PAC are overly sensitive to changes in hyperparameters.

E. *Interpretation of results*

The tables below (figure 1, Appendix 1) show the main performance metrics for 11 different classification algorithms (Linear Regression not included) on 3 different train/test splits (50/50, 70/30, 80/20) of the input data. Looking at the results, we can find that the models generally perform similarly across different splits with some slight variations.

In terms of the well performing models, Gradient Boosting, Decision Tree, and Logistic Regression all perform well, with accuracy, precision, and recall scores consistently above 60% across all splits.

Some of the poorer performing models include Perceptron and SVM with RBF Kernel, which consistently achieved scores worse than simply predicting a single class. The Passive aggressive classifier also teneds to perform poorly, achieving an F1-Score below 0.4 in two of the three splits.

F. Notes on linear regression

Because our linear regression models did not work for our classification problem, it was impossible to compare their performance with the other models. In addition, it is impossible to evaluate them on the test set using confusion matrices or the sklearn classification report. For future reference, we will investigate introducing a non-linear element such as a sigmoid function to the linear regressor to make them compatible with our evaluations and like that of the Ridge Regression Classifier.

G. Gradient Boosting – An Alternative Assessment

In pursuit of better performance an alternative assessment of Gradient Boosting using a sperate feature set was performed. The intuition arose based on the concept of gradient boosting and weak leaners. Given that the feature set used in the project was reduced to an idealized set, it was (naively) considered that investigating Gradient Boosting utilizing the entire feature set could produce better results than had been produced before.

The assessment was conducted by discarding all data transformations used prior except those required to produce SSI and reduce the essays to a useful value. This data set, which contains the same number of entries, was then passed through the standard Scikit-learn Ordinal Encoder and run through the standard pipeline to examine the results on the Gradient Boosting algorithm. Of interest this produced a model (on the 0.7/0.3 split) with weighted averages reaching the 0.65 mark and a test set F1 average of 0.64. The best model chosen, Decision Tree, had a maximum output of 0.62. It was noted that this effect was only valid for Gradient Boosting, while some of the other models had reduced performance.

While this increase was mainly tied to the circumstances of the seed used for the project, the model was overall the highest in consistency all things considered as it typically averaged .63 on other seeds tested. The suspected result is that the Gradient Boosting algorithm was able to take advantage of unused features that had low, but positive correlations in a way that other models were unable to due to the effect of combining weak learners to minimize error, which is benefitted by including additional features/trees.

References

[1] Allum, N. (2011). What Makes Some People Think Astrology Is Scientific? Science Communication, 33(3), 341–366. https://doi.org/10.1177/1075547010389819

[2] Orth, T. (2022) *One in four Americans say they believe in astrology*, *YouGoveAmerica*. https://today.yougov.com/topics/entertainment/articles-reports/2022/04/26/one-four-americans-say-they-believe-astrology

[3] Kirkegaard, Emil O. W. & Lasker, Jordan (2019). Intelligence and Religiosity among Dating Site Users. Psych. 2. 25-33. 10.3390/psych2010003. https://www.researchgate.net/publication/338125762_Intelligence_and_Religiosity_among_Dating_Site_Users

[4] C. van der Lee, T. van der Zanden, E. Krahmer, M. Mos, and A. Schouten, “Automatic identification of writers’ intentions: Comparing different methods for predicting relationship goals in online dating profile texts” in *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, Nov. 2019, pp. 94–100. doi: 10.18653/v1/D19-5512.

[5] D. Boller, M. Lechner, and G. Okasa, *The Effect of Sport in Online Dating: Evidence from Causal Machine Learning.* 2021.

[6] I. Backus, "Predicting Gender from OkCupid profiles using ensemble methods" self-published, Mar. 22, 2018. [Online]. Available: https://raw.githubusercontent.com/wiki/ibackus/okcupid-gender-prediction/okcupid_gender_prediction.pdf.

[7] M. Campbell, "Investigating OkCupid profiles: Can we predict someone's city?" Medium, July 21, 2022. [Online]. Available: https://medium.com/@macrotentional/investigating-okcupid-profiles-can-we-predict-someones-city-31a4734e96dd

APPENDIX I RESULT TABLES

Split 0 (50/50)	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.59	0.59	0.59	0.59
Perceptron	0.51	0.5	0.51	0.48
Naive Bayes	0.51	0.52	0.51	0.48
Logistic Regression	0.6	0.59	0.6	0.59
SVM - Linear Kernel	0.59	0.59	0.59	0.59
SVM - RBF Kernel	0.5	0.51	0.5	0.47
Multilayer Perceptron	0.53	0.54	0.53	0.51
Gradient Boosting	0.61	0.61	0.61	0.61
Ridge Regression	0.6	0.6	0.6	0.6
K-Nearest Neighbors	0.51	0.51	0.51	0.51
Passive Aggressive	0.52	0.49	0.52	0.37

Split 1 (70/30)	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.61	0.61	0.61	0.61
Perceptron	0.51	0.49	0.51	0.46
Naive Bayes	0.55	0.55	0.55	0.55
Logistic Regression	0.6	0.6	0.6	0.6
SVM - Linear Kernel	0.59	0.6	0.59	0.59
SVM - RBF Kernel	0.49	0.5	0.49	0.46
Multilayer Perceptron	0.52	0.52	0.52	0.51
Gradient Boosting	0.62	0.62	0.62	0.62
Ridge Regression	0.6	0.6	0.6	0.6
K-Nearest Neighbors	0.51	0.51	0.51	0.51
Passive Aggressive	0.53	0.52	0.53	0.45

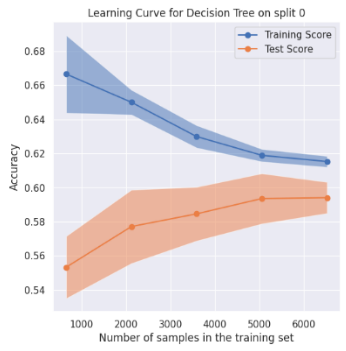
Sprint 2 (80/20)	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.62	0.62	0.62	0.62
Perceptron	0.5	0.48	0.5	0.45
Naive Bayes	0.55	0.55	0.55	0.55
Logistic Regression	0.59	0.59	0.59	0.59
SVM - Linear Kernel	0.58	0.59	0.58	0.58
SVM with RBF Kernel	0.49	0.5	0.49	0.46
Multilayer Perceptron	0.53	0.53	0.53	0.53
Gradient Boosting	0.61	0.61	0.61	0.61
Ridge Regression	0.59	0.59	0.59	0.59
K-Nearest Neighbors	0.5	0.5	0.5	0.5
Passive Aggressive	0.53	0.44	0.53	0.37

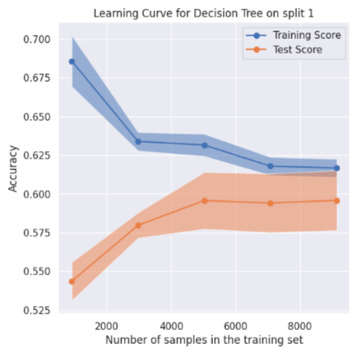
Fig 1. Result tables, for each split the best cross validated slection for every model is displayed

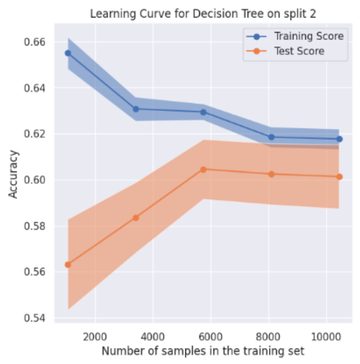
Appendix II. Learning Curves

Split 0 (50/50) **Split 1 (70/30)** **Split 2 (80/20)**

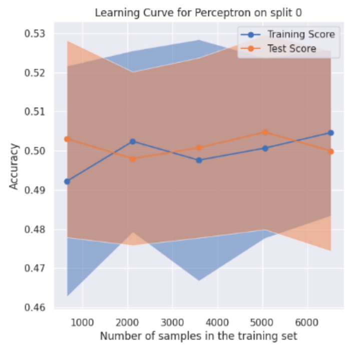
Decision Tree

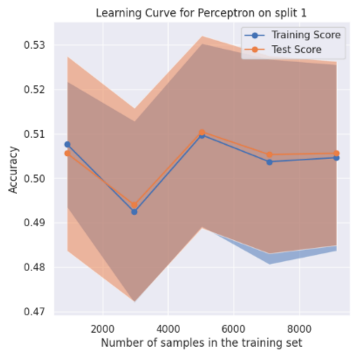


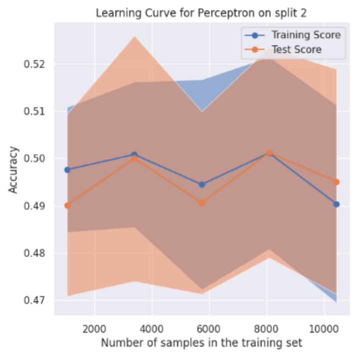




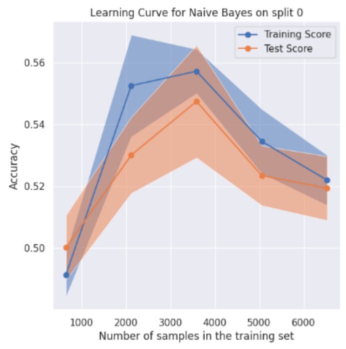
Perceptron

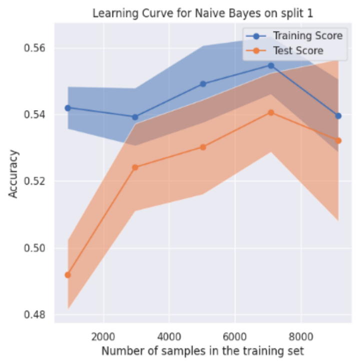


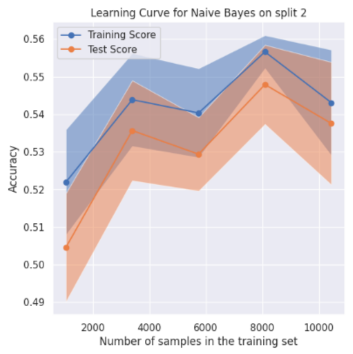




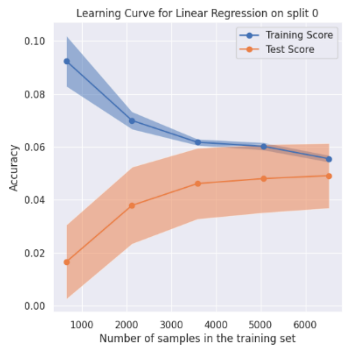
Naive Bayes

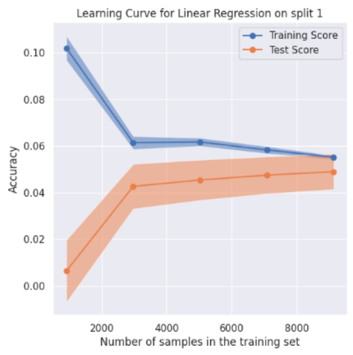


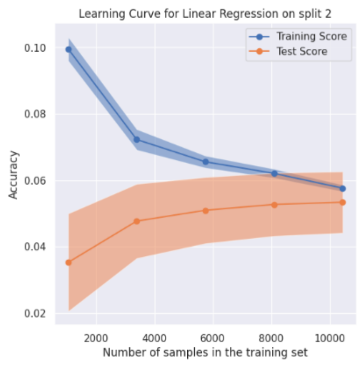




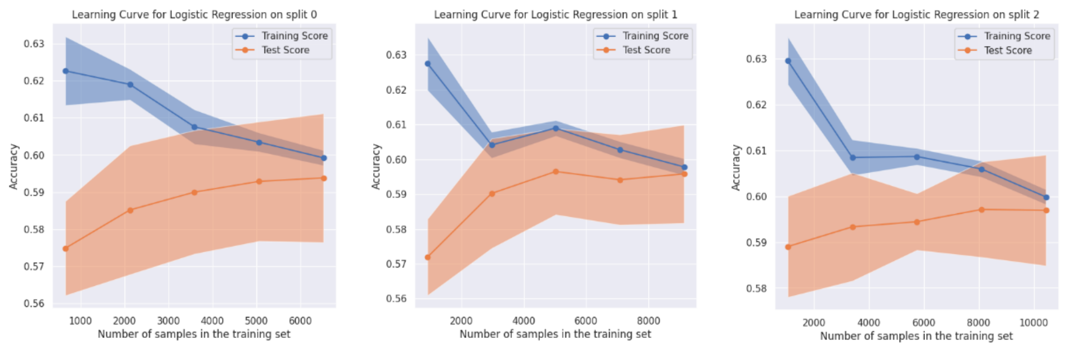
Linear Regression



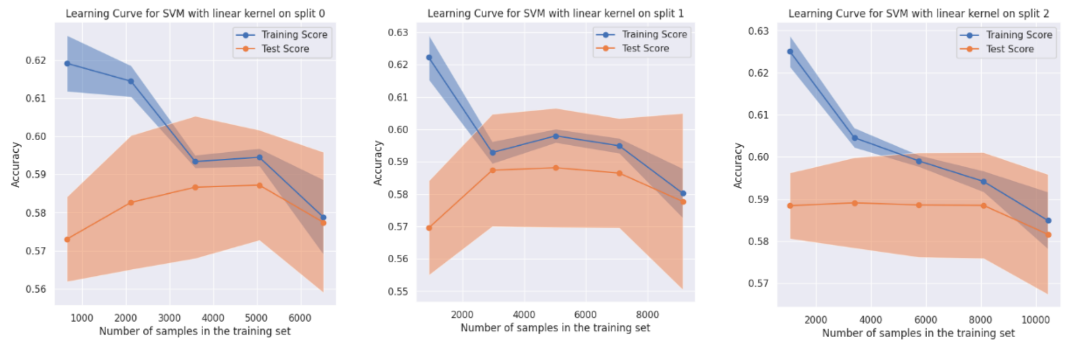




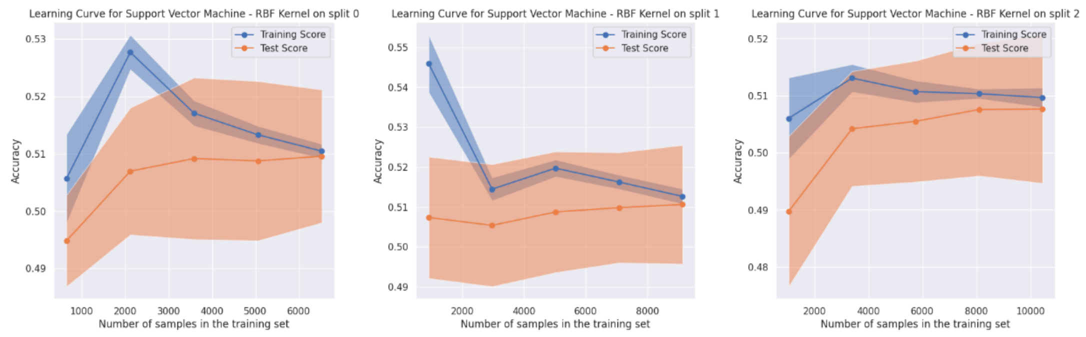
Logistic Regression



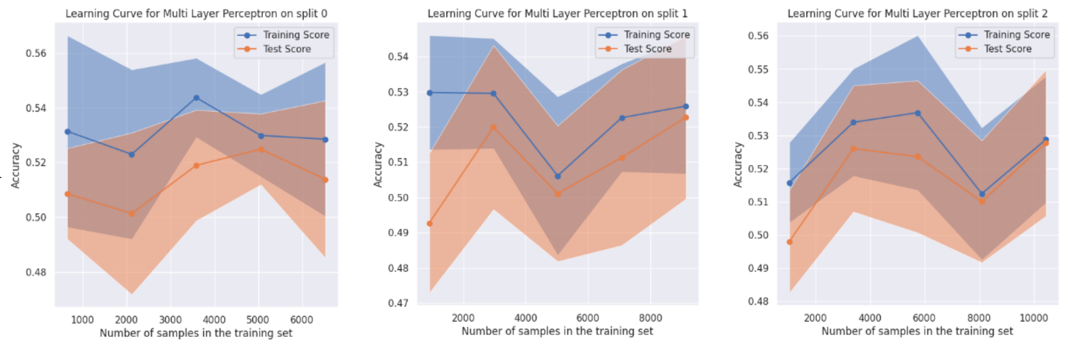
SVM - Linear Kernel



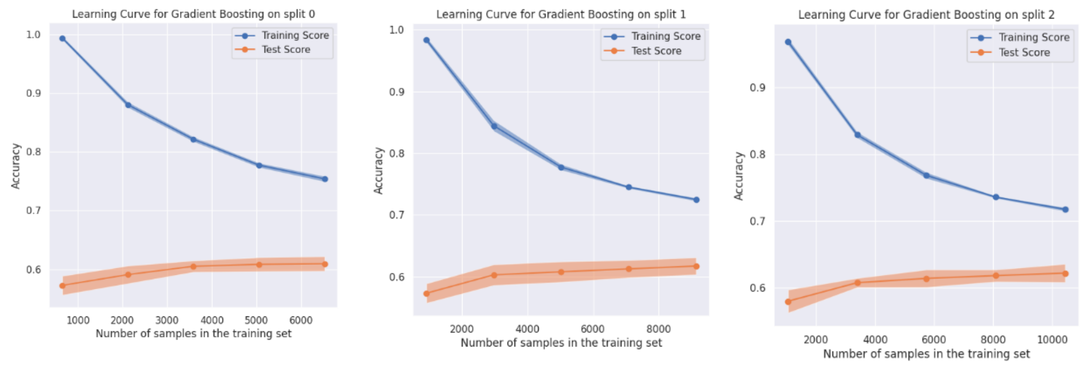
SVM - RBF Kernel



Multilayer Perceptron



Gradient Boosting



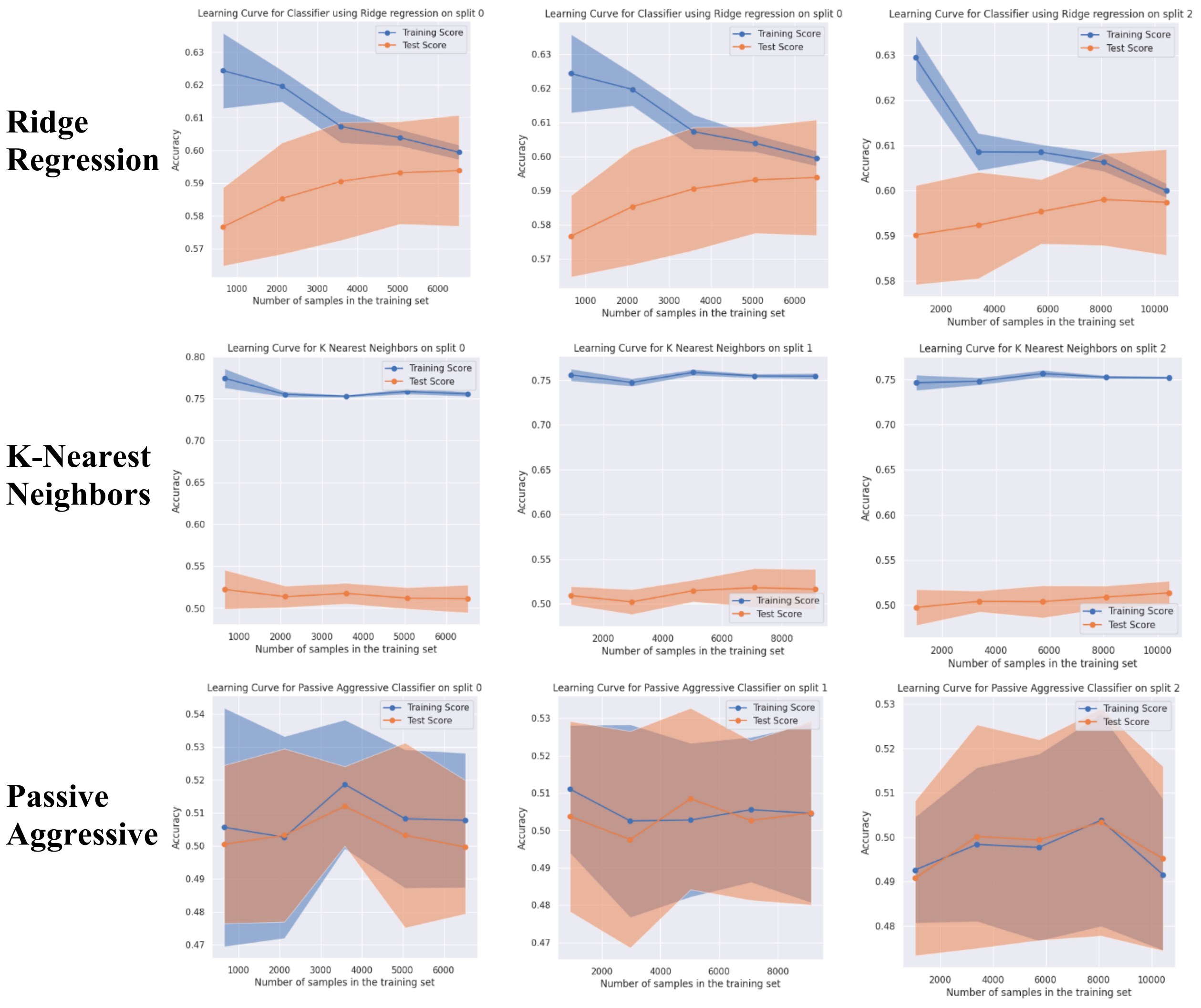


Fig 2. Learning curves for all examined models per each split