Personality Classification Modeling

The Data Science Union at UCLA

Project Leads: Karina Santoso, Madison Kohls

Project Members: Anish Dulla, Elizabeth Gallmeister, Ethan Duong, Louis Zhao, Rebecca Chua, Sarah Kosic, Sean Tjoa, Tara Jaigopal, Trina Nguyen

Abstract

- There are many patterns between a person's personality traits, demographic characteristics, and interests—can we use some of this information to predict others?
- Though there are numerous approaches to go about this, the models we chose to explore were Logistic Regression, Naives Bayes, Support Vector Machines, Random Forests, and KNN.
- Feature selection, hyperparameter tuning, and SMOTE were added to our models for optimization, efficiency, and to aid in prediction accuracy.
- The biggest challenge we faced in the modeling process was dealing with an unbalanced distribution of classes.
- We were able to achieve an accuracy of 94% for binary prediction with SVM using linear kernels with feature selection, and an accuracy of 71% for multinomial prediction with Naive Bayes classification.

Introduction

From renowned psychological assessments to Buzzfeed quizzes, predicting aspects of a person from a given set of responses has always been of great interest. These predictions can allow us to understand the behavior of an individual or group, which can: refine clinical diagnoses of psychologists, enhance advertisement strategies for businesses, and bring self-awareness to individuals. For these reasons, our team had two main objectives. First, we wanted to see if we could accurately predict different aspects of a person based on a series of responses ranging from demographic to preference-based. Second, we wanted to determine what classification models were the best predictors for certain questions pertaining to our dataset.

Data Collection and Analysis

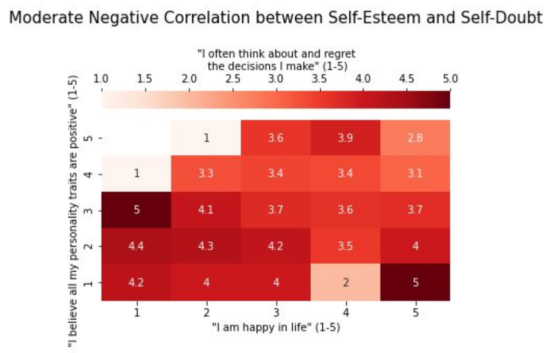
We chose to fit our models using the Young People Survey dataset from Kaggle. The dataset has 1010 respondents between the ages of 15 to 30 and recorded their responses to questions regarding their music preferences, movie preferences, hobbies, phobias, health habits, opinions, spending habits, and demographics. The majority of the questions have the respondent indicate their rate of agreement with the statement on a scale from 1-5, and some of them had the respondent pick one of multiple options to describe themselves. So, most of the 150 columns in the dataset are numerical (139), and 11 of the columns are categorical.

Data Cleaning

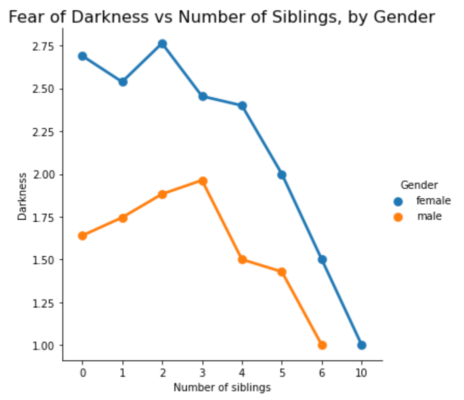
For the data cleaning, we first dropped rows with more than 3 missing values, removing 33 entries out of 1,010 in total. Then, we imputed the remaining missing values using the median for numerical measures and mode for categorical measures, modifying 303 entries. With all of the missing values removed or imputed, we were able to convert numerical measures from floats into integers, wrapping up our cleaning. For easy accessibility in future analysis, we exported the cleaned dataset to a csv file and pushed it to our GitHub repo.

Data Analysis

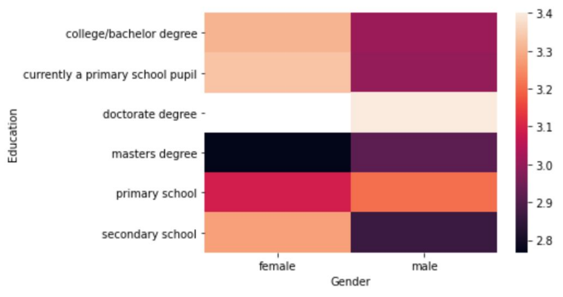
To better understand the trends and patterns in the various subsections in our dataset for modeling, we began our exploration by investigating if there was any notable correlation between question responses.



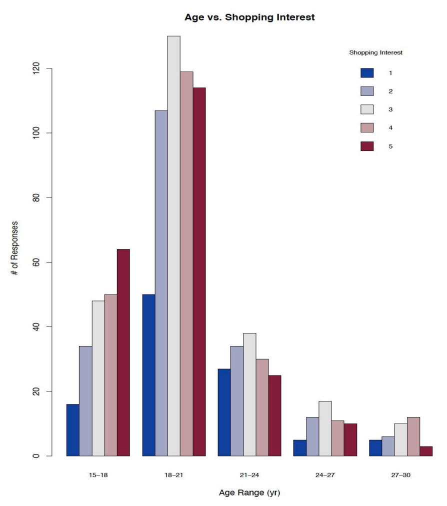
Comparing respondent's happiness and how they view themselves, we observed a moderate negative correlation between self-esteem and self-doubt. Those who said they were happy in life and believed their personality traits were positive were less likely to think about and regret decisions they had previously made.



We also found a negative correlation between a person's fear of the dark and the number of siblings a person has. Those who grew up with more siblings were less likely to be afraid of the dark. Men were also less likely to be afraid of darkness than women.



We then observed a breakdown of males and females by education level and how much these groups spend on their looks. Least surprising, it appears that females pursuing a bachelor's degree spend proportionally more on their looks. Generally, there aren't any trends between degree level and spending on looks, yet it seems that females are more likely to spend than males.



Comparing an individual's age with their interest in shopping (on a scale of 1-5) we observed a moderate negative correlation between these two variables. This means that as the individuals' ages increased, their interest in shopping decreased. The most notable drop in interest occurred between the ages of 18-21 and 21-24.

With a better understanding of the possible correlations in our dataset. We then proceeded to train the classification models to predict various dependent variables from our dataset.

Model Analysis

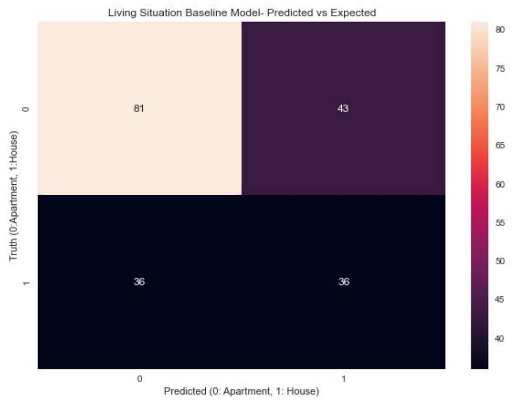
Logistic Regression Classification

Logistic regression is used to find relationships between a nominal/ordinal dependent variable and independent data under the assumption the independent data is nominal, ordinal, or continuous and contains no multicollinearity values. The model uses stochastic gradient descent—an optimization algorithm that determines the coefficients of a logistic function in order to minimize the cost function (a function which measures how poorly the model is performing). For these reasons, we were interested in seeing if a logistic regression model would be fairly accurate in predicting a binary response and a likert scale response. The only concern was with the multicollinearity assumption due to the 139 total predictor variables. To combat this, we used feature selection on both models to remove variables that did not have a high feature importance.

For our evaluation metric, what we were most interested in was how well our model could predict a response overall since there was no real "positive" or "negative" class for the model to predict. We decided to use sklearn's score method which simply returns the mean accuracy of our model's predictions.

Binary Classification: Living Situation

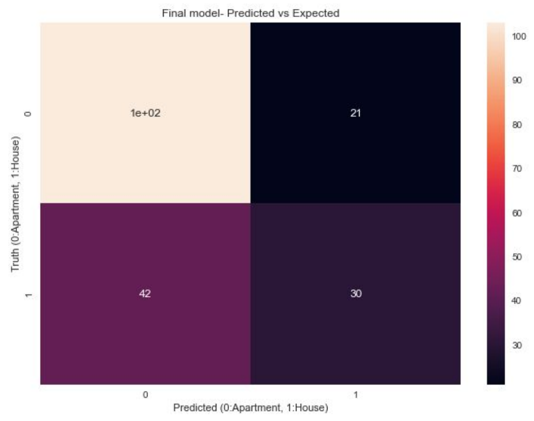
We began with a binary prediction by trying to see how accurately our model could predict whether respondents lived in an apartment or a house. We split our data using



train-test-split, setting aside 20% for testing. We then trained our model using the default hyperparameters and received a baseline score of 59%.

Looking at the confusion matrix produced by our model's predictions we observed that our model was fairly accurate at predicting those who lived in an apartment, however there seemed to be an even split between accurately and inaccurately predicting those who lived in a house.

To try and improve the accuracy of our model, we implemented RFE (recursive feature elimination) to get rid of weak features while still maintaining or improving the model's accuracy as well as increasing its simplicity. This approach recursively eliminates features while validating the model's score in order to select the best set of features. Implementing RFE with 30, 20, 15, and 10 features, we found we were able to increase the accuracy of our model to 67% by selecting the top 20 features.



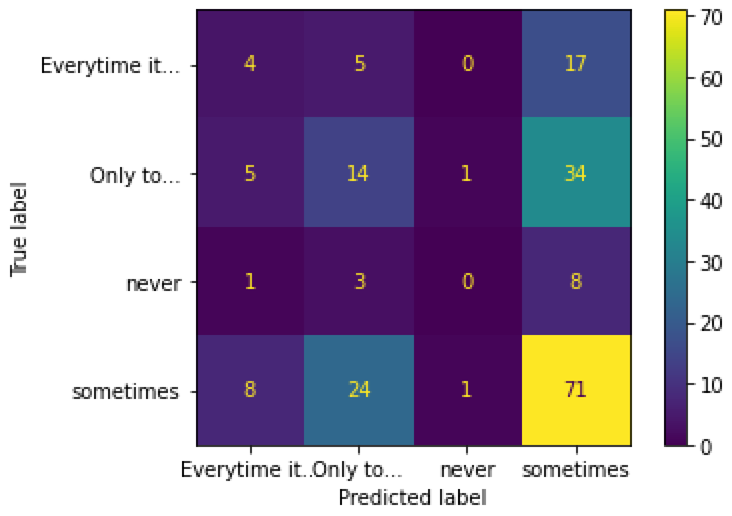
Looking at the confusion matrix of the new model, we can see an increase in accurate predictions for those living in apartments, however a slight decrease in accurate predictions for those living in homes.

One possible reason for this is that our dataset is pretty imbalanced. That is, one class takes an unequal share of the data. In this instance, 61% of the responses were for apartment, making apartment the "majority class" while only 39% were for house, making it the "minority class."

One approach we tried in order to overcome the imbalance data was regularization. By default, our model adds an L2 penalty equal to the square of the magnitude of the coefficients, shrinking coefficients evenly (the coefficients being the size and direction of the relationship between a predictor and the response variable). An L1 penalty encourages coefficients to tend towards zero, which may help feature selection with such a large number of independent variables. Creating a model with an L1 penalty and solver set to "saga", we got an accuracy score of 68%, a 1% increase from before.

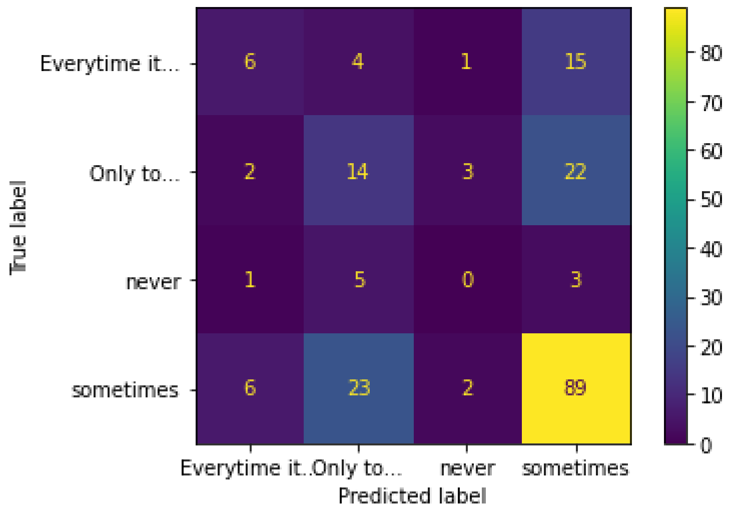
Multinomial Classification: Lying Tendencies

Next, we created a multinomial model to predict a respondent's answer to the question "Do you lie to others?" There were four possible answers to this question: "Never," "Only to avoid hurting someone," "Sometimes," and "Everytime it suits me." Similarly to the first model, we split the data using a train-test-split with a test size of 20%. The model was again trained with default hyperparameters which produced a baseline score of 43.4%.



The confusion matrix for this first model showed that the majority of both correct and incorrect guesses fall in the “Sometimes” category. As the majority of responses in the dataset fall in this category, it makes sense again that the model would choose to predict this category more often. We see that there are a lot of incorrect guesses between the “Only to avoid hurting someone” and “Sometimes” categories which also makes sense as these category descriptors are pretty similar, so respondents who pick these two options likely share many similarities as well.

We again used feature selection to strengthen the model by reducing its complexity and removing unneeded variables, but this time with the SelectFromModel method. This method takes in the logistic regression model and transforms it by discarding features with importance lower than a certain threshold. For our model, we used the mean of all features’ importances as the threshold, meaning features with importance greater than or equal to the mean were kept, while those lower were discarded.



When this was run on our original model, the resulting output chose 58 out of the total 139 predictor columns to keep in the new model. The unneeded columns were removed and then the model was retrained and tested, to get a score of 58.2%. This increase in accuracy of about 15% seemed to be a significant improvement.

Looking at the new confusion matrix for this new model, we see that there is an increase in most of the squares in the diagonal, meaning that the model made more correct predictions for all outcomes. Additionally, we see less incorrect guesses between the “Only to avoid hurting someone” and “Sometimes” categories.

A 2% increase in accuracy was seen with this model when the L1 penalty and saga solver were utilized, so similarly lowering the coefficients of less predictive features seemed to show some small improvements. Because this model used a different method for feature selection that resulted in 58 features being kept compared to the 20 in the first model, L1 regularization is likely more effective in this model due to the larger number of predictor variables.

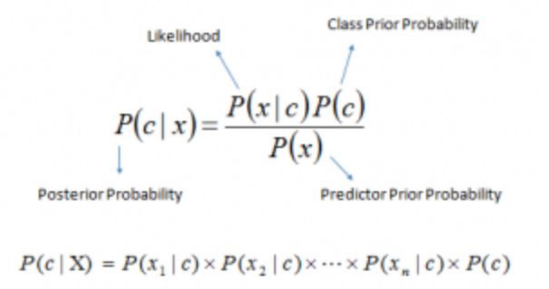
Another possible optimization would be using stochastic gradient descent.

Stochastic Gradient Descent

Gradient descent itself is an optimization algorithm where it aims to incrementally take small steps over time in order to minimize an initial cost function. The Stochastic Gradient Descent function in Python is a machine learning application of this algorithm in which it calculates the cost of one random observation from our dataset. It repeats this process until it converges to the global minimum. While sklearn's logistic regression package gradient descent, SGD uses batch gradient descent which averages the cost of every sample rather than a single random sample, which in turns provides better accuracy on smaller datasets. This is because random effects are more likely to affect our results due to the smaller training size.

Naive Bayes Classification

Naive Bayes Classifier is a set of supervised learning algorithms based on the Bayes' Theorem. One of the main assumptions is independence among all the model's predictor features. While in real life, it is unlikely that all features are independent of one another, it is still a relatively simple model to use and occasionally outperforms other classification methods.



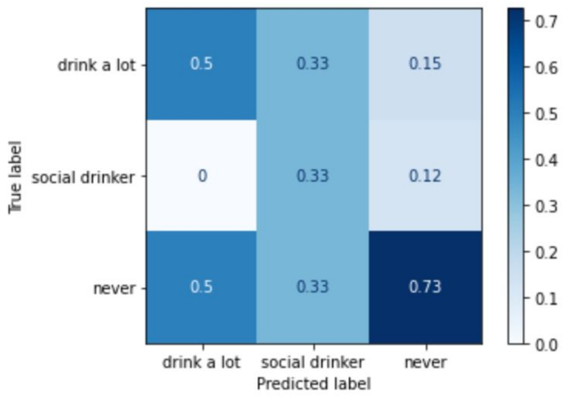
$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

We chose a Naive Bayes classification model to explore how well a model based on conditional probabilities of independent features would perform with and without feature selection. Because the Naive Bayes model works best if features are independent, we expect that our feature selection would improve our accuracy tremendously, since it would strictly limit the amount of features and create more independence, thus providing a much higher increase in accuracy score relative to other models.

Multinomial Classification: Alcohol Usage

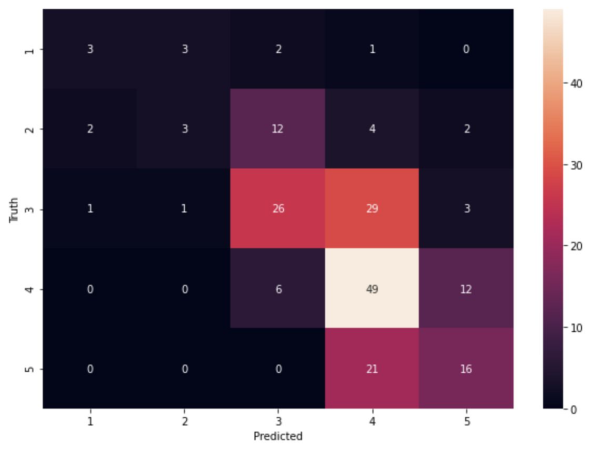
The first Multinomial NB model looked to classify an individual's alcohol usage given their responses to all other survey questions. The participants could be classified as 'never' drinking, 'social drinker', or 'drink a lot.' First, we first took our cleaned dataset, used hand-picked columns based on what personality traits we thought would be impactful predictors, and used train-test-split to split the data into training and test data, setting aside 20% of the rows for testing and setting a random state for consistent results. After applying sklearn's MultinomialNB model we received an accuracy score of 63%. To fine tune our model, we decided to include all questions and utilize feature selection. We used all columns throughout the dataset, used sklearn's Ordinal Label Encoder for categorical variables and created dummy variables to convert binary categorical variables into numerical columns. We then applied sklearn's SelectFromModel feature selection method, reducing our columns

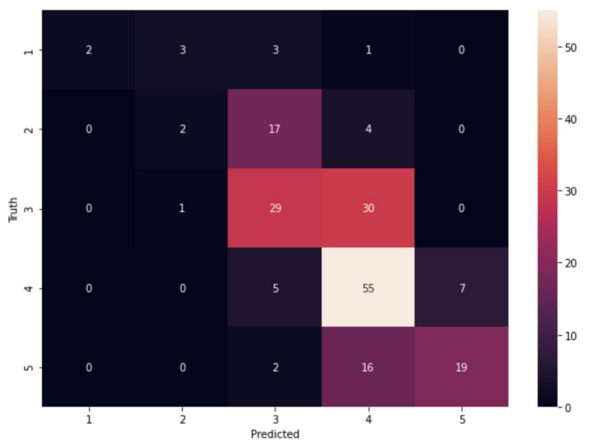


used from 160 to 34. Our accuracy score improved to 71%. The reason our accuracy improved was because restricting the amount of features reduced overlap and created more independence, which is the key assumption to the success of Naive Bayes classification models. With a significant increase in accuracy, we created a confusion matrix to visualize how our model was performing, letting us see what it was classifying right and wrong.

Multinomial Classification: Spending Habits on Healthy Food

The second Multinomial NB model looked to classify an individual's spending habits on healthy food on a scale from 1-5 given their responses to questions in the opinions category. We first subsetted data from opinions and spending habits category and transformed categorical variables into numeric values using sklearn's Ordinal Encoder. Using train-test-split, we split the data into training and testing data, setting aside 20% of the opinions data for testing, choosing our target variable to be "Spending on healthy eating", and setting a random state for consistency. We then applied sklearn's MultinomialNB model with default parameters to attain a baseline score of 49.5%. Since MultinomialNB has very few parameters to tune, we decided to improve upon the model via feature selection. Applying sklearn's SelectFromModel to our data gave us new training data that only used 23 out of 64 columns in the opinions category. By running MultinomialNB classifier on this new training set, we attained an updated score of 54.6%. The improved accuracy stems from limiting the amount of features used as predictors in the model, since the SelectFromModel chooses those of most importance that contribute to the model. Noticing that there was only a 5% increase in accuracy, we decided to take a look at the confusion matrix and compare how well the model predicted in each case.



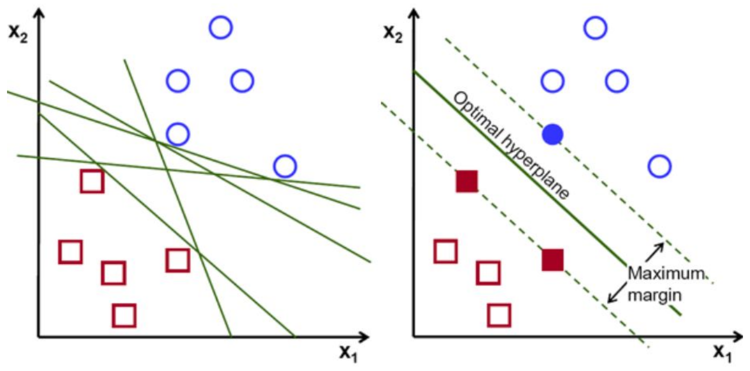


The confusion matrix to the left shows the models predictions before applying feature selection. We first observe that the responses are skewed towards the higher end, with the majority of responses in the 3-5 range. Comparing the results produced after feature selection, we note that the model slightly increased in accuracy when predicting a response of 4 but decreased when predicting other values.

The discrepancy between the two models may be due to differing predictors used. For the first model, the baseline model used handpicked features, then was improved upon feature selection on all columns in the dataset whereas the second model used all columns from the opinions section, then applied feature selection on those variables. This led to a greater increase in accuracy for the first model because running feature selection on all columns reduced overlapping features and created more independence across the whole dataset. Another reason for the discrepancy is the differing target variables. The first model was predicting more clearly defined categorical responses that were more evenly distributed whereas the second model was predicting responses on a scale of 1-5 that is up to the individual's interpretation.

Support Vector Machines

Support Vector Machines is a machine learning model that aims to separate distinct classes based on the largest distance possible between them.

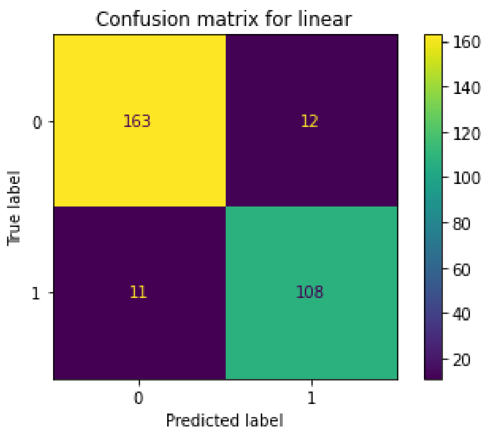


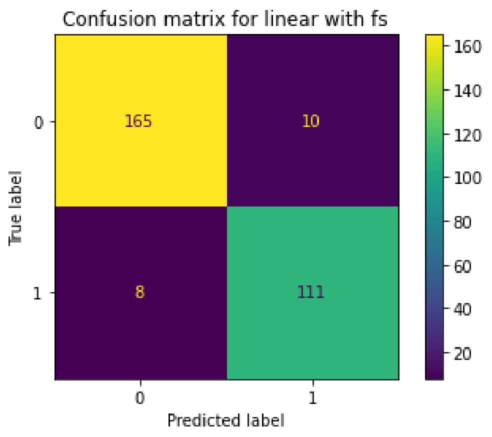
The figure on the right represents a classification model using support vector machines where the dashed lines indicate the maximum distance between the two classes. Support vector machines have a myriad of hyperparameters that affect how these dashed lines are actually determined. First, these lines of separation do not necessarily have to be linear nor all exclusive where points from different classes can exist within the boundaries of these lines. These conditions are determined by the hyper parameters of the model and can lead to underfitting/overfitting and changes in accuracy accordingly. Also, because the model does not standardize the data itself, it is very sensitive to feature scaling. Since our data is very uniform, we chose a min-max scaler to improve the accuracy of our model.

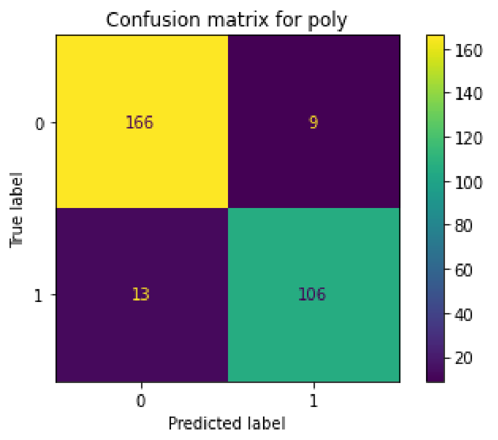
Moving onto the model itself, we tested three different versions of SVM to predict gender and happiness. These three versions included: linear kernel, linear kernel with feature selection, and polynomial kernel. We chose these three because we knew that feature selection would be very important for this dataset and wanted to see if we could create a linear model that would work better with feature selection compared to a polynomial model.

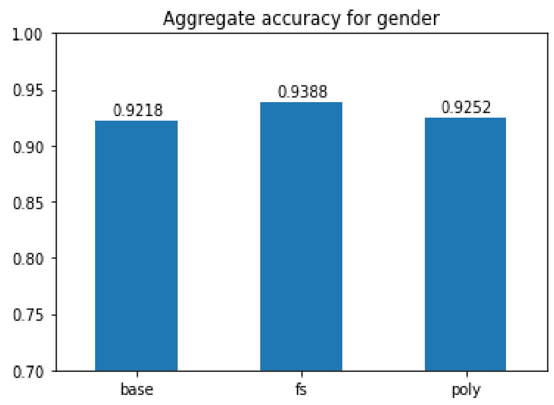
Binary Classification: Gender

For a baseline test, we tried predicting gender. Below are the three confusion matrices for gender along with their corresponding accuracies. When it came to feature selection, we used the RFE function in sklearn. The function recursively removes features every iteration by ranking the importance of each feature every iteration and removing the least important ones. To actually get the most optimal number, we ran the function over a wide range of numbers of features and chose the number of features that gave the best accuracy.



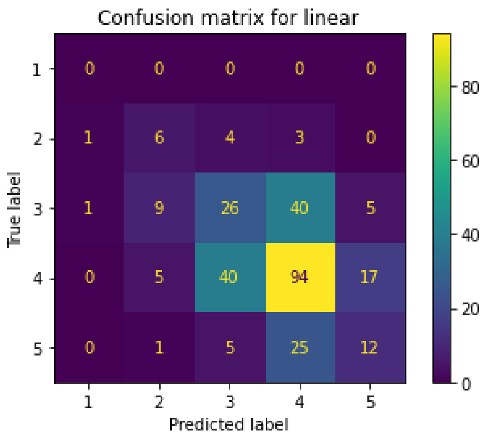


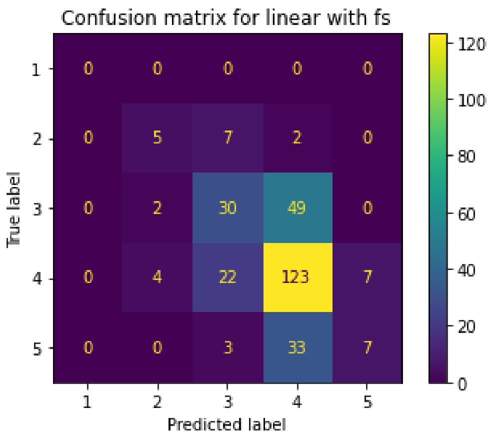


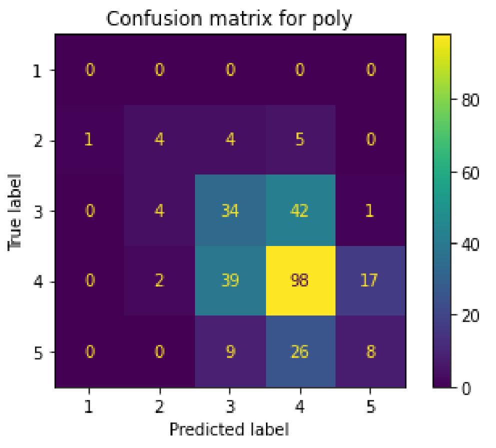


As we can see, our baseline accuracy for predicting gender was relatively high. Because of this high base accuracy, although feature selection and choosing a more precise kernel did improve the accuracy, it didn't really improve it by a very significant degree. However, this changes when we choose a more complicated variable with much more variance and categories. Below are the same plots but for the variable happiness which is from a scale of 1 to 5.

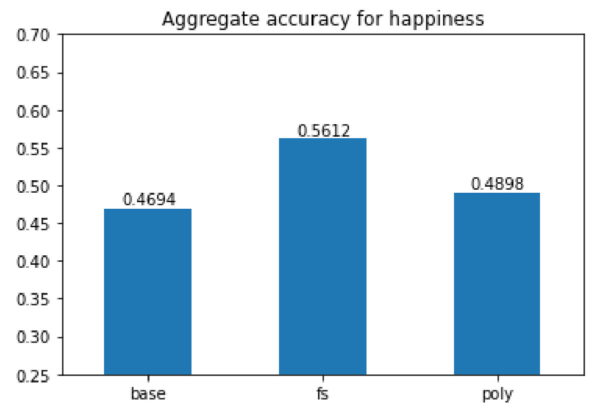
Multinomial Classification: Happiness







Unlike our SVM model for predicting gender, our baseline model for predicting happiness had a much lower accuracy. This could be due to two reasons. One could be simply that someone's happiness is a lot more complex than someone's gender. The second explanation is that instead of the two choices (Male or Female) from before, we now have five choices, a 1 to 5 scale of happiness. Then, it becomes a much harder variable to predict, because a random assignment will only net you with a base 20% accuracy rather than a 50%. As a result, feature selection becomes much more important when trying to improve accuracy. After implementing feature selection, we ended up keeping only 39 of the original 156 variables, but we improved our accuracy by 10%. This is a much bigger improvement than our previous model trying to predict gender.



Random Forest Classification

Random Forest is a supervised ensemble machine learning algorithm that constructs and aggregates the results of numerous decision trees. Random Forest usually outperforms decision trees as decision trees tend to overfit training data. Because Random Forest takes the mean or mode of multiple decision trees' results, it avoids bias in overfitting, and the number of decision trees in the model can be altered accordingly to combat bias and variance.

Binary Classification: Education

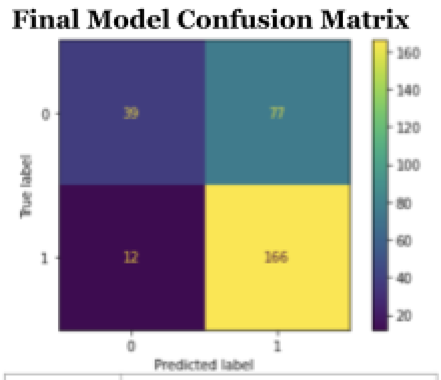
For this project, our goal was to predict respondents' highest level of educational attainment. However, during our EDA, we found that there were no clear relationships between any predictor variables and educational attainment. The variables with the highest correlations to educational attainment included more demographic information like respondents' height, age, and weight rather than their beliefs and preferences. In addition, height and weight contained some of the most missing values in our raw dataset, so our decision to impute with the mean may have had a substantial impact on our results. For these reasons, Random Forest seemed to be the best model to implement given the scope and ambiguity of our dataset.

We began modelling by predicting educational attainment as a multi-class output with six different possible responses, but our model seemed to classify all testing data as the mode of the classes. Because the classes were so unbalanced in the dataset, we decided to make educational attainment a binary output and have the model predict only whether or not a respondent's highest attained level of education is secondary school.

After making a 70/30 train-test split, we used scikit-learn's RandomForestClassifier to make our model with default hyperparameters, obtaining a baseline accuracy of 66.5%. At this point of modelling, it was reassuring to see that our Random Forest was no longer predicting only one class, but we needed to hone in our accuracy. We decided to start with feature selection, exploring RFE, RFECV, and SelectKBest from scikit-learn. Between the three, RFE and SelectKBest with Chi-squared scoring were head-to-head in the scoring of our baseline model. Unfortunately, it became clear that both feature selection methods produced rather inconsistent results in baseline model scores due to random chance, but RFE takes much longer to process than SelectKBest. For that reason, our final model trained on a dataset of features chosen by SelectKBest.

Next, we used GridSearchCV to search for optimal hyperparameters for our model. The main hyperparameters that had a large impact on our model included the number of Decision Trees, max depth of each Decision Tree, minimum samples required to split an internal node or become a leaf node, and scoring criterion. The only differences between our final model and the baseline model were the number of estimators and the minimum samples to split an internal node.

After tuning, our RandomForestClassifier achieved a mean accuracy score of 70.4% over 50 trials compared to the baseline mean accuracy score of 66.5%. In order to evaluate our model's performance more precisely, we ran a repeated 10-Fold Cross Validation and found an accuracy score of 67.6% and 62.6% for the final model and baseline model respectively.



0	Highest Educational Attainment ◆ Currently in primary school ◆ Primary school ◆ Bachelor's degree ◆ Master's degree ◆ Doctorate degree
1	Highest Educational Attainment ➔ Secondary school

Final Model	Mean Accuracy	Standard Deviation	Highest Accuracy
50 Trials	0.704	0.021	0.748
Ten-fold Cross Validation	0.676	0.089	---

Baseline Model	Mean Accuracy	Standard Deviation	Highest Accuracy
50 Trials	0.665	0.021	0.711
Ten-fold Cross Validation	0.626	0.086	---

Overall, our final Random Forest had a better prediction accuracy than the baseline, but is still quite weak. Early in our modelling, we experimented with a baseline XGBoost multi-classifier, which surprisingly was able to successfully identify respondents who acquired a master's degree but still frequently misclassified most respondents as "secondary school," the mode of the dataset. While promising, we abandoned it as we were unfamiliar with gradient boosting algorithms and opted for Random Forest. Although we have most likely not utilized the full potential of Random Forest by exploring undersampling, SMOTE, or more rigorous hypertuning, we think that if we were to continue trying to predict educational attainment, we should continue with some gradient boosting algorithm like AdaBoost, Catboost, or XGBoost. Nevertheless, Random Forest still performs decently as a machine learning classic for our use case.

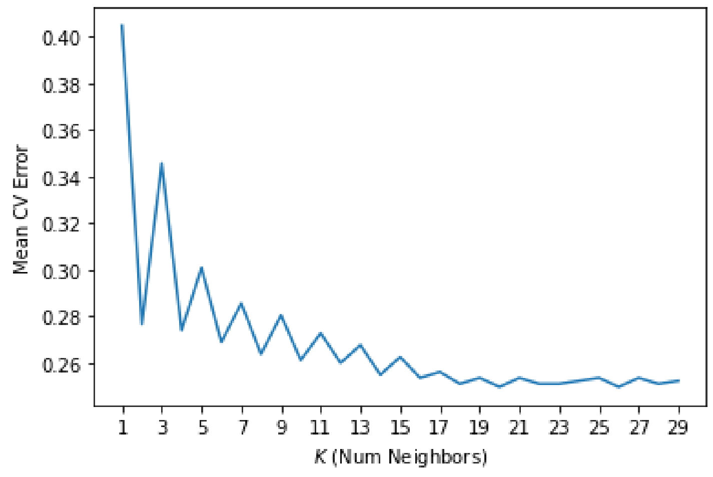
KNN Classification

KNN Classification is a supervised machine learning algorithm that classifies data into discrete categories. In our model, k-nearest neighbors calculates the Euclidean distance between each object and classifies based on the most common class among the k objects with the smallest distance. By changing the value of k, one can tune the KNN Classification model to produce more accurate results. We chose to explore KNN Classification with our model because it can easily be adapted to classify both binary and multi-class variables, as well as its intuitive implementation and tuning.

Binary Classification: Only Child

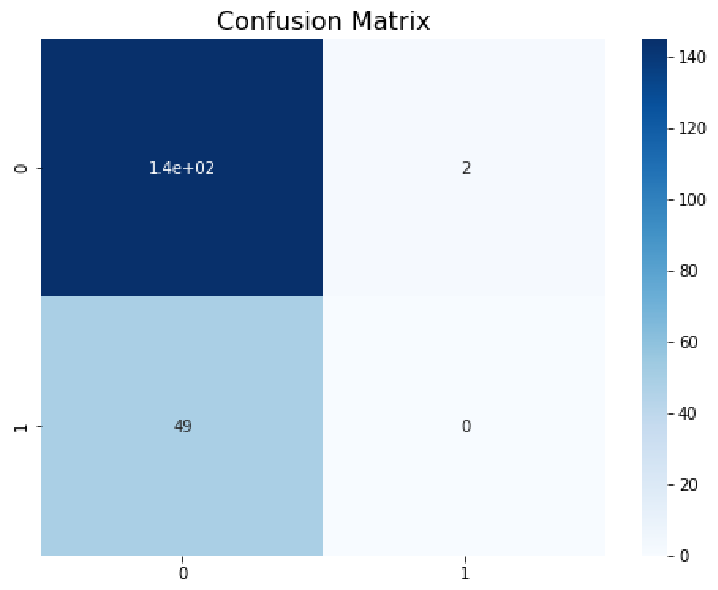
We decided to examine how accurately our model could use people's phobias/fears to predict whether or not they were an only child, which is a binary classification problem. We used all the variables within the phobias category of the Young People dataset as predictors, and the "Only child" question as the outcome variable. We split the data with train-test-split by using a random state for consistency and setting aside 20% for testing. We used sklearn's KNeighborsClassifier and trained our model with k=3 to give an accuracy of about 65% and k=5 to give an accuracy of about 67%.

Following the initial split, we then instead used k-fold cross-validation to (1) give a more accurate score for our model, and (2) help us find out an optimal k value. Since we were attempting a binary classification, it was imperative that our k value be an odd number to ensure that there were no 'ties' as the model cannot provide an output when half the nearest neighbours have a value of '1' or 'yes' for the only child attribute and the other half have a '0' or 'no'. While our original 'holdout' method depended on how the training and testing sets were broken up, cross-validation fits and scores data multiple times, with different splits of training and testing data. The mean score from these iterations should more accurately represent the accuracy of the method for new data. To find an optimal k value, we iterated through the k-fold cross-validation method with a cross-validation value of 5 (meaning it fits and scores the data five times), with k values from 1 to 29, to find the average score for each given k. We graphed the mean error (the difference between the mean score and 1), and chose the k value where the mean error plateaus at its minimum.



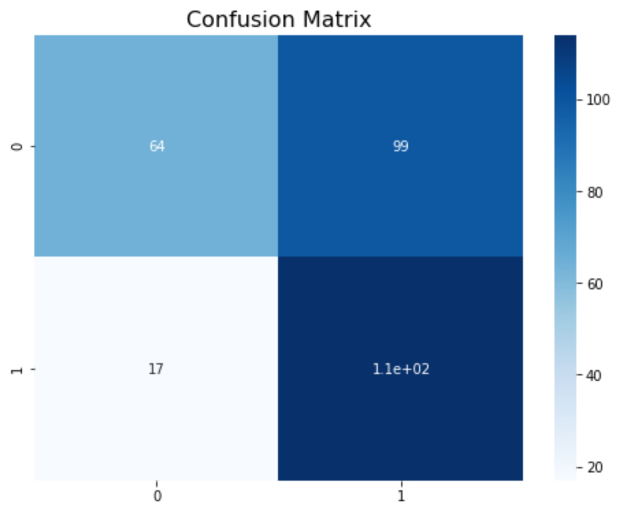
For the case of phobias predicting whether someone is an only child, the optimal k-value was 23. When we ran our model with k = 23, the accuracy improved to around 74%. However, when we looked at the confusion matrix, we saw the reason for this was mainly because the model predicted "not an only child" for 194 out of 196 individuals in our test split.

Upon further investigation, we noticed that 75% of the full set of data were not only children, so it made sense that the best model accuracy also fell around that value when it predicted that 99% of the testing set were not only children.



It became clear to us that this model was not very usable due to its tendency to predict 'not an only child' for almost every object. We wanted it to be more discerning, and thus decided to optimize the recall score or sensitivity of the algorithm. One way to interpret this in this case is the conditional probability that the predicted outcome is 'only child' given that the person is in fact an only child. For our basic KNN model, this recall score came to be 0%, which left much room for improvement.

In order to improve this metric, we decided to try and fix the inherently imbalanced nature of the dataset using SMOTE (synthetic minority oversampling technique) as previously mentioned since it was one of the simplest methods to improve our recall metric. Furthermore, research revealed that KNN classifiers based on Euclidean distance tended to benefit from this approach. We used the SMOTE() function from the imblearn package in order to generate more samples of the minority ('only child') class, after which we once again split the dataset into training and test sets and repeated the entire KNN process including cross-validation.



Since more samples were generated, it was possible to increase the value of k. The optimized value of k found by cross-validation came to be 35. Upon using this optimized model to predict the only child attribute, the model accuracy fell to around 61%, but recall increased sharply to 87% and thus the model was far more sensitive and capable of predicting the minority class accurately.

There was clearly a tradeoff between the model accuracy and sensitivity; however, we believe that using SMOTE made the model more usable since it did not blindly predict ‘not an only child’ for each object. SMOTE seems to be a promising technique which could perhaps be even more effective for a larger dataset when used in conjunction with the KNN algorithm.

Conclusion

Model Conclusions:

- Logistic Regression: RFE was used for feature selection in the binomial model, and sklearn’s SelectFromModel was used in the multinomial model. Binary classification achieved a 68% accuracy rate; multinomial achieved 58%. Uneven distribution in the data created the biggest challenge in prediction accuracy.
- Naive Bayes: Sklearn’s SelectFromModel for feature selection to create more independence among the variables and thus a more accurate model. Our first model predicted a person’s alcohol usage with 71% accuracy, and our second model which predicted one’s willingness to pay for healthy food at 55% accuracy.
- Support Vector Machines: SVM effectiveness varies largely on feature scaling and selection. Our model predicted gender extremely well (94%) but had trouble predicting multiclass variables. Feature selection boosted accuracy immensely for multiclass classification but only marginally improved binomial classification.
- Random Forest: Due to class imbalance and lack of strong relationships between predictors and response variables, our Random Forest model struggled to predict test cases. RFE and SelectKBest produced similar results in baseline accuracy scores, but SelectKBest was significantly faster, making it more efficient.
- KNN: Imbalanced distribution of the classification variable led to a relatively high accuracy but reduced sensitivity. Using SMOTE to alter the distribution of minority and majority classes improved sensitivity; however, this also led to a fall in accuracy, indicating a tradeoff between the two metrics.

Through the results of our different models, we found a pretty wide range in accuracy between models, response variables, and binomial vs multinomial models. The biggest challenge in increasing accuracy in most, if not all, the models was the imbalanced distribution of the dataset used to train the models. Since the dataset was made up of survey data, it was very natural for some outcomes to be more common than others, e.g. having siblings being much more likely than being an only child. This led to many of the models simply predicting the majority class(es) more often to increase its accuracy score, which in turn made it very difficult to increase accurately predicting classes in the minority and led to a “plateau” in accuracy scores.

Despite this challenge, there were models that were able to attain a high level of accuracy—especially with binary classification—which indicates possible relationships between personality traits, interests, and demographic information. In the future, we would look further into exploring these relationships but with a much larger and possibly more diverse dataset, as well as look into the incorporation of other machine learning methods such as xgboost and neural networks.